Qualitative and Quantitative Temporal Reasoning

Stanley Letovsky

YALEU/DCS/RR #296
December 1983

QUALITATIVE AND QUANTITATIVE TEMPORAL REASONING

Stanley Letovsky

YALEU/CSD/RR#296

December 1983

This work was supported by a grant from ITT-Programming, Stratford, CT

Table of Contents

List of Figures

1 Introduction

This paper deals a particular species of causal reasoning, which may be called *process prediction*. This is the problem of predicting the behavior of a set of mutually constraining, possibly continuously variable quantities in some situation, from a knowledge of the structure of that situation and the processes operating in it. An example which will be considered is how one might reason about the thermal behavior of a hot cup of coffee. Two different approaches to programming this kind of reasoning will be described and contrasted, and the insights provided by this exercise will be used to elucidate general issues in the design of representations for use in predictive and explanatory reasoning. A survey of AI literature on causal and temporal representation is included.

2 Two Programs

This section describes two programs that were written as part of a research effort in automated problem solving and temporal reasoning. The first program, **LESTER**, implements the organization proposed in "Reasoning About Time-Dependent Systems: A Research Prospectus" by Drew McDermott [20]. The second, **KF**, implements a theory independently proposed by Kuipers [15] and Forbus [11]. Both programs solve a similar problem. The input to the reasoner is a set of *relations* holding among a set of *quantities*, along with *initial values* for those quantities. The output is to be a representation of how those quantities will behave in the time following the initial instant. Both, therefore, involve something like a *simulation* of a system. The two approaches differ largely in the kind of information they manipulate: LESTER keeps track of the magnitudes of the quantities involved, and hence is able to make quantitative predictions. KF keeps track only of the signs of quantities, and hence its predictions are purely qualitative.

In addition, both programs are concerned with representing and manipulating incomplete or imperfect representations of situations. LESTER uses fuzzy numbers and fuzzy arithmetic to model uncertainty about numerical quantities, while KF relies on the weakness of qualitative description to capture only those aspects of a situation that are appropriate to a given level of knowledge about it.

2.1 LESTER: A Quantitative Reasoner

For representation of relations among quantities, LESTER provides a small vocabulary of devices: adder/subtractors, multipliers, integrators, comparators and delays. These devices are joined together to create networks which represent the causal relationships in some situation. For example, a cup of coffee cools at a rate proportional to the temperature difference between the coffee and the surrounding air. In LESTER this would be expressed by the following relations:

```
TEMP_DIFF   = COFFEE_TEMP - ROOM_TEMP
TEMP_RATE   = TEMP_DIFF * PROPORTIONALITY_CONSTANT
COFFEE_TEMP = ∫TEMP_RATE
```

Alternatively, the relationships could be expressed in diagram form, as in Fig.1.

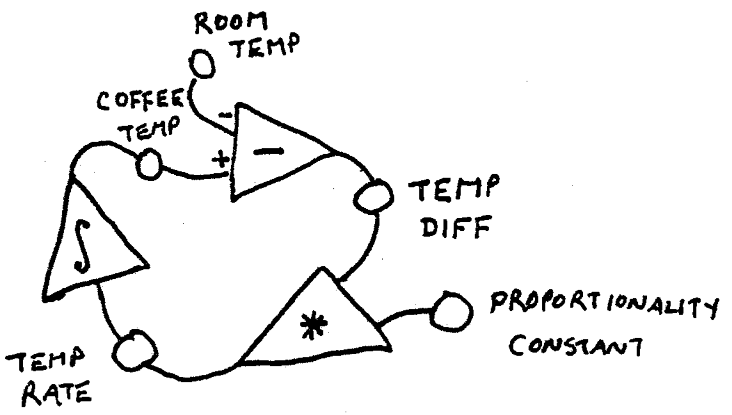

Figure 1: LESTER's Input Representation of Coffee Cooling

Note that the textual and diagram forms describe the identical internal representation. The variables in the equations correspond to the nodes in the network; in either case they are called *quantities*. The devices in the network correspond to the relations in the equation representation. We will refer to devices and relations interchangeably.

In order to operate on the above network, LESTER needs some additional information, namely the initial values of all the quantities involved, and whether or not those quantities are allowed to vary with time. These values must be numeric, but they need not be exact: LESTER operates on *fuzzy numbers*. A fuzzy number is a pair of numbers delimiting the region in which the *true value* of the quantity is believed to lie. The vaguer one's knowledge of the value is, the larger the interval of the corresponding fuzzy number will be. Thus we can give LESTER initial values along the following lines:

```
ROOM_TEMP   is constant between 18 and 22 degrees (celcius)
PROPORTIONALITY_CONSTANT is constant between -5 and -3 degrees/minute
COFFEE_TEMP is initially between 80 and 95 degrees but variable
```

This information is sufficient to allow LESTER to predict roughly what is going to happen. This is done by propagating *curves* through the network. A curve is a representation of the behavior of a quantity over time. This is to be contrasted with the approach taken by Steele & Sussman [33], who modeled algebraic relations as networks of constraints. The "devices" in their system propagated known or unknown *instantaneous numerical values* which represented the solution of the algebraic constraints on the quantities. LESTER propagates descriptions of the behavior of the quantities over spans of time, rather than instants. Its representation has an additional temporal degree of freedom/expressiveness.

Curves are considered to be the sum of a set of *curve elements*. LESTER's vocabulary of curve elements consists of constants, linear changes, exponentials, oscillations, and boundary conditions. See Fig.2. (A boundary condition specifies only the value at the initial instant, and specifies nothing about the behavior at any other times.)

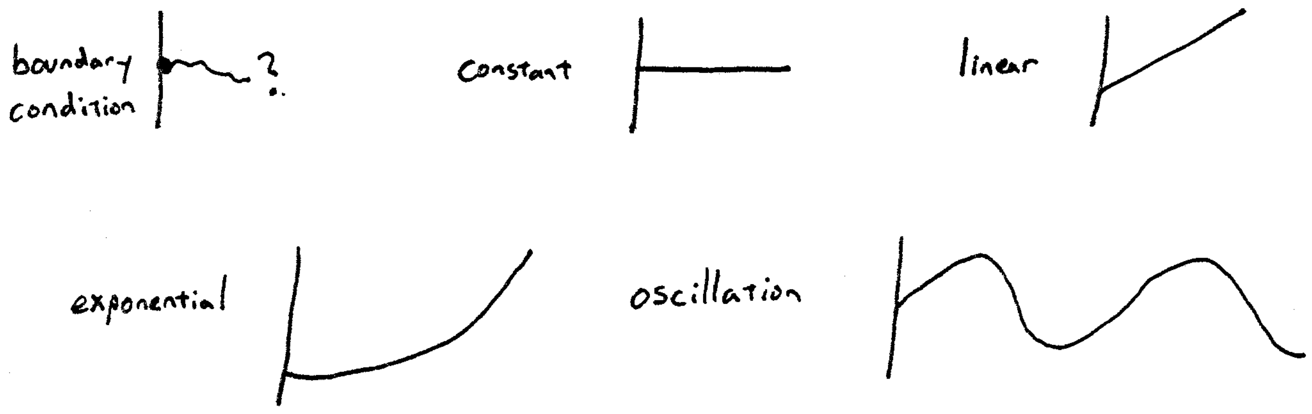

Figure 2: LESTER's Vocabulary of Curve Elements

LESTER's devices propagate information in one direction only, from their inputs to their outputs. The output is the term on the left side of the equal sign in the equation representation. A device only propagates when it has information on each of its inputs. Thus, when the values of the constants are supplied, nothing happens, because neither the subtractor nor the multiplier has enough information. When the initial value of COFFEE_TEMP is supplied, however, the subtractor can propagate. Its inputs are a curve with a single boundary-condition element having the fuzzy value [80,95] on the positive terminal, and a constant curve of fuzzy value [18,22] on the negative terminal. The result is an curve with a single boundary-condition element of value [58,77] on the output terminal TEMP_DIFF. This propagates through the multiplier in a similar fashion, and then though the integrator. But here something different happens: the integrator already has a value on its output, and LESTER recognizes that a curve has propagated back on itself.

This situation occurs because of a loop in the topology of the network, where the devices on the loop are only adder/subtractors, multipliers, and integrators.[1] Causal loops in general are a complicated problem. They come up in electronics, where, for example, the output of an amplifier is frequently "fed back" and subtracted from its input. The resulting circuit is a better amplifier than the original, but it is more difficult to analyze, and electrical engineers employ special rules of thumb to reason about them [8]. They come up frequently in the social sciences; for example in economics, increasing prices cause workers to demand higher wages, which in turn causes the corporations employing those workers to charge higher prices for the goods they produce to offset the increased cost of labor, which in turn results in higher consumer costs. This is the familiar inflationary spiral. In biology, many systems exhibit causal loops: the relationship

[1]Loops containing comparators are not really loops, because comparators output constant curves only. The form of the output curve is unaffected by the form of the input put curve. Loops containing integrators and delays together were not implemented in LESTER.

between predators and prey is an example. An increase in the population of the prey species results in fat times for the predators, resulting in increased predator population, hence increased predation, hence a decrease in the prey population.

Simple causal loops can be understood qualitatively quite well. Basically, there are two types. Positive loops, like the inflationary spiral, amplify an initial disturbance, causing an uncontrollable runaway -- colloquially, a *vicious circle*. Negative loops, such as the predator-prey balance, tend to seek an equilibrium value and stay there, resisting disturbances. If each relation in the loop can be classified as to whether its input quantity (in the loop) and its output quantity change in the same (+) or opposite (−) direction, then the loop type may be found by taking the product of the signs of all the relations.

The *quantitative* understanding of such loops is embodied in the theory of differential equations. For example, if we rewrite the third equation in the coffee example as

$$\text{TEMP\_RATE} = \frac{d}{dt}\,\text{COFFEE\_TEMP}$$

and algebraically eliminate the TEMP _RATE and TEMP _DIFF terms, we have

$$\frac{d}{dt}\,\text{COFFEE\_TEMP} = \text{PROPORTIONALITY\_CONSTANT} * (\text{COFFEE\_TEMP} - \text{ROOM\_TEMP})$$

which is a first-order differential equation of the form

$$\frac{d}{dt}\,f(t) + c1*f(t) + c2 = 0$$

It is *first*-order because the loop contains a single integrator; two integrators gives a second-order equation, and so on. *Solving* a differential equation means finding a function *f* which can substitute into the differential equation so as to make it true. This can be done for some first and second order forms, but there is no known general technique.

The problem is multiplied in the extreme when networks with more than one loop are considered, as for example, in any reasonably detailed model of the economy. Here, the limited predictiveness of qualitative techniques fails us utterly, or rather, we can find qualitative arguments in favor of many contradictory predictions. Differential equations are likewise of little use, because the equation generated by such networks are not usually solvable. The only remotely general technique for producing quantitative solutions is by numerical approximation. When there is uncertainty about the parameter values, numeric techniques must be coupled with sensitivity analysis to determine the dependence of the form of the solution on the parameter values. These techniques have been employed in circuit analysis programs [29].

How does LESTER deal with loops? Recall that a curve had propagated to the input of a device which already had a curve on its output. In the general case, LESTER need only pretend to propagate the input curve, and verify that the result is consistent with the curve that is already there. They will not usually be equal, for arithmetic with fuzzy numbers tends to compound error, so the likelihood is that the propagated curve will be compatible with but fuzzier than the curve that is already there. If, however, the curves contain boundary-condition elements, LESTER applies a different technique: it attempts to formulate and solve the differential equation describing the loop. If it is successful, the result will be a completely specified curve with no boundary-condition elements. The only type of differential equation LESTER can handle currently is first-order linear. Fortunately, this is sufficient to handle the coffee cooling example, whose solution, the curve describing COFFEE_TEMP, is an exponential decay from the coffee's initial temperature of [80,95] down to an equilibrium with the room temperature of [18,22]. This curve is then repropagated about the network, and when it arrives back at COFFEE_TEMP, it is the same curve, neglecting fuzz differences, and propagation ceases, having determined the future history of all the quantities from the initial instant out to infinity. (See Fig.3)

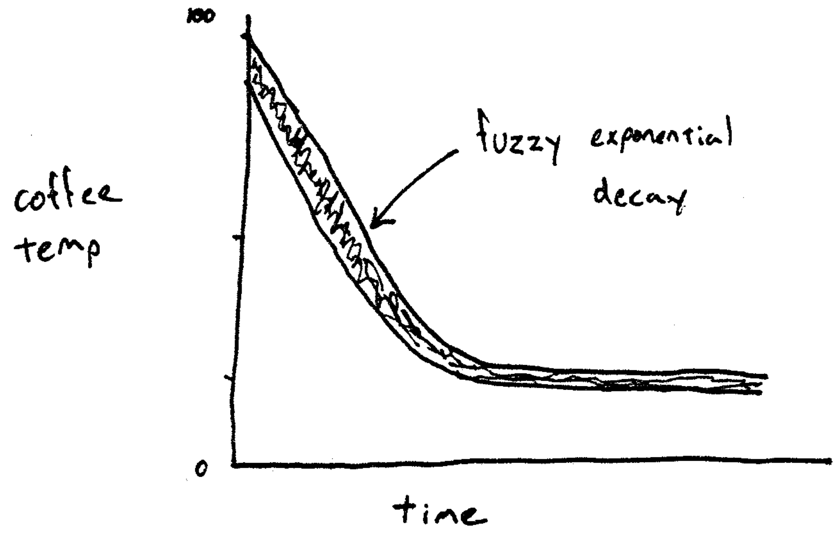

Figure 3: LESTER's Output Representation of Coffee Cooling

We have exercised only a fraction of LESTER's capabilities with this example. In addition to its ability to handle continuous change by propagating representations of curves, the program can model the effects of discrete, instantaneous events. For example, if the coffee were being kept warm in a heating pot, there might be a heating element which switched on when COFFEE_TEMP passed below a certain point. To represent this, we would have to augment our coffee cooling network with some additional devices. First, we would need to detect the event of the temperature dropping below the threshold. This is done with a *comparator*. A comparator has an output and two inputs, one labelled $>$ and the other $=<$. The output of the comparator is 1 when the $>$-input is greater than the $=<$-input, and 0 otherwise. If we attach COFFEE_TEMP to the $=<$ input of a comparator, and a threshold constant, say [65,70], to the $>$ input, then the output of the comparator will be 0 initially, but as the coffee cools its temperature will eventually fall below the threshold and the comparator output will go to one.

The implementation of this goes as follows. The initial values propagate in the manner that I have already described. The comparator ultimately receives two curves on its inputs, a constant and an exponential decay. It computes its immediate output — a constant 0 — and it also checks to see whether its inputs will cross in the future. If they will, it computes the fuzzy instant that the cross will occur, and schedules an *event* for that instant. The existence of scheduled events adds a new level of simulation above the curve propagation already described. This level is called *event-driven simulation* and is frequently used in simulating logic circuits. After all the curve-propagation has settled down, the simulator will take the next scheduled event from the queue and begin a new wave of propagation. Events are caused by comparators changing state or by previous events propagating through delays. The new wave of propagation begins at the device that scheduled the event.

In our example, the output of the comparator is not yet connected to anything. Conceptually we want to couple the downward threshold cross of COFFEE_TEMP with the turning on of a heat supply to the coffee. This can be modeled by multiplying the comparator output by a constant, the heat-production-rate of the heating element, and having this rate be added to the cooling rate determined by the temperature difference. COFFEE_TEMP will then be the integral of the NET_RATE. The resulting network is shown in Fig.4.

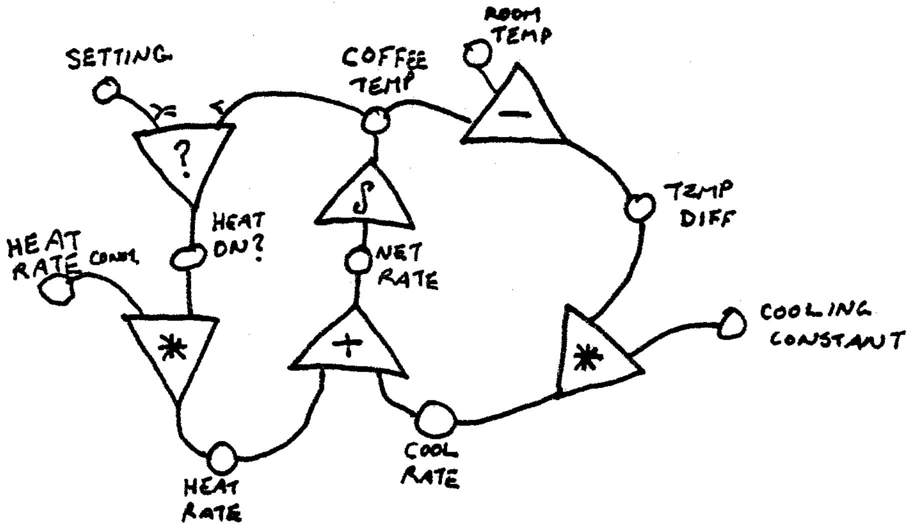

Figure 4: Input Representation of Thermostatically Controlled Coffee Heater

This simulation begins with initial conditions being supplied as before. These propagate about the network; the comparator, being initially off, has a 0 output, and the additional components therefore have no effect. The comparator schedules a threshold-crossing event, however, and after the first wave of propagation, the simulated time is advanced to that event, and a new wave of propagation begins with a 1 output from the comparator. This is multiplied by a heating constant to produce a heating rate, which in turn is added to the cooling rate from the loop, which then propagates around the loop, and results in another differential equation solution. The result is an upside-down exponential decay, rising to an equilibrium at a point where the cooling rate is equal to the heating rate. This will get propagated through the network, and the comparator will schedule another event, representing the upward intersection of the new curve with the temperature threshold. After that, the system is qualitatively in the same state as when it started. *This system therefore oscillates about the threshold.*

LESTER was intended to be the process-predicting component of a larger problem solving system, and it would be of little use to that system if it continued to simulate an oscillation forever and never reported back. What we want is for LESTER to make the cognitive leap that is italicized at the end of the previous paragraph: to summarize the repeating pattern with a single assertion. It turns out that the only way oscillations can occur spontaneously in LESTER is if the network contains loops with comparators or delays. These devices are necessary because they are the only ones which can schedule events. The topological requirement stems from the fact that for oscillation to occur, each event must cause another event: thus, the output of the event-scheduling device must be coupled in some way back to one of its inputs. If an oscillation is truly periodic, i.e. each cycle is identical to one before except for fuzz differences, then it can be detected by a fairly straightforward pattern-matching technique applied to the output of the event-scheduling device. The technique relies on the fact that a cycle of an oscillation must consist of an even number of events. Some small integer, eg., 3, is chosen as the minimum number of cycles which must exist before an oscillation will be noticed. Then, by matching the current curve against the curves 2*3*i, i=1,2,3... events back in the device's history, one can detect oscillations whose cycle-length is 2i events long. The matching must say whether the current curve could be a fuzzier version of the original, for fuzz will have increased over the course of the oscillation.

Comparators and delays, then, have three duties in the course of propagation. They must compute their immediate output, they must schedule events if necessary, and they must determine whether or not they are in a state of oscillation. If they are, then they may redescribe their output as an oscillation. (Recall that oscillations were part of the vocabulary of curve elements.) This should propagate through the oscillating part of the network without giving rise to any more events internal to the oscillation. This is because the various devices know how to propagate oscillations. For example, a comparator with one input oscillating about a constant second input does not compute its current value and schedule an event as described earlier; rather, it computes an oscillation as its output, and no events are scheduled. Thus, once an oscillation in one part of the network is detected and redescribed in terms of the oscillation curve element, and this has propagated, the periodic behavior which had previously manifested itself as events constantly being added to the event queue is then manifested in the *form* of the curves describing the device behaviors.

Following such an oscillation redescription, if there are any other processes operating that will alter the oscillation eventually; for example, if running out of fuel will eventually stop a furnace/thermostat system, then this event will manifest itself next. In our coffee example, once LESTER had redescribed the system's behavior as an oscillation, the simulation is complete.

In summary, LESTER consists of the following components: a set of propagation routines for each of the devices in its vocabulary, a differential equation solving routine, and a next-event queue and associated controller. I will now consider certain criticisms that can be made of LESTER's approach to process prediction.

The major criticism concerns the description of the behavior of quantities with the vocabulary of mathematical analysis. LESTER describes any continuous behavior as a sum of a constant term, a linear term, an exponential term, and any number of sinusoidal terms. There are three problems with this.

The first stems from from a concern with cognitive fidelity. LESTER's curve elements seem unlikely as components of naive human reasoning. Exponentials and sinusoids are something that educated people are (sometimes) taught about in school, not something they are born knowing or that they pick up on the playground. McDermott [21] has argued that the ability of people to physically solve problems which could be modeled mathematically with these concepts – for example, that one can coordinate one's actions to catch a bouncing ball, which requires extrapolating the behavior of an oscillation – indicates that they have, at least, a hardware level awareness of them. This is by no means the only or even the best explanation of this sort of ability. Servo-control and piecewise linear approximation spring to mind as alternatives. Even if humans did have a general-purpose analog computer for muscular coordination, however, the conclusion that this hardware-level awareness is accessible to symbolic reasoning is unwarranted.

Whether cognitive fidelity is worth worrying about is a point of debate that is as old as AI. Two examples of researchers who thought it wasn't and changed their minds spring to mind: Brown & Burton and the SOPHIE [3] electronics tutoring system, and Pople and the INTERNIST [24] medical diagnosis system. In both systems, although the cognitive fidelity of the representation was originally felt not to be an important issue, the desire to achieve a constructive human-system interaction led to the conclusion that the system needed to be able to describe its reasoning in the same vocabulary as the person it was interacting with used. If the LESTER-type process is buried deep in an intelligent agent and never needs to interact with anyone but its fellow modules, it may yet be reasonable to ignore this lesson, though.

The second objection is that even on its own terms the analytic representation is extremely limited: it is very easy to generate with LESTER's vocabulary of curves and devices, situations which LESTER cannot handle because it lacks the math. For example, second-order differential equations, which describe many familiar physical phenomena such as oscillating springs, will cause LESTER to excuse itself and give up. There is no representation for polynomials above first order, thus LESTER cannot handle problems involving falling bodies, which involve quadratic (2nd order) equations. Some of these issues could be solved by plugging in yet other mathematical techniques. Others cannot be solved by any existing mathematical techniques. For example, it is impossible to predict (analytically) if and when an exponential and a linear curve will cross. Thus the system is far from closed in the sense of being able to solve all or even most of the problems it is capable of representing.

The third objection is that the requirement that all relations among quantities be represented analytically is extremely stringent. While it permits LESTER to estimate not just *that* the coffee will cool but *when* it will reach room temperature and the shape of the cooling curve, it could not do anything with the information

```
The coffee is hotter than the air.
The coffee is in physical contact with the air.
Physical contact between objects at two different temperatures
 produces a flow of heat from the hotter object to the cooler.
```

This qualitative information is sufficient to predict that the coffee will cool. It does not tell us the exact shape of the cooling curve or permit us to predict when it will reach a given temperature, but it does allow some valid conclusions to be drawn from a description that is too weak for LESTER to work with. Although the analytic representation gives more precise answers when it gives answers, it cannot give any answers at all without extremely explicit information, information that humans do not need to reason about things.

The use of fuzzy numbers was an explicit attempt to permit LESTER to operate on low-grade information. Unfortunately, fuzzy quantitative information is still quantitative information. It permits imprecise knowledge, but not missing knowledge. The argument might be made that missing knowledge is just extremely imprecise knowledge. This is not true in situations like the above example, where one knows that one quantity depends on another, but one knows nothing about the form of the dependence, other than that it is monotonic. Such situations are common, especially in domains with many complex interactions, such as economics. Moreover, fuzzy numbers carry along a host of problems of their own which can introduce considerable spuriousness into the computations. For example, fuzzy multiplication and division are not inverses; thus in computations involving oscillations it is necessary to be careful about whether one wants to, say, multiply a number by the period or divide by the frequency. The solution of differential equations involving fuzzy numbers is a tricky issue. In LESTER, the fuzzy values of the coefficients of the solution were computed using a Monte Carlo method: the values of the parameters of the equation were allowed to vary randomly and simultaneously within their fuzz ranges, in the hope that the solution coefficients would tend to be pushed towards the limits of their fuzz ranges. The extremes encountered by each coefficient were kept track of, and after several repetions of the computation, became the fuzzy values for the coefficients. This method is probably an inadequate substitute for a sensitivity analysis, and would be a mess to program for differential equation orders of higher orders than the first, which have solutions of different forms depending on the parameter values.

When LESTER encounters an uncertainty that could lead to qualitatively different predictions, it is supposed to split the simulation and follow each one separately. I never implemented this.

Fuzzy numbers have an extremely insidious tendency of generating this situation in the course of computations which seem straightforward from the point of view of ordinary arithmetic. This stems from the fact that simple trichotomy in ordinary arithmetic, the fact that any number is either less than, equal to or greater than any other number, becomes enormously complicated for fuzzy numbers. In addition to the three normal relations, two fuzzy numbers may overlap in several distinct ways. Moreover, equality among two fuzzy numbers does not necessary imply equality of the quantities they represent. Therefore unless the two fuzzy numbers being compared have no points in common in their respective intervals, the simulation must be split into three. Each branch considers one of the possible relations between the real values of the two numbers. In the paper which proposed LESTER, McDermott argues that by not keeping track of quantitative information, qualitative reasoning systems become prone to worrying about alternatives which simple computations could easily rule out [20]. Thus quantitative information was supposed to prevent needless worrying about spurious alternatives. This did not seem to happen in practice. Fuzzy quantitative reasoning detects all of the possibilities detected by qualitative analysis, as well as others that qualitative analysis is capable of suppressing.

This problem might be improved if the system were able to take more information into account when operating on two fuzzy numbers than just their interval boundaries. Examples of such additional information might be that number A was constrained to be equal to number B, or that number A was derived from number B by multiplying by a non-fuzzy constant greater than 1, so that A must be greater than B even if their intervals overlap. One way to keep track of this information might be with a system of data-dependencies on both the qualitative and quantitative relations among the quantities, along the lines of [19].

It may be surprising to some that fuzzy numbers are such a source of trouble. Davis [5] reports reasonable success in using them to represent imperfect knowledge of spatial relations. I believe that the answer lies in the qualitatively different nature of the manipulations that Davis was performing, which in turn depends on the qualitatively different nature of the questions which temporal reasoning tends to ask as compared with spatial. Davis was concerned with integrating separate fuzzy estimates of a distance to come up with a best guess at the actual value of the distance. That is, he was combining multiple sources of information to produce similar information of enhanced quality. The situation in LESTER is quite different: each fuzzy time estimate represents a single computation lineage. The longer that computation, the more fuzz has been introduced into the estimate. There is no other information to shore it up. Moreover, the salient feature of time, as opposed to space, is its theoretical complete ordering. Many of the inferences one wants to draw in the temporal domain have to do with relative precedence. Thus the operation to which these estimates are frequently subjected is comparison, and fuzzy comparisons, as we have seen, are fraught with difficulties.

One of the motivating factors in LESTER's design was that the reasoner should be capable of *understanding* the system it was reasoning about, as opposed to merely simulating it. As McDermott put it,

> In both numerical and qualitative simulation, the problem is the same: the computer is merely "reliving" a sequence of events, not really "understanding" it.

[20] McDermott gives two examples of what this difference actually amounts to. One is the issue of detecting oscillation and "jumping back" to redescribe it as oscillation, as described earlier. The other is being able to abstract away irrelevant details of the network; for example if some large subnetwork is behaving in a simple way, it could be replaced by a simpler *equivalent network*[2] with the same behavior, effectively focusing the system's attention on the remaining portion of the network. These techniques may be called *temporal* and *topological abstraction*,

[2]Analogous to the voltage and current equivalent circuits that are such useful abstraction tools in the analysis of electrical circuits

respectively. The former was implemented: a simple kind of pattern recognition which looks for oscillation in the behavior history of a comparator output is all that is needed. The latter was not implemented, except insofar as inclusion of comparators in the device vocabulary allows certain kinds of abstraction to be built into the network. In the extended coffee cooling example, the additional part of the network has no effect when the comparator is off. Thus the network structure itself causes that irrelevant portion of the network, the heating element which is off, to be "removed from consideration". In fact, it is still considered, but it propagates zeros. Thus comparators can be thought of as mediating between different abstractions of the system, switching them in and out as they change relevance. None of this is really LESTER's doing, however. LESTER is not analyzing topologies or replacing networks with their equivalents. It seems, in fact, rather irrelevant to LESTER's approach: if the original subnetwork can be solved, then why not just solve it? If it can't then an equivalent network couldn't be either. Abstraction involves using qualitatively simpler descriptions to arrive at qualitatively simpler (but relevant) conclusions. LESTER has no use for the qualitatively simpler: it is an all or nothing approach.

I will now discuss another program I have written, based on the work of Kuipers and Forbus, that reasons in a purely qualitative way.

2.2 KF: A Qualitative Reasoner

Like LESTER, KF provides a vocabulary of relations or devices by which the causal dependencies of a situation can be represented. These relations are addition, multiplication, integration, and positive and negative monotonicity with or without a common zero. Unlike LESTER there is no directionality to these relations, so the inverse functions, subtraction, division and differentiation are equally available. KF does not keep track of the magnitudes of the quantities it reasons about, only their signs and the signs of their derivatives. Each quantity in the input is considered by the program to have two *aspects*: its *level* and its *derivative*. Each aspect takes on values drawn from a four-valued qualitative arithmetic system. These values are *unknown* (?) , *negative* (−), *zero* (0) and *positive* (+), and are here called *quits* by analogy with the *bits* of two-valued arithmetic. The addition and multiplication tables for this system are as follows:

+	-	0	+	?
-	-	-	?	?
0	-	0	+	?
+	?	+	+	?
?	?	?	?	?

x	-	0	+	?
-	+	0	-	?
0	0	0	0	0
+	-	0	+	?
?	?	0	?	?

Each relation among quantities in the input representation is mapped onto one or more relations in the internal representation, as follows:

INPUT RELATION	LEVEL	DERIVATIVE
$C = A + B$	$C = A + B$	$C' = A' + B'$
$C = A * B$	$C = A * B$	$C' = A*B' + B*A'$
$A = \int B$		$A' = B$
$A\ M^+\ B$		$A' = B'$
$A\ M^-\ B$		$A' = -B'$
$A\ M^+_0\ B$	$A = B$	$A' = B'$
$A\ M^-_0\ B$	$A = -B$	$A' = -B'$

The internal representation therefore consists of aspects related by the relations + , * , unary − and = . The equality denotes equality of quits only, and not of the quantity values they denote.

Each quantity therefore is represented at any one time as a pair of quits, one for its value and one for its derivative. Many people instinctively regard such a simple description as having impoverished expressive power. This is an error. To get a sense of what can be captured by such a simple description, consider Fig.5. These curves are generated by assuming that both the first and second derivatives of a quantity are known. The first and second derivatives of a quantity will always be explicitly represented when any other quantity is dependent on its first derivative. If either value is not known, its curve representation may be thought of as a disjunction of

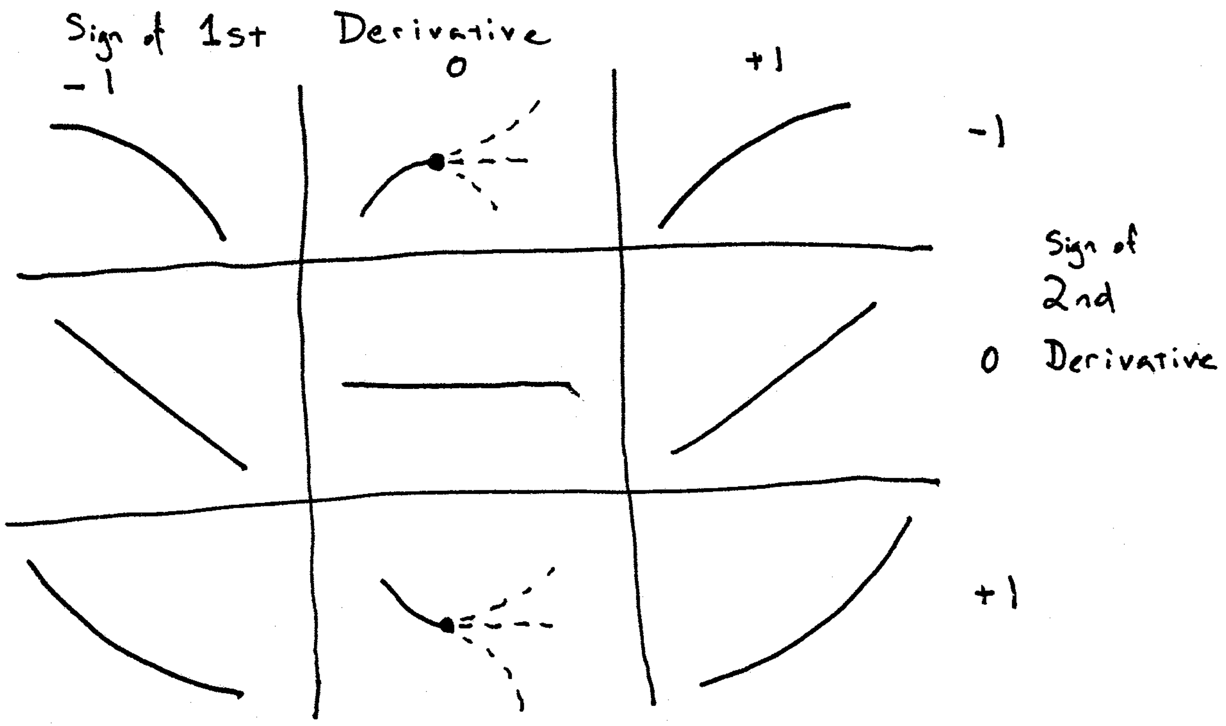

Figure 5: Primitive Curves of Qualitative Analysis

several primitive curves. Note that some qualitative curves correspond to more than one type of analytical curve. Note also the adequacy of this vocabulary for segmenting curves of arbitrary shape into regions corresponding to some primitive. Witkin and Tenenbaum [35] argue that segmentation based on such qualitative features is a universal feature of perceptual systems because such qualitative features have a strong likelihood of corresponding to salient features of the environment. Thus, the qualitative representation is weak, in that it has states of knowledge which correspond to many different states of a stronger analytic representation, but it is expressive in that those states are intuitively meaningful and support useful reasoning.

Let us consider how KF would handle the coffee cooling example. KF's vocabulary of relations is sufficient to permit us to borrow Fig.1 directly, but for purposes of illustration we will describe the coffee cooling situation in a qualitatively simpler way, as follows:

```
COFFEE_TEMP = ROOM_TEMP + TEMP_DIFF
TEMP_RATE   $M^{-}_{0}$ TEMP_DIFF
COFFEE_TEMP = $\int$TEMP_RATE
CONSTANT(ROOM_TEMP)
```

which translates into the following internal representation: (where unprimed variables denote the *level* aspect of the corresponding quantity)

```
COFFEE_TEMP  = ROOM_TEMP + TEMP_DIFF
COFFEE_TEMP' = ROOM_TEMP' + TEMP_DIFF'
TEMP_RATE    = -TEMP_DIFF
TEMP_RATE'   = -TEMP_DIFF'
COFFEE_TEMP' = TEMP_RATE
ROOM_TEMP'   = 0
```

This set of relations is the one that KF uses to make its predictions. Note that unlike LESTER's version of this problem, I have not specified in detail *how* the cooling rate depends on the temperature difference. I have merely said that it does.

Like LESTER, KF needs initial conditions to predict the future. The initial conditions needed are ***partial orderings*** relating those quantities that are measured in the same ***quantity space.*** A quantity space is less complicated than it sounds: it means a dimension of measurement. In the example, ***temperature*** is the salient quantity space. There are two temperatures: COFFEE_TEMP and ROOM_TEMP.[3] The initial condition is therefore

```
COFFEE_TEMP > ROOM_TEMP
```

In addition, some quantity spaces have ***distinguished values,*** i.e., significant constants at which something qualitatively different happens. For example, the temperature space for water includes 0 and 100 degrees celcius as distinguished values. Zero is often a distinguished value. Distinguished values maybe included in the partial order of their quantity space. For example, the assertion

[3]TEMP_DIFF is not a temperature, but a temperature difference. Since it is not referenced to the same scale, the relative magnitudes of TEMP_DIFF and the temperatures are not meaningful.

$T_{freezing} < T_{water} < T_{boiling}$

normally permits the inference that the water is in the liquid state. The use of quantity spaces thus permits salient qualitative inferences to be drawn without requiring absolute knowledge about the magnitudes of the quantities.

The initial conditions yield inferences about initial quit-values of quantity aspects, which are then propagated around the internal representation in direct analogy to LESTER. In this case, knowing that COFFEE_TEMP is greater than ROOM_TEMP allows KF to conclude that TEMP_DIFF is positive. This tells it that TEMP_RATE must be negative, and so must COFFEE_TEMP'. This implies that TEMP_DIFF' is also negative, and in turn TEMP_RATE' must be positive. The resulting assignment of values is as follows:

Quantity	Level	Derivative
COFFEE_TEMP	?	−
ROOM_TEMP	?	0
TEMP_DIFF	+	−
TEMP_RATE	−	+

We don't know the signs of the absolute temperatures, but since zero has not been assigned any special significance in the coffee temperature scale this has no import. We know everything else about how the quantities in the system are changing. At this point a set of rules for determining what happens next are applied.[4] There are certain configurations of interest to these rules. Quantities which have level and derivative of opposite sign, for example, are moving towards zero, and thus one can predict that they will intersect zero. Similarly, quantities with zero level and nonzero derivative will move away from zero. Quantities in the same quantity space may become equal to each other or to distinguished values, or they may cease to be equal. All these situations give rise to events. Here we have an analogy with LESTER's discrete or next-event

[4]All of the qualitative reasoning rules mentioned are described in admirable detail in the appendices of Kuipers' paper, and will be presented here only as needed.

level of simulation. Unlike LESTER, KF has nothing to say about *when* the next event will occur; it merely predicts, at best, what it will be. It cannot always even do that, and sometimes must resort to trichotomizing the simulation or even giving up.

In this case, we can predict that both TEMP_RATE and TEMP_DIFF are moving towards zero. In general, we would have to split the simulation here according to which of them gets there first, but in this case we can do better: we know that they are negations of each other at the quit level and hence have a common zero. Therefore they both reach zero at the same time and the next event is uniquely determined. This event also corresponds to the situation

```
        ROOM_TEMP = COFFEE_TEMP
```

In the new state, the zero values for TEMP_RATE and TEMP_DIFF are introduced and propagate to yield zero values for the derivatives of all the quantities. This is recognized by a rule which asserts that the system is in equilibrium, and terminates the simulation.

To summarize, KF predicted that the coffee would cool monotonically towards room temperature and that all processes would then cease.

The extension of the coffee cooling system to include a thermostatically controlled heating element is not possible in the systems of Kuipers and Forbus, because it requires a comparator, something which is not part of their systems. There is nothing to preclude adding one, however. In the qualitative arithmetic, we can get the effect of a comparator by introducing a device which outputs a positive constant when its input is positive or zero, a constant zero when its input is negative, and unknown when its input is unknown. Then, to compare two numbers, we feed their difference into such a device. We will call this relation NONNEGP .[5] The augmented system

[5]Note that NONNEGP is not fully invertible under the arithmetic described here. If the output of a NONNEGP is driven positive, there is no way of knowing whether the input should be + or 0; yet ? overstates our ignorance of the situation. One would need a seven-valued arithmetic that included NOT-NEGATIVE, NOT-ZERO and NOT-POSITIVE symbols to deal with this.

thus has the quantity relations

```
COFFEE_TEMP = ROOM_TEMP + TEMP_DIFF
THRESHOLD   = COFFEE_TEMP + TH_DIFF
HEAT_ON     = NONNEGP(TH_DIFF)
HEAT_RATE   M+0 HEAT_ON
COOL_RATE   M-0 TEMP_DIFF
TEMP_RATE   = HEAT_RATE + COOL_RATE
COFFEE_TEMP = ∫TEMP_RATE
CONSTANTS(ROOM_TEMP ,THRESHOLD)
```

and the initial values

```
ROOM_TEMP < THRESHOLD < COFFEE_TEMP
```

As with LESTER, the comparator is really providing a way to describe changes in the process structure within the confines of a single system description. An alternate solution is to simply acknowledge that the coffee-cooling with the heat off is a different situation from the coffee-cooling with the heat on, and reason about each separately. This has the bug or feature that reasoning about transitions between the two configurations is no longer the responsibility of the qualitative reasoner.

The first stage proceeds as before, but the next event is no longer equilibration of the system but the occurrence of

```
COFFEE_TEMP = THRESHOLD  ⇔ TH_DIFF = 0
```

At that point HEAT_ON suddenly becomes positive. This is propagated through the relations to yield a positive HEAT_RATE, and so TEMP_RATE is now the sum of a positive and negative quantity, which is indeterminate. This is because we don't know how strong a heater it is. It may be strong enough to cause the coffee to start heating, or perhaps to hold the temperature right where it is, or it may be so weak that it just slows the rate of cooling. The simulation must trichotomize on this missing information and simulate the three possibilities separately.

In the first case, we assume that TEMP_RATE is negative. The table of values is as follows:

(neglecting constants)

Quantity	Level	Derivative
COFFEE_TEMP	?	−
TEMP_DIFF	+	−
COOL_RATE	−	+
TH_DIFF	0	−
HEAT_ON	+	0
HEAT_RATE	+	0
TEMP_RATE	−	+

Here we have three quantities which are moving towards zero: TEMP_DIFF, COOL_RATE, and TEMP_RATE. It can be readily inferred from the causal relationships that the first two are not independent, but must reach zero together. Moreover, since COOL_RATE and TEMP_RATE are both negative, but they have a constant positive difference due to HEAT_RATE, TEMP_RATE must reach zero first. When it does, all other derivatives also become zero, and the system is in equilibrium at a value somewhere between ROOM_TEMP and THRESHOLD, with the heat on.

In the second case, we assume that TEMP_RATE is zero. The table of values is as follows:

Quantity	Level	Derivative
COFFEE_TEMP	?	0
TEMP_DIFF	+	0
COOL_RATE	−	0
TH_DIFF	0	0
HEAT_ON	+	0
HEAT_RATE	+	0
TEMP_RATE	0	0

Here the system is in equilibrium with the heat on at COFFEE_TEMP = THRESHOLD.

In the third case, we assume that TEMP_RATE is positive. The table of values is as follows:

Quantity	Level	Derivative
COFFEE_TEMP	?	+
TEMP_DIFF	+	+
COOL_RATE	−	−
TH_DIFF	0	−
HEAT_ON	+	0
HEAT_RATE	+	0
TEMP_RATE	+	−

Here, since TEMP_RATE is positive, TH_DIFF will again go negative, turning off the heater. The system will then be back in its original state. Oscillation is very easy to detect in qualitative reasoning: the entire state of the system is summed up in the values of the aspects. Since these are on a four-valued scale, and there only 14 aspects, most of which are redundant, the determination that one state is qualitatively identical to another is easy.

The complete qualitative description of the system's behavior then, is:

> Initially COFFEE_TEMP decreases until it reaches THRESHOLD. Then the future splits depending on the relationship between HEAT_RATE and the value of COOL_RATE at that point:
>
> If COOL_RATE(THRESHOLD) > HEAT_RATE, COFFEE_TEMP will continue to drop until it reaches a point above ROOM_TEMP where COOL_RATE = HEAT_RATE , where it will remain.
>
> If COOL_RATE(THRESHOLD) = HEAT_RATE, the system is in equilibrium.
>
> If COOL_RATE(THRESHOLD) < HEAT_RATE, COFFEE_TEMP will begin to rise. This will cause TH_DIFF to go negative, shutting off the heat. The system is then in the initial state. This oscillation will persist indefinitely.

This can be represented graphically as a *time-map,* as in Fig.6.

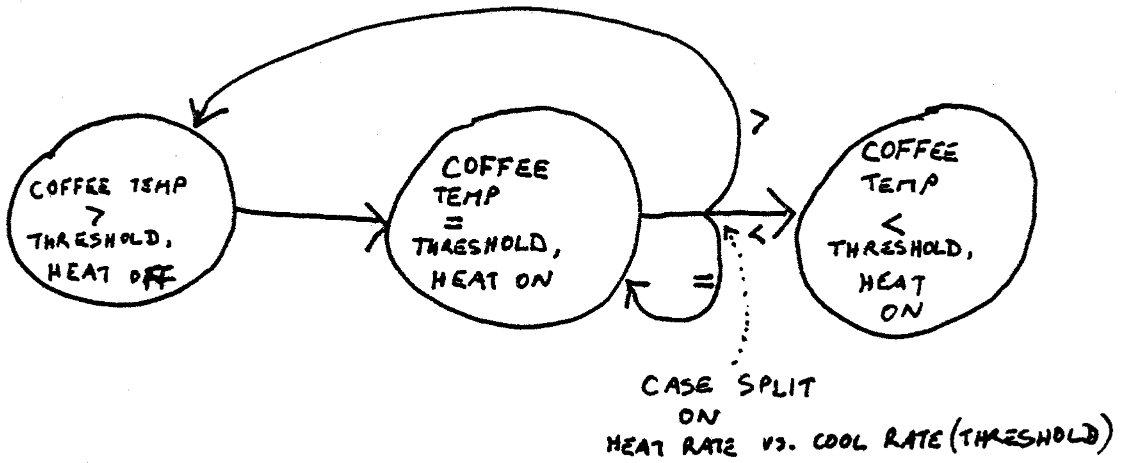
COFFEE TEMP > THRESHOLD, HEAT OFF
COFFEE TEMP = THRESHOLD, HEAT ON
COFFEE TEMP < THRESHOLD, HEAT ON
>
<
=
CASE SPLIT ON HEAT RATE VS. COOL RATE (THRESHOLD)

Figure 6: KF Output for Thermostatic Coffee Heater

This description is in many ways considerably richer than the one LESTER gave us. LESTER demanded that we provide the heating rate of the heating element, and with a suitably chosen value we got one of the above behaviors back. What we did not get is the significant information that ***the system will exhibit qualitatively different behaviors depending on the value of that parameter***. Moreover, KF gave us this result using a thermodynamics that was less specific than the one required by LESTER.

The qualitative system is surprisingly robust. It can handle second order systems like the oscillating spring using only the inferential machinery I have already described [15]. The number of rules required to completely describe it is not large, and none of the rules are very complex. The simplicity of the description of the system's state makes it possible to sometimes reverse the combinatoric tendencies of trichotomizing by fusing qualitatively equivalent states. Forbus offers examples of analyses by this method of rather complicated systems, such as pumped oscillators with friction. There are some limits to its representational capabilities, for example the idea of a delay is difficult to capture because of the lack of information about the duration of events. On the other hand, the quantity of inferences that can be made from such a simple description is surprising.

2.3 Conclusions

We have seen two different approaches to reasoning about the behavior of systems of mutually constraining quantities. One required quantitative information while the other used only qualitative information. I think it is pretty clear that the latter system derives considerable power from the weakness of its description, and that the lack of power of the former system is a consequence of the mandatory detail of its description. I think that an appropriate way to build a LESTER, if one wanted one, is by combining a knowledge base of computation techniques with a qualitative reasoner. In other words, the program should first predict qualitatively what the behavioral ***options*** for the system under consideration are, and then if it needs to and if it has the computational techniques it can attempt to augment the qualitative analysis with detailed predictions such as which possibility will actually occur and when. A program of this kind has been constructed by De Kleer [7] for the solution of simple mechanics problems.

A major unsolved question for reasoning systems of this type is, where are the causal models supposed to come from? All these models are descriptions of situations that carefully include certain features and ignore others in order to produce descriptions that are precisely tailored for the performance of a specific task. How is such a model extracted from a richer description? How does the extractor know which features are worth including in the causal model, and which can be neglected for the current purposes? This question definitely deserves further consideration.

3 Representational Dimensions of Time-Dependent Reasoning

This section reviews work by a number of authors, working in what are usually considered distinct areas of AI. The problems they address include commonsense physical reasoning, expert reasoning about designed artifacts, and story understanding. What they have in common is that they have confronted the problem of representing causal and temporal relationships. My purpose in juxtaposing their separate solutions is twofold. First, I wish to place the work I have just described in the context of a larger theory of Time Dependent Reasoning. Secondly, I want to provide a review of relevant literature for others who may be concerned with these issues. The discussion is largely concerned with issues of representational *adequacy*; that is to say, there are many important issues concerning the implementation of practical causal and temporal databases which I do not treat at all here. This should not obscure the fact that it is in precisely these areas that the most work remains to be done.

3.1 A Glance At Some AI Systems With Causal/Temporal Representations

I will briefly discuss the tasks to which the various representations to be discussed in this paper were put. One cluster of authors worked on engineered systems. This includes the Programmer's Apprentice group, [26], who worked on representations for programming plans and their use in understanding programs and predicting the effects of proposed modifications; Allen Brown [2], who wrote a program to understand the causal mechanism of a radio and demonstrate its understanding by troubleshooting broken ones; and de Kleer, who pioneered the qualitative reasoning techniques described in this paper, and investigated the tacit causal calculus that engineers use to reason about circuits.

Also working on mechanisms, but from the perspective of commonsense reasoning, were Reiger and Grinberg [25], who developed a representation for a household heating system that supported prediction and explanation. Others working in the domain of commonsense reasoning about the physical world, which may include both simple mechanisms and naturally occurring physical processes, are the qualitative reasoners Kuipers [15] and Forbus [11], as well as McDermott [20] and this paper. All of these were concerned with the issue of predicting the evolution of a system over time from an initial static description of its structure. A number of commonsense authors developed pure representation schemes. Although these schemes were not used for any tasks, they are intended to support equally the description, prediction and understanding of everyday physical reality. Hayes [12] worked on representing the behavior of liquids, and Charniak [4] worked on representing commonsense knowledge about painting.

Sacerdoti [27], Sussman [32], Charniak, and McDermott [18] are concerned with planning and problem solving, usually construed as being the development of a history map which achieves an intended result, starting with a goal, a world, and causal knowledge.

Schank and Abelson [31] were concerned with finding knowledge structures which could support the understanding of natural language stories. They noted that natural language descriptions of stories do not mention many significant facts or events in the stories because these can be readily inferred by human understanders. Computer understanders must therefore fill in (back-predict) the missing details using causal knowledge. Another issue concerned the implicit reliance of natural language descriptions on the understander's knowledge of the details of stereotypical situations and routines in the world, such as going to the movies or buying a newspaper. They suggest that such details are provided by *scripts*, i.e., history maps for stereotypical situations. They also provide the beginnings of a theory of naive psychology, i.e., how to understand and predict the behavior of people. This sort of theory presumably exists on top of the level of description described here.

These applications share a concern with two common subtasks: explanation (understanding) and prediction. Roughly speaking, prediction is concerned with using causal knowledge to generate time maps, while explanation is concerned with causally connecting events in time maps. In both cases the result is a history map, i.e., it has both causal and temporal structure. Charniak [4] and Wilensky [34] have argued that representations should be neutral with respect to planning and understanding, i.e., that the same knowledge base should support the planning of activities and the understanding of stories. This is equivalent to a demand that the representation be neutral with respect to explanation and prediction. They argue for this on the grounds of economy of storage. To this I would add the observation that a common underlying representation means that plans and causal knowledge can be transmitted between understanders in story form. This fact forms one of the central foundations of human civilization.

3.2 Time Maps

Every system that reasons about time requires a representation which is capable of capturing the temporal structure of a set of events. Following the terminology of McDermott [21], I call such representations *time maps.*

The simplest form of time map is perhaps a dateline, such as is used in history books. This maps the set of instants in the history of some world onto the set of points in the real line, in a one-to-one correspondence. This model works for history because history is a totally ordered sequence of events, to which numerical dates can be assigned which add and subtract like well behaved numbers.

The real line is too constraining a model to handle all the kinds of temporal situations that one might want to reason about. For example, supposing I want to represent a prototypical event sequence, such as going to a movie. Such representations are useful for understanding stories, and are called scripts by Schank & Abelson. [31]. I can't use real dates for the script events, since they don't happen at any one particular time. Nor can I use real numbers to indicate the relative temporal positions of the events, since these will vary from one actual occurrence of the script to the next. All that is really constant is the *ordering* among the events, and perhaps a range of values for the time between any successive pair.

For a more complex example, consider a case where I have decided to paint a ceiling and paint a ladder, but I have not decided which to do first. I know, however, that I must procure paint and brushes before I do either. This plan fragment has neither total ordering nor absolute nor even relative dating. This suggests that a partial order or directed acyclic graph may be a possible model of temporal structure. This was the model used by Sacerdoti's benchmark problem solver NOAH [28]. NOAH represented a plan as a series of layers of partial orders. The "vertical" dimension corresponded to elaboration in the detail of representation of plan steps and in the ordering among steps, and will be discussed later. Each "layer" was a time map, represented as a partial order among actions. The problem solver derived considerable power from its ability to postpone ordering decisions until criteria relevant to the choice of an ordering emerged from the elaboration of the plan. This ability in turn follows from the choice of a partial order for representing time maps.

Partial orders can occur in story contexts as well as in plans. A prototypical example is the Western story fragment "You two men go that way, the rest of you men come with me". In separating from each other in space, the two groups of men inaugurate separate histories. The events in these histories will not, in general, be completely ordered with respect to each other, although the events in the histories of each group may be totally ordered. In the example, it may be impossible to ascertain whether the two men had reached the mountain top before the rest entered the canyon, or simultaneously, or just after. Ordering of events in the two time lines can be reestablished later in the story in several ways. The two groups may regroup, or perhaps, one of the two men may call the rest on the telephone. This is a variation on an actual meeting where the histories remain spatially distinct, but causal connection among events allow some of them to be ordered exactly with respect to each other: specifically, the phone call events in each time chain overlap. The effect on the temporal structure is the same as would be produced by an actual meeting.

Another kind of temporal structure which cannot be captured by either the real line or a partial order is mutually exclusive possibilities. For example, Ronald Reagan may or may not run for reelection in 1984, and if he runs he may or may not win. A possible representation for this set of scenarios is a branching time map, with a single stem up to the point where Reagan announces his reelection plans, followed by one branch leading to "Reagan doesn't run", and another leading to "Reagan runs" which then splits into "Reagan wins" and "Reagan doesn't win". This issue has been addressed by McDermott [18] as well as a number of logicians. McDermott's solution is along the lines I have described: a branching tree of alternative futures.

Another group that has confronted the need to represent alternative possibilities is computer programmers. Their solution is the IF statement. The plan calculus, developed by Rich, Shrobe and Waters [26] to represent algorithms in a language independent manner, includes precisely these constructs in its primitive temporal vocabulary: a sequentiality or control-flow link, which partially orders plan steps, and an IF-link, with a test and two alternative outcomes. The computer's IF overconstrains the general situation, however, in that it demands that a test be specified. A real life counterexample is, "Either the Democratic candidate or the Republican candidate will win the election". One and only one branch will occur, but there is no test which can be performed to determine which other that waiting for the result of the election, ie, waiting to see which one occurs. In other words, people can reason nondeterministically about alternatives.

This type of branching must be clearly distinguished from the branching of the partial order. I will refer to partial order branching as AND-branching, and possible worlds branching as XOR-branching. In the former, both time chains are asserted to happen, while in the latter only one branch manifests. The partial order concerns ordering information within a single time map, while the possible worlds can be thought of as a way of having distinct time maps share common representational structure. By this interpretation XOR-branching plays a role not unlike an ISA or virtual copy link in a semantic net.

If the event at which the alternatives must manifest is known, then the time map can be represented by inserting some representation of an XOR-branch in the appropriate place. A more complicated situation is what Charniak call the "continuous IF", and what operating systems hackers know as *interrupts*: the dreaded asynchronous conditional. For example, say I plan to do my laundry, then take a shower, then cook dinner, but if the phone rings at any point I will stop what I am doing and answer it, then go back to what I was doing. How do we represent this as a time map?

There are two issues intertwined in this question. One is how to represent the continuity of time, or equivalently, how to reference the infinite number of instants in which the phone may ring. This will be addressed later in connection with continuous change; for now, suffice it to say that there is an axiom which asserts that there are instants between any non-identical instants. The other issue is, how to state that the phone may ring at any one of this infinite set of instants. This can be handled in a similar manner: an assertion which states that for all instants after the start of the plan and before its end, there is an XOR-branch which either continues the plan or is followed by a phone ring, followed by certain actions to be taken by me, followed by a return to the plan execution at the point where it was interrupted.

The key feature of this example is that the asynchronous conditional is represented *implicitly*.

It is represented by what may be called a universally quantified statement or backward chaining inference rule or consequent theorem, and not as a set of explicit propositional links between objects. It is often the case that a designer of representations feels free to choose what information will be represented explicitly and what implicitly according to programming convenience. There are, however, several theoretical criteria which can constrain this decision. For example, it may be desirable to have certain information always computed dynamically from certain data which varies in time, so that the result is always as up to date as the most current information. A different case is when the propositions to be represented are infinite in number. An explicit representation is always finite, because it must ultimately be encoded in some finite physical device. Implicit infinities are not uncommon, however: consider the integers, or, as in the example, the set of instants in which the phone may ring. The solution in these cases is to generate members of the infinite set only as needed. This can be thought of as a strong "computational convenience" issue. To see this, consider a finite but very large set: say, the set of temporal ordering (BEFORE) links between the set of all events that you know. This is $O(n^2)$ with the number of events. Here it is not impossible to generate them all, merely a profligate use of space.

In this example, the forced choice of an implicit representation for asynchronous conditionals provides additional constraints on the the temporal representation. In the example, we have an explicit representation of a plan as a partial order of sequentiality links, and an implicit representation of an XOR-branch at each of these links. The view from "the knowledge level", to borrow Newell's [23] term for the union of the implicit and explicit representations, is that there is an infinite number of instants, each with an XOR-branch following. The interpretation of a sequentiality link between two events therefore cannot always be simply that *B followed/will follow A*. Wherever asynchrony is involved, the interpretation must be rather that

B might have followed/might follow A. Typically this occurs in planning and prediction, i.e., future oriented situations where information is very incomplete. However, the same points apply to mysteries in the past.

Planning in the face of asynchronous conditionality involves the generation of a new type of goal, and its negation, called by Schank & Abelson *p-goals* and *a-goals*, for *preservation* and *avoidance* of some state. This issue has also been treated by McDermott.

Still another type of temporal structure is recurrence or repetition. For example, Charniak [4] describes the plan for painting a wall in terms of repeated application of the plan "dip brush in paint, apply to wall". The salient feature of this description is that it is concise and non-redundant, yet capable of being interpreted by a suitable executive to produce an arbitrarily long sequence of behavior. Recurrence can be transmitted in stories in a similar way, with one or a couple iterations related explicitly, and the rest alluded to by some convention, like "and they wandered like this for the next forty years". Alternatively, they can be explicitly spelled out, as happens sometimes in juvenile jokes. In relation to recurrence, then, one needs two specialized capabilities: detection of recurrence, which involves going from a perceived repetition to a concise non-redundant representation, and execution of recurrence, which is the reverse. The former is simply a particular case of recognizing similarity between current and past event sequences.

Recurrence can be represented simply as cycles in the graph of sequentiality links, in which case the routines which traverse the graph need to have cycle-smarts; or alternatively, the acyclic constraint on the sequntiality graph may be maintained, and a distinguished link-type can be used to encode the looping back links. This was the approach taken in the plan calculus. The composition of loops with the implicit representation of asynchronous conditionality is straightforward, permitting asynchronously terminated loops, eg., "Keep pedalling til the bell rings". For representation of repetition in stories, there are two distinct situations: one where we have something different to say about particular repetitions, and one where we do not. In the latter case, the simple cycle is adequate: the XOR-branch that denotes the loop exit can be annotated with the details of how much repetition took place. One can imagine two ways of handling the former case, the case where we want to distinguish particular repetitions. One is to *extrude the cycle,* i.e., unravel it into as many linear copies as we want to distinguish. (This is what happens in execution of a looping plan, when a cyclic plan is effectively extruded into a linear history.) What is lost in this is the explicit representation of the relationship between the linearized sequences, namely, their derivation from a common schema. It can also get messy if one wants to represent a sequence of repeated generic repetitions interspersed with distinguished ones: one might need a different cycle for every unbroken interval of generic repetitions to capture ordering relations among the distinguished ones, even though all such cycles would be identical. An alternative way makes use of hierarchical organization principles. If the repetition schema is represented as a one node loop at one level, and that node is expanded to an acyclic time map at a lower level, then particular repetitions can be represented as instantiations of the acyclic map, viewed as a script or schema. This leaves completely free the specification of ordering relations among the instantiations. A related structuring is where the nodes in the loop are instantiated and the instantiations are not ordered amongst themselves, so that one knows where in the loop a particular episode occurs but not in which repetition. That both of these last two representations occur in people is suggested by a certain common confusion in memory of songs that the reader can readily validate internally. It is common for people to remember the complete tune of a song, but only some of the lyrics, and in particular to mix up lines from different verses, and/or to get the verses in the wrong order. Any such confusion can be readily captured by the representation outlined here. The loop is the musical structure, and the lyrics are bound to lower level instantiations of that structure. A correctly recalled verse misplaced in the song corresponds to a complete instantiation of the acyclic part of the loop unordered with respect to there such instantiations. Mixing up of lines corresponds to loss of ordering relationships within the acyclic portions of the different repititions. (See Fig.7). This suggests that it would be far rarer for someone to misplace lyrics within the melody, eg., to place an opening line of a verse at the end of the verse.

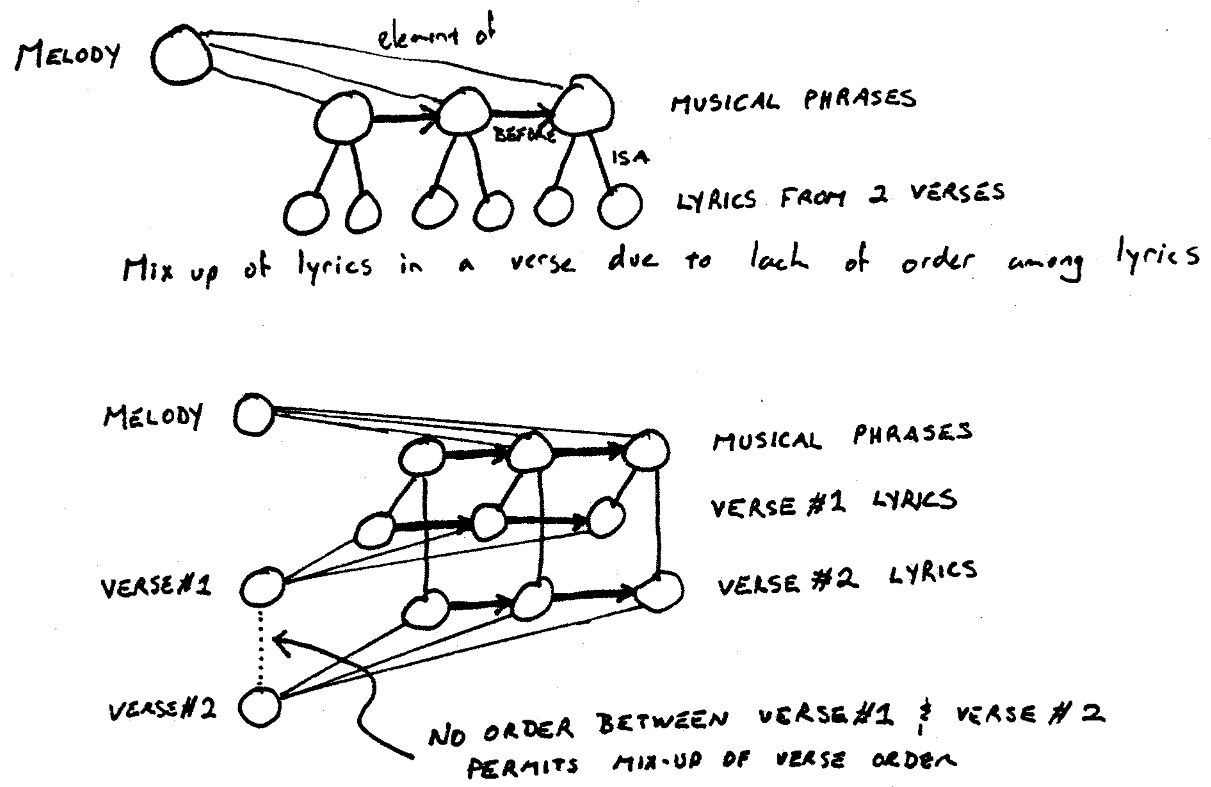
MELODY
element of
MUSICAL PHRASES
BEFORE
ISA
LYRICS FROM 2 VERSES
Mix up of lyrics in a verse due to lack of order among lyrics
MELODY
MUSICAL PHRASES
VERSE #1 LYRICS
VERSE#1
VERSE #2 LYRICS
VERSE#2
NO ORDER BETWEEN VERSE#1 & VERSE #2
PERMITS MIX-UP OF VERSE ORDER

Figure 7: Diagrams Illustrating Loss of Ordering Relations In Memory For Songs

A simple cycle will have an entry node with two predecessors and an exit node with two successors, one a within-loop node and one not in each case. The exit node is an XOR-branch, an IF, if you will, whose test is the termination test for the recurrence. The entry node is a kind of join that we have not seen yet. Corresponding to AND-branching there is AND-joining, as when the two groups of cowboys rendezvous to cut 'em off at the pass, and thereby reestablish ordering relations between their histories. Similarly, corresponding to XOR-branching there is XOR-joining, which represents irrelevance of prior events. When two linear histories XOR-join to form one, the implication is that, whichever one of them actually happened or will happen prior to the join, the same history follows after the join in any case. For example, Reagan may win the election or Reagan may lose the election, but the economy will nosedive afterwards no matter what. This was described in the previous chapter as case splitting or trichotomizing (XOR-branching) and case joining. In the case of the cycle, the steps following the initial entry into the loop and succeeding entries will be qualitatively identical. An example of this occurs in KF's solution to the extended coffee cooling example,in the branch describing the oscillation.

Up to now I have discussed only the structure of time maps. Now I will consider what it is that is being structured. What are these nodes and arcs? The earliest problem solvers [9] [10] utilized a distinction between states and operators. A state is an object of which facts hold true, while an operator is a function that maps states satisfying its preconditions onto states satisfying its postconditions. This description is so general that it applies to descriptions of problem solving activity in many kinds of problem spaces, not just temporal ones. The situation calculus proposed by McCarthy [16] was a straightforward casting of this general picture into a specifically temporal context. The states were *situations*, and the standard predicate calculus notation was augmented so that temporally scoped predicates could take a situation argument. Situations were totally ordered and had no duration, a restriction which precluded treatment of continuous change, asynchrony, simultaneity, and alternatives. A similar temporal representation is implicit in Schank's conceptual dependency representation, where actions play the role of operators, changing the values of certain key functions such as the location or possession of objects across situations. The problem solvers NOAH and HACKER also operated within these restrictions, although NOAH relaxed the ordering requirement, as described elsewhere. The plan calculus also operates within the constraints of discrete, durationless instants, a view that is compatible with the actions of a single serial computer operating in an interrupt-free environment.

Hendrix [14] extended this description to handle continuous change. In his formalism, processes had beginnings and endings as well as points of interesting change, such as a bucket overflowing, and processes could overlap in time. A consequence of this extension is that *situations* differentiated into two kinds of entities: *instants*, corresponding to points of change of facts in the world, and intervals during which facts held true. Instants must be totally ordered, but intervals can participate in a number of types of interactions, such as overlap, containment, abutment, disjunction, and identity. Allen [1] has developed a temporal representation which drops all explicit mention of instants, and describes time purely in terms of intervals and relations between them. The increased parsimony in terms of the fewer types of objects (from two to one) is offset by the increase in the number of of explicit relations between them (from two to nine). Neither scheme has any more expressive power than the other, and whether either is better by some other criterion remains an open question. Dean [6] designed a system for maintaining the consistency of a set of facts that were quantified over time. The system was designed to support the moving around of event sequences in time, so that a planner could schedule its actions and detect possible interactions in its plans. The system used the instant-plus-interval representation, which permitted a simple and concise expression of the algorithms for maintaining temporal consistency of facts. By contrast, an Allen-style representation would have to be considerably more complex due to the variety of transitions among relations between intervals that occur as they are displaced in time with respect to each other.

McDermott's temporal calculus [18] includes states, events, and instants. States and events both behave like intervals, but they participate in slightly different kinds of causal interactions, as we will see later.

Hayes [13] appears to be the big maverick in time map theory. He proposes a representation for describing the physics of liquids in which the objects are 4-dimensional volumes of spaces time. In this view, pouring water from a pitcher onto a table would be represented as a *containment* which overlaps temporally and abuts spatially with a *falling,* which overlaps temporally and abuts spatially with a *splat.* A related representation was employed by Miller [22] in a scheduling program which scheduled the actions of a problem solver subject to realistic location and transportation constraints. Embedded within both these descriptions, however, is the classic time map of instants and intervals. The additional focus on space and spatiotemporal interactions augments but does not contradict what we have discussed so far.[6]

To summarize, time maps are used both to represent things which did occur (stories), things which might occur (alternatives, plans), and things which frequently occur (scripts, plans, programs). Ideally a true map of real time is linear and totally ordered, however, departures from this structure in the form of loops, *and*-branching and *xor*-branching permit the compact representation of repetition, incomplete ordering information, and alternative possibilities, respectively.

3.3 Change

The situational calculus described a world which changed in discrete, instantaneous hops. It is not clear whether our world ever behaves in this way; electrons might but you can't catch them at it. Other phenomena that we tend to consider instantaneous and discrete, such as the turning on of a light, can be redescribed as processes with considerable internal detail. The light switch

[6]One of the psychologically important spatiotemporal constraints is called *object permanence,* which may be expressed as the constraint that objects' paths through space-time must be continuous, or equivalently that you can't get from here to there without going through points in between, or again that things don't appear and disappear. The validity of this law in the physical universe, with its tunnelling electrons and black holes, is dubious, but its importance as an organizing principle for our understanding of reality is significant.

continuously changes position over some range, at some point bringing two pieces of metal into contact. Current flow in the circuit causes the bulb filament to heat up, and eventually glow; the photons released take time to traverse the room to the observer's eyes. Within this description, there are a number of discrete events: the coming into contact of the wires, the associated onset of current flow, and the onset of luminescence in the filament. We can play the same game with each of these. As the two pieces of metal come closer and closer together, is there an instant such that before that instant they are not touching and after it they are? Possibly, but we can also imagine that at a sufficient level of spatial and temporal magnification we would observe a more or less continuous increase in electrical coupling, as the atoms in the respective surfaces influence each others' electric fields in increasing numbers and strength. The point is that it is frequently useful to consider a change as instantaneous in one context; yet we must retain the ability to change our minds an consider it an arbitrarily complicated sequence of states in another context.

Discrete change is important because it permits us to divide time up into chunks (states). The situational calculus represented all changes as discrete, but this was excessively limiting. The trick is to represent discrete change in such a way that we retain the power to talk about continuous change when we want to. Hendrix [14] noted that although something that is continuously changing takes on a different value every instant, there is nothing interesting to say about most of those values. In fact, the only values about which one has anything interesting to say are a finite number where something qualitatively different happens in the world as a result of the continuous change passing through them. For example, if I am filling a bucket with water, I can safely describe the filling process with a single static fact which asserts that the amount of water in the bucket is continuously increasing from one instant to the next, and the next state needing a new description begins when the water level passes the rim of the bucket.

Most representations capable of handling continuous change rely on this fact. Rieger and Grinberg [25] provide a causal link in their commonsense reasoning representation which they call a *threshold* link. It links a node representing continuous change in some quantity to a node representing the passage of the quantity's value through a threshold value in a particular direction. In their system this latter node is called a *state*. When this actually occurs during a simulation, the state is asserted, and any consequences linked to it as well. McDermott [20] provides a similar construct which he calls a *comparator*[7]. A comparator is a device with a binary output which changes states when its inputs – arbitrary mathematical curve descriptions – cross.

Qualitative reasoning, due to Kuipers and Forbus, develops discrete changes from trichotomy. Measures of the quantities being reasoned about are not used, but there is information relating them to each other. A pair of quantities A and B in the same *quantity space* are related by one of $A < B$, $A = B$, or $A > B$. Sometimes it can be inferred that a relationship changes from one of these states to the next. A particularly interesting case is when, say, A is constant and B is initially less than A but continually increasing. Three states are produced in succession: $A < B$, $A = B$ and $A > B$. The middle state is notable in that it has no duration; it is instantaneous, in contrast to the other two which may have arbitrary non-zero duration.

The method of describing continuous change in terms of the signs of the first and second derivatives can be thought of as defining a vocabulary of primitive curve forms (See Fig.5). To describe a particular curve, one need only break it up into intervals which can be described by one of these forms.

The fact that some quantity is continuously changing is a static fact; it may be true or false in

[7]...described in detail elsewhere in this paper.

some interval. The infinite number of values taken on by that changing quantity are individually of no interest unless they provoke a qualitative change somewhere. However, since any one of them *could* provoke such a change we need to be able to talk about any of them.

The distinctive properties of continua can be specified using the *Axiom of Density*:

$$\forall\ x,y \in C \quad \exists\ z \in C \quad \text{s.t.} \quad \text{BETWEEN}(x\ z\ y)$$

where C is a continuum, and x,y and z are points in it. In other words, between any two points there is another point. McDermott [18] uses this to capture the continuity of time. What the axiom does is *imply* the existence of an uncountably infinite number of points in a continuum without actually mentioning them. One can similarly define the meaning of monotonic increase and decrease of a quantity.

Two other means of representing continuous change should be mentioned. One is representing a continuous function as an array of values, ie, doing a discrete approximation with uniform ticks of the simulator clock. The other is representing change using mathematical functions, as was done in LESTER. This has the nice property that one can determine the value of the function at any instant in time. A criticism of both these methods is that they impose a high level of mandatory detail in the descriptions they operate on before they can produce any results. As Kuipers frequently puts it, they lack interesting "states of partial knowledge".

3.4 Causality

In this section, we ask, what is causality, and what is it good for? *Causality* denotes another kind of relationship between states, distinct from but intimately related to temporality. Intuitively, causality is to *why* as temporality is to *when*. They constrain each other very weakly by virtue of the following *Axiom of Causality*:

$$\forall\ a,b \in SS,\quad \text{CAUSES}(a,b) \supset\ \sim\text{BEFORE}(b,a)$$

where SS is the set of all states; i.e, effects never precede their causes. What does it mean to say

A causes B? Basically, it means that we have the ability to perform various kinds of useful inferences. We can predict that if we see an A then we will see a B shortly. Having seen a B we can conclude that there must have been an A. Also, we can *explain* an observed occurrence of A following B by invoking the causal rule, eliminating any need to infer other states to explain the occurrence of A. None of these conclusions is logically sound as given, but they illustrate the point.

Causality is frequently confused with two similar concepts, logical implication, and correlation. Causal relations are a proper subclass of implications, in that there are implications which are non-causal, eg., if A=B & B=C then A=C. Correlation is distinguished from causality by its symmetry: if A correlates with B then B correlates with A. Causality is asymmetric.

Since causality is a means of connecting states, representational structures analogous to time maps can be built up by connecting states with causal links: let's call them *causal maps*. In general, the same states will be connected both by a time map and a causal map; and what we really have is a hybrid structure containing states connected by both causal and temporal links: a *history map*, if you will. Schank [30] has shown that the causal connectivity of such maps partially accounts for data on which features of a story people will remember long after they heard it. Much of the causal and temporal information in a given history map is redundant in the presence of the causal rules that permit the formation of the causal component of the map; hence it is likely that human memory never explicitly stores a complete history map, rather it stores select fragments which can support approximate inferential reconstruction of the original. In particular, since the Axiom of Causality allows causal links to partially order events in time, explicit temporal links will often be redundant.

Causality occurs in one form or another in all the representational schemes considered in this paper. In the situational calculus, actions are represented as bundles of *pre-* and *post-conditions*.

Actions *cause* their post-conditions, and are caused by the problem solver's decision to do them in the presence of their preconditions. In Rich's application of the calculus to programming plans, causality is identifiable with *data-flow.*

Many types of causality have been distinguished by different authors in different representational systems. Hendrix distinguishes *fact causation,* which is the initiation of a process whenever a conjunction of facts becomes true, and actor causation, which is when an actor decides to do something. The actor's ability to do something is gated by a set of *enabling* facts. Schank & Abelson [31] distinguish *cause, enablement, disenablement, and initiation* of a mental state in an actor by a fact, and *reason links* from an actor's mental state to an action. Reiger & Grinberg espouse *one-shot causality,* where the one time occurrence of cause is sufficient to bring about the effect, which persists until something makes it go away (eg., light switch); *continuous causality,* where the effect persists only as long as the cause; and, in addition to providing for *gating* of both kinds of causality by facts, there is an *enablement* link which does the same thing. Charniak's system has an ordinary causal link, plus a link which he called a *leads-to* or *reason* link. Reason links connect actions in a plan to the state which the action is supposed to bring about or prevent; hence they indicate the *reason* for doing that step. McDermott nominates *event causation,* which is the causing of one event by another, allowing for a possible delay; and *persistence causation,* which is the causing of a persistence by an event. A *persistence* is, roughly, an interval during which a fact remains true.

Some have gotten by with just a *cause* link, such as Allen Brown [2] and de Kleer [8], but the fact that they were considering electronic circuits where cause and effect are (approximated as) instantaneously coupled makes this suspect. Hayes eschews any mention of causality. altogether, preferring logical implication. It is not clear how he would state the Axiom of Causality.

Several of these distinctions are not strictly necessary. Hendrix's actor causation and Schank's mental state links are both specialized kinds of causality for talking about sentient beings, in other words, their arguments are more constrained than a generic causal link, but otherwise the meaning is the same. A distinction between causality and enablement has been drawn by several authors. The argument runs like this: if the door is open I can leave the room, but there is no sense in which the door being open *causes* me to leave the room, hence cause and enablement are distinct. To my mind this could be better expressed using necessary and sufficient conditions: the necessary conditions for my leaving the room are my ability to move my body, my will to move, and the existence of an unobstructed path, all of which must persist until my action of leaving the room is complete. This way hairy issues like whether flipping a light switch *causes* or *enables* the light to come on never come up.

There is a rough analogy between Reiger and Grinberg's two kinds and of causality and McDermott's. R&G have, essentially, threshold crosses which cause facts and facts which cause other facts. McDermott's formulation is complicated by an explicit provision for delays; however we can factor this out by introducing an intermediate causal agent that propagates the delayed causality. For example, if A causes B after a delay X, we can pretend that A set in motion a process P which evolved until an instant X time-units after A, when it crossed a threshold, causing B in the process. Ignoring delays, then, McDermott's causal types are, the simultaneous causation of one event by another, and the causation of a new, persistent fact by an event. This distinction is also made in qualitative reasoning, where the simultaneous causation is associated with proportionality links between quantities, and the threshold causation is represented as changes in the partial order of the quantities in a quantity space.

Another kind of causal representation is exemplified by Kuipers, Forbus and McDermott [20], where causal relations are expressed as mathematical relations. An expression of the form

```
A = B + C
```

is really a shorthand for a set of implicational rules, such as

```
When A is constant: An increase in B implies a decrease in C
                    A  decrease in B implies an increase in C
                    vice versa
When B is constant: An increase in A implies a decrease in C
etc.
```

Typically, we do not intend all of these implications to be causal: for example, if the temperature in my house is constant and the temperature difference between inside and outside increases, this does not *cause* the outside temperature to drop; rather, it implies that the outside temperature must be dropping. Thus, arithmetic descriptions express implications, some of which are causal, some reverse-causal, and some definitional. The reasoning systems described in the previous section are oblivious to this distinction, i.e., they will propagate influences in a direction if they can, regardless of the semantics of the link. Forbus has made some attempts to address this.

A different sort of causal relation has been described by Charniak, which he calls a *reason* link. Reason-links are teleological: they express purposes, which are related to but distinct from causes. If B is a reason for doing A, then doing A will either help cause or prevent B, and the causal relationship between A and B will be redundantly expressed by *cause* links. Reason links capture the interaction between the planner's goals and the world's causality. Typically an action may cause a number of effects, and in any use of that action in a plan, some of these will be the desired effects, and some will be side-effects, perhaps unwanted and necessitating clean-up or prevention tasks in connection with them. For example, in painting a wall, the desired effect is that paint gets on the wall, while an undesired effect is splatter, whose negative consequences must be prevented by a dropcloth. This would be represented by a reason-link from painting to color-change, cause-links from painting to both color-change and splatter, and a prevention-reason-link from place-dropcloth to splatter. Although the reason links are strictly redundant to the cause links, the latter define a combinatorically large space of possibilities in which plans implicitly exist. Known plans therefore embody real computational work, and must be saved if the planner is to improve the speed with which it can reason about purposive behavior.

The main points of this section are that there are two kinds of causality, one involving statechanges and one which is instantaneous, that they are not the same as implication, correlation, temporal order or mathematical relations, nor are they the same as purpose, which is a useful and complementary relation.

3.5 Abstraction

The idea of hierarchical description is pervasive in the causal literature, and although it is not part of the ontology of causality and time, in must inevitably be part of any attempt to represent them, especially if the representation is to be large and is to be used. The people working on representing causality in mechanisms tend to pay the most attention to this issue. Brown includes in his representation for a radio circuit both the schematic and the various layers of block diagrams. He notes that the representation of the same design at different levels of detail makes possible a kind of binary search through the device when trying to localize a bug. The procedural nets used by NOAH to represent plans consisted of layers of time maps, with "lower" layers being more detailed refinements of "higher" layers. Reiger and Grinberg represent multiple causalities of the same event, each at a different level of resolution. Rich uses *overlay* links to connect plan-steps to plans capable of implementing them. Charniak's system for representing painting plans uses an analogous construct, which he calls *method* links. They are just like the standard goal-to-plan links, except that the "plan" can be a physical phenomenon, as in evaporation as a method for drying paint. If temporal links tell *when* something happened, and causal links tell *why*, method or abstraction links tell *how*.

Abstraction links separate history maps into layers, each with internal causal and temporal links, and each layer being a specialization of the one "above" it. There are a number of distinct ways to specialize history maps. A time map can be specialized by

- adding events to it adding ordering constraints between existing events replacing single events with complex detailed descriptions

The last type of specialization can take two forms, called by McDermott *refinement* and *revision* [17]. Refinement adds detail without denying any assertions that pertained to higher levels, while revision replaces a crude high level description with a more detailed to which some high level assertions may no longer apply. For example, in the section on change, I indicated that the temporal representation ought to be able to consider a phenomenon an instantaneous statechange, and yet be able to rescind that description in favor of a more detailed one which described it as an arbitrarily complex set of states. Revision and refinement are also applicable to causal relations.

4 Conclusion

In the previous sections, I have considered a number of issues in the representation of time dependent systems, and a number of representational systems that have confronted these issues. The main issues concerned representing time, causality, purpose, and abstraction, or *when, why, what for* and *how.* I suspect that the alignment of these issues with the basic question words of English is not coincidental. Rather, it seems plausible that the question words indicate the fundamental dimensions of organization of human knowledge. This organization follows from the constraints imposed by the tasks that that knowledge serves: the explanation and prediction of observed, narrated and hypothesized events.

References

[1] Allen, James.
A general model of action and time.
Technical Report TR97, U. of Rochester Department of Computer Science, 1981.

[2] Allen Brown.
Qualitative Knowledge, Causal Reasoning, and the Localisation of Failure.
Thesis 362, MIT, March, 1977.

[3] Brown, J.S., Burton, R.R. and deKleer, J.
Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II and III.
Academic Press, 1981, .

[4] Charniak, Eugene.
A framed PAINTING: the representation of a common sense knowledge fragment.
Working Paper 28, Fondazione Dalle Molle, 1976.

[5] Davis, Ernest.
Organizing Spatial Knowledge.
Technical Report 193, Yale University Computer Science Department, 1981.

[6] Dean, Thomas.
Time Map Maintenance.
Technical Report 289, Yale University Computer Science Department, 1983.

[7] de Kleer, Johan.
Qualitative and Quantitative Knowledge In Classical Mechanics.
Thesis 352, MIT, December, 1975.

[8] DeKleer,J.
Causal and Teleological Recognition in Circuit Recognition.
Thesis 529, MIT AI, 1979.

[9] Ernst, George W. and Newell, Allen.
GPS: A Case Study in Generality and Problem Solving.
Academic Press, 1969.

[10] Fikes, Richard and Nilsson, Nils J.
STRIPS: A new approach to the application of theorem proving to problem solving.
Artificial Intelligence 2:189-208, 1971.

[11] Forbus,K.
Qualitative Process Theory.
AI Lab Memo 664, MIT AI, 1982.

[12] Hayes, Patrick.
Ontology for Liquids.
1979.

[13] Hayes, Patrick.
The Naive Physics Manifesto.
1979.

[14] Hendrix, Gary.
Modeling simultaneous actions and continuous processes.
Artificial Intelligence 4:145-180, 1973.

[15] Kuipers, Benjamin.
Commonsense Reasoning About Causality: Deriving Behavior From Structure.
Working Papers in Cognitive Science 18, Tufts University, May, 1982.

[16] John McCarthy.
Programs With Common Sense.
MIT Press, 1968, pages 403-418chapter 7.

[17] McDermott, Drew V.
Flexibility and efficiency in a computer program for designing circuits.
Technical Report 402, MIT AI Laboratory, 1977.

[18] McDermott, Drew V.
A temporal logic for reasoning about processes and plans.
Cognitive Science 6:101-155, 1982.

[19] McDermott, D.V.
Data-dependencies On Inequalities.
In *Proceedings of AAAI-83,* pages 266-269. American Association for Artificial Intelligence, 1983.

[20] McDermott, Drew V.
Reasoning About Time-Dependent Systems: A Research Prospectus.

[21] McDermott, Drew V.
Personal communication.

[22] Miller, David.
Scheduling Heuristics for Problem Solvers.
Technical Report 264, Yale University Computer Science Department, 1983.

[23] Newell, Allen.
The Knowledge Level.
The AI Magazine 2(2), Summer, 1981.

[24] Pople, Harry.
Personal communication.

[25] Reiger,C. & Grinberg,M.
The Causal Representation and Simulation of Physical Mechanisms.
Computer Science Technical Report 495, University of Maryland, 1976.

[26] Rich, C.
A Formal Representation for Plans in the Programmer's Apprentice.
In *Proceedings of IJCAI-81.* International Joint Conference on Artificial Intelligence, Vancouver, B.C., 1981.

[27] Sacerdoti, E.D.
A structure for plans and behavior.
Technical Report 109, SRI Artificial Intelligence Center, 1975.

[28] Sacerdoti, Earl.
A Structure for Plans and Behavior.
American Elsevier Publishing Company, Inc., 1977.

[29] Sakallah, Karem A.
Mixed Simulation of Electronic Integrated Circuits.
Thesis DRC-02-07-81, DRC-CMU, November, 1981.

[30] Schank, R.C.
The structure of episodes in memory.
In D. Bobrow and A. Collins (editors), *Representation and Understanding,* pages 237-272. Academic Press, New York, 1975.

[31] Schank, R.C. and Abelson, R.
Scripts, Plans, Goals and Understanding.
Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[32] Sussman, G.J.
Artificial Intelligence Series. Volume 1: *A computer model of skill acquisition.*
American Elsevier, New York, 1975.

[33] Sussman, Gerald J. and Steele, Guy L.
CONSTRAINTS-- A language for expressing almost-hierarchical descriptions.
Artificial Intelligence 14(1), 1980.

[34] Wilensky, Robert.
Metaplanning.
Memo UCB/ERL M80/33, Berkeley Department of Computer Science, 1980.

[35] Witkin, A.P. & Tenenbaum, J.M.
What is Perceptual Organization For?
In *Proceedings of the 8th IJCAI,* pages 1023-1026. International Joint Conference On Artificial Intelligence, 1983.