Doing Time: Putting Qualitative Reasoning on Firmer Ground

Brian C. Williams
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139
(williams%mit-oz@mit-mc.arpa)

Abstract

Recent work in qualitative reasoning has focused on predicting the dynamic behavior of continuous physical systems. Significant headway has been made in identifying the principles necessary to predict this class of behavior. However, the predictive inference engines based on these principles are limited in their ability to reason about time.

This paper presents a general approach to behavioral prediction which overcomes many of these limitations. Generality results from a clean separation between principles relating to time, continuity, and qualitative representations. The resulting inference mechanism, based on propagation of constraints, is applicable to a wide class of physical systems exhibiting discrete or continuous behavior, and can be used with a variety of representations (e.g., digital, quantitative, qualitative or symbolic abstractions). In addition, it provides a framework in which to explore a broad range of tasks including prediction, explanation, diagnosis, and design.

1 Introduction

The physical world around us is continually changing. Thus in order for an agent to make intelligent decisions about his interaction with the surrounding environment he needs to be able to predict the effects of his actions and of changes he observes. Recent work in qualitative reasoning[9,10,6,5,3] has focused on predicting the dynamic behavior of continuous physical systems (e.g., predicting fluid flow, pressure stability or mechanical oscillations). Significant headway has been made in identifying the principles necessary to predict this class of behavior. However, the predictive inference mechanisms based on these principles exhibit a number of severe limitations, such as 1) forcing one to make unnecessary temporal distinctions, 2) overrestricting the language used for describing temporal behavior, 3) performing weak temporal inference, 4) constructing incomplete justifications, and 5) making irrelevant domain restrictions.

This paper presents a general approach to behavior prediction which overcomes many of these limitation. Generality results from a clean separation among principles relating to time, continuity, and qualitative representations. The resulting inference mechanism, based on propagation of constraints[8], is applicable to a wide class of physical systems exhibiting discrete or continuous behavior, and can be used with a variety of representations (e.g., digital, quantitative, qualitative or symbolic abstractions). In addition, it provides a framework in which to explore such tasks as prediction, explanation, diagnosis[4], and design.

I begin with a summary of current qualitative reasoning systems and their limitations. Based on these limitations, a more robust language for describing behavior over time is presented. Next, an inference mechanism, referred to as a *Temporal Constraint Propagator (TCP),* is introduced; TCP predicts the behavior of the desired class of systems in terms of the behavioral language. Finally, the power of this approach is demonstrated with an example taken from qualitative reasoning.

2 Qualitative Reasoning In A Nutshell

Given a description of a physical system and its initial conditions, qualitative analysis typically involves 1) describing the temporal behavior of the system's state variables, in terms of a particular qualitative representation and 2) explaining how this behavior came about.

The description of a physical system consists of a set of state variables (e.g., force and acceleration) and a system of equations, parameterized by time, which describe the interactions between these variables (e.g., $f(t) = ma(t)$). A *qualitative representation* divides the range of values a quantity can take into a set of regions of interest (e.g., positive, negative and zero). The particular representation selected depends on properties of the domain and the goals of the analysis. The *qualitative value* of a quantity is then the region it is in.

The behavior of the system can be viewed in terms of a *qualitative state diagram,* where each state describes the qualitative value of every state variable in the system.[1] The behavior of the system over time can be viewed as a particular path through this state diagram. Each state along this path represents an interval of time over which the system's state variables maintain their values. The duration of this interval is dictated by principles involving continuity and rates of change.[9]

We say that the system changes state whenever *any* state variable changes its qualitative value. The values in the next state are then determined by 1) identifying those quantities which cannot change value (e.g., if Q is positive in a particular state and its derivative is positive or zero then it will remain positive in the next state), and 2) propagating the effects of those quantities that are known to change. The qualitative reasoning system also keeps track of the reason for every deduction in 1) or 2), using the record, among other things, to generate explanations (e.g., "an increase in force causes the mass to accelerate").

This work was supported in part by an Analog Devices Fellowship, and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505.

[1] The process of constructing a qualitative state diagram is called Envisionment[3].

3 Limitations of the Existing Approach

The above approach has been adequate for describing the behavior of a number of simple systems such as pressure regulators, harmonic oscillators, RC networks and rudimentary plumbing. However, the approach has a number of limitations which make it difficult to use for analyzing large systems and describing complex behaviors. A few of these limitations will illustrate the crucial difficulties.

One limitation arises from the fact that a state-based approach imposes a total ordering on events. That is in order to describe a system's behavior as a sequence of states, we must specify the value of *every* variable at every point in time. The result is a total ordering on events. This need to specify the value of every variable in turn forces the inference engine to acquire and manipulate a significant number of irrelevant relations. For example, suppose we are interested in studying the behavior of two bears, a panda bear and a polar bear living oceans apart. Specifically we are interested in how a bear's sleep cycle (i.e., when they wake and sleep) affects its eating habits. In a state-based approach we are required to determine the order in which the bears fall asleep. However, this ordering doesn't affect either bear's eating habits since they will never interact. If instead we were studying one hundred bears living in separate remote areas of the globe then the bedtime of every bear would have to be ordered resulting in 10,000 irrelevent orderings! If a particular ordering cannot be determined (as it is often the case in qualitative reasoning), we must split cases, creating an explosion in the number of interpretations. Even in the above example the number of interpretations is clearly unbearable.

The additional relations also obscure the resulting behaviors and dependencies, minimizing their utility for explanation, diagnosis or design. Filtering out irrelevant information at the end of analysis is difficult and computationally expensive.

A second limitation is that the lack of an explicit representation for time restricts the class of analyzable behaviors. Typically we only specify an initial state of the system. Without an explicit representation for time, there is no easy way to describe inputs that vary (e.g., an external clock in a digital circuit). Furthermore, the lack of an explicit representation of temporal relations prevents adequate reasoning about durations, delays and feedback. Finally, it is difficult to change the model for time or the temporal relations allowed without changing the underlying qualitative reasoning mechanism. A solution to the last two problems is the focus of Sections 10 and 11.

In the next section we will see that a history-based approach allows us to separate out the specification of the behavior of each quantity from the description of the interrelationship between these behaviors. It is then necessary to specify only the relevant interactions between behaviors. In addition, relations have become explicit, making it possible for other reasoning mechanisms to use them. This is crucial since it allows us to change our underlying model for time without modifying the predictive inference mechanism.

4 Representing Behavior Over Time

To avoid the above limitations we describe a system's behavior in terms of 1) the behavior of each state variable over time, and 2) the relevant temporal relations between events. For each variable, we are typically interested in intervals of uniform behavior and points at which these behaviors change. For example, we might describe a person's life history in terms of where he lives: "Fred was born in 1910 and raised in Montana, went to school in Massachusetts from 1928 to 1936, and then spent the remainder of his life in Alaska, where he died in 1980." In this example "uniform behavior" means Fred's state of residence is unchanged. In general, what we mean by "uniform behavior" is determined by particular properties of interest based on our analysis goals. For example, we may be interested in the interval over which a state variable maintains a particular value, is bounded within a certain region, or is sinusoidal.

The description of a state variable's behavior over time is referred to as a *value history,* or simply *history.* A history is a contiguous, non-overlapping sequence of interval/value pairs, called *episodes.* The time interval associated with each episode, e, is referred to as the episode's *temporal extent,* and is bounded by the end points $t\text{-}(e)$ and $t\text{+}(e)$. In the above example, the value history for "Fred's state of residence" is composed of three episodes, the first being "Montana from 1910 to 1928." We say that histories use a qualitative representation for time because they break the time line into a set of regions of interest. In this case a region of interest is an interval of uniform behavior. This model of behavior is very general, permitting a variety of discrete and continuous temporal representations. This is discussed further in Sections 10 and 11.

To avoid the limitations noted in Section 3, a history must avoid introducing distinctions that are irrelevant to the analysis. There are many ways of describing a particular behavior which fail to satisfy this property. For example, instead of saying "Fred was in Montana from 1910 to 1928," we could say, "Fred was in Montana during 1910,1911, 1912, . . . 1928." In this case the boundaries between the intervening years are irrelevant and obscure the description. The case becomes absurd if we enumerate the same description in terms of days, minutes, or seconds. Nevertheless, incorporating irrelevant temporal distinctions is a real problem encountered in most existing qualitative reasoning systems.

To achieve the desired descriptions we want every episode in a history to encompass the largest contiguous interval of time during which the state variable maintains a single qualitative value. More precisely, we say that an episode, $e1$, is *maximal* if there exists no episode, $e2$, with the same value such that $e1$'s temporal extent is a proper sub-interval of $e2$'s extent. We then say that a history is *concise* if every episode is maximal. Thus every point in a concise history where two episodes meet denotes a change in value. According to this definition, the example of Fred's residence given at the beginning of this section is a concise history.

Representing behavior in terms of concise histories makes explicit all events of interest (i.e., the changes in values), while suppressing "uninteresting" details. These events can be used in expressing temporal relationships. The set of relevant relations between events provides the second component of a behavioral description. Instead of representing all relations, as in a total ordering, we are only interested in relations between events whose interaction can result in the change of other quantities. The interactions of interest are defined by the system's equations, with each equation typically specifying a single, local interaction. We will see that those interactions which affect the system's behavior can be identified during the analysis process.[2] Given this

[2]This is one solution to what Forbus refers to as the intersection/interaction problem: "Which intersections of histories actually correspond to interactions between the objects?"[5].

representation, the problem remaining is to efficiently generate behavioral descriptions of physical systems.

5 Propagation of Constraints

Most systems performing some type of reasoning about physical systems have been based on *propagation of constraints*[8]. These include a wide variety of applications such as digital, quantitative and qualitative analysis, explanation, synthesis, diagnosis, and troubleshooting. One reason for the pervasiveness of this approach is that constraints naturally reflect the structure of the physical world around us. Because of its generality for physical problem solving, constraint propagation provides a framework in which to reason about temporal behavior. The remainder of this paper incorporates time into constraint propagation, but does so in a way that avoids the limitations described in Section 3. We will see that concise histories play a key role in making this happen. The next section presents a brief overview of traditional constraint propagation.

6 The Basic Constraint Propagator

Constraint propagation operates on cells, values and constraints. A cell contains a single value, while a constraint stipulates a condition that a set of cells' values must satisfy. Cells and constraints can be used to model state variables and equations respectively (e.g., $f = ma$, is represented as a constraint among the three cells f, m, and a). Values can be anything including real numbers, ranges, logic levels, signs or symbolic quantities.

A constraint propagator performs two functions. First, given a set of initial values, constraint propagation tries to assign each cell a value that satisfies the constraints. Second, it tries to recognize inconsistencies between constraints and values, and identify the cause of this inconsistency.

The basic inference step during propagation is to select a constraint that determines a value for a previously unknown cell. For example, if the propagator has discovered values $f = 12$ and $a = 6$, then it can use the constraint $f = ma$ to calculate the value $m = 2$. In addition, the propagator records m's dependency on f, a and the constraint $f = ma$ (typically using a truth maintenance system (TMS)). The newly recorded value resulting from this inference may cause other constraints to trigger and more values to be deduced. Thus, constraints may be viewed as a set of conduits along which values can be propagated. The dependencies trace out a particular path through the constraints that the inputs have taken.

Constraints are very general. A constraint is implemented as a collection of partial functions or *rules*, each involving a subset of the cells mentioned in the constraint. For example, $f = ma$ is implemented as three functions: $f(m, a) = ma$, $m(f, a) = f/a$ and $a(f, m) = f/m$. A function is applied whenever all of its inputs are known. Since the function may be partial, it may not deduce an output value for every set of inputs.

7 Temporal Constraint Propagation

Standard constraint propagation is based on little more than function application. A *temporal constraint propagator (TCP)* adds to this knowledge about time, delay and feedback, using the concise history representation described in Section 4. Separating this knowledge from that specific to qualitative reasoning about continuous systems extends the propagator's range of applicability, including for example both quantitative and digital domains.

The next few sections describe the basic components of TCP. The remainder of this section describes the basic inference step for propagation. This differs from traditional constraint propagation in that the objects being propagated are episodes (i.e., values over time intervals), rather than values. Section 8 describes how recording episodes in terms of histories aids in determining the sets of episodes on which each rule should run. Section 9 describes how newly deduced episodes are checked for consistency and then incorporated into a concise history. To manage and reason about temporal relations, TCP uses a facility referred to as a *time box*. The job of the time box is to answer questions like: "Which of the following episodes end first?" The expressive power of the time box determines the way in which delays, durations and temporal relations can be specified, and is discussed in Sections 10 and 11.

In TCP, values are concise histories, and rules are functions parameterized by time (e.g., $f(m, a(t)) = ma(t)$). New episodes are deduced by applying rules to known episodes. That is, given a rule and an episode for each of the rule's inputs, a new episode is deduced in two parts. First, the extent of the new episode is the *intersection* of each input episode's extent. If this intersection is empty then no new episode is deduced. Second, the value of the new episode is deduced by applying the rule to the values of the episodes corresponding to each of the inputs. For example, given $A = 8$ over an interval $(30, 100)$ and $B = 3$ over $(50, 140)$, then the rule $C = A - B$ is used to deduce $C = 5$ over $(50, 100)$. If the rule is a partial function then it may not return a value; in this case no episode is deduced.

Next, the new episode is recorded in the cell for the rule's result (e.g., the cell for C gets the value 5 with extent $(50, 100)$). We indicate to the time box that the new episode's extent is the intersection of each supporting episode's extent. This information will be used during further propagation to determine the extent of episodes which depend on this new episode. Finally, the propagator uses a TMS to record the new episode's dependence on 1) the applied function, 2) the input episodes, and 3) the deduction that the new episode's extent is non-empty. This dependency information can be used for a variety of tasks, such as explanation, deduction caching, conjectural reasoning, diagnosis and guiding search.

Often a time lag is involved when quantities interact. To model this we can associate a delay with each rule. In the analysis of many systems, this delay is considered infinitesimal; changes propagate almost (but not quite) instantaneously. Infinitesimal delay, along with feedback, is essential for modeling such properties as stability, inertia, memory and causality, and is discussed in Section 11.

8 Histories

To produce the desired behavioral descriptions TCP uses the deduced episodes to construct a set of value histories. Recording sets of episodes as histories has a number of computational advantages. When applying a function to the episodes of a set of input cells, the constraint propagator must determine which combination of episodes and rules will result in new episodes. In a moment we will see that value histories allow us to accomplish this without having to consider the cross-product between the episodes of every input cell.

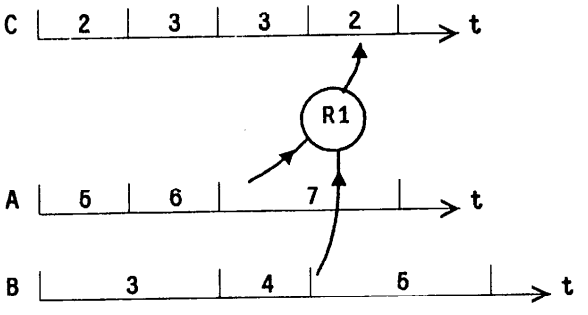
In analyzing a system's behavior we believe that our models are (or should be) internally consistent. Thus we are particularly interested in detecting any inconsistencies. When TCP records a new episode it must be checked for consistency with existing episodes. A cell must be single-valued at any point in time; thus an inconsistency arises if two episodes with differing values over-

lap in extent. Histories allow this test to be performed without having to consider every episode already recorded for the cell. This is discussed in Section 9.

For the moment we assume that every rule is a complete function (i.e., deduces a value for every set of inputs) and that each cell has at most one rule mentioning it as an output cell. To perform propagation, each rule R "walks" along the histories of its input cells from start to end, deducing new episodes and adding them to the end of the output cell's history. Each propagation step consists of 1) applying R to a set of input episodes E, 2) recording the new result at the end of the output cell's value history, 3) determining the set C of one or more episodes in E which end first, and 4) constructing the next set of episodes to be propagated. This is accomplished by modifying E so that every episode which appears in C is replaced by its successor. One episode being the successor of another in a history depends on the two episodes being adjacent. This dependency is also recorded.

The propagation step is repeated on successive sets of episodes for R, moving monotonically forward along contiguous episodes of each of the input histories until the end of one of the input histories is reached. If that input history is later extended, then propagation using R continues.

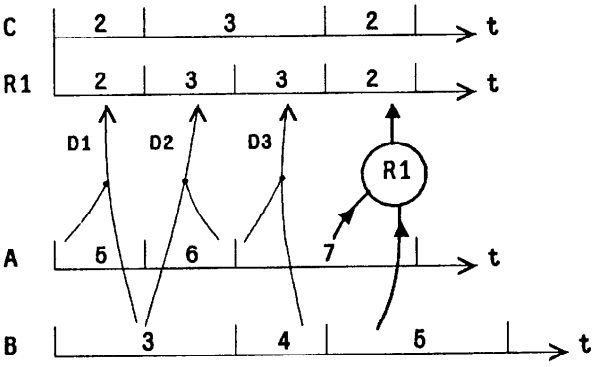
For example, given cells: A, B, and C and rule $R1$: $C = A \quad B$, the following shows how $R1$ is used to deduce the value history for C from A and B:



9 Making Histories Concise

It is often the case that a rule deduces the same value for two successive episodes in its output history; thus, the sequence of episodes deduced is not necessarily concise. Before being propagated the sequence of deduced episodes is "summarized" by accumulating each contiguous sequence of episodes with the same value into a maximal episode. It is this concise history which is then used for further propagation.

To understand how the concise history of values is constructed we need to add one more level of detail to the picture above. Value histories are made concise by constructing them in a two step process. The product of a rule invocation is actually a *justification episode* and is added to the *justification history* for the quantity. A more precise picture is:



where D1, D2 and D3 are dependencies left behind by previous invocations of R1. (Note: the justification history for a quantity is shown directly under the quantity's value history, and is labeled by the rule used to construct it.)

A justification episode contains the value computed along with the input episodes used in computing that value. As suggested above the concise history for C is constructed by summarizing contiguous justification episodes with the same value. Hence, the first two justification episodes for C are summarized into the maximal value episode shown.

So in fact the whole process is one in which rules walk along value histories, producing as output new justification histories, which are then summarized into concise value histories.

Justification episodes allow us to maintain a complete dependency record of the computation, while still maintaining the property that every value history is concise. To avoid irrelevant distinctions in the behavioral description, it is important that justification histories be concise as well. This property is satisfied since each justification episode is concise with respect to the set of dependencies.

Justification histories are an important component of a system's behavioral description. For tasks like explanation and diagnosis, knowing **why** a quantity has a specific value is at least as important as knowing **what** the value is. This is one of the important ways in which TCP differs from traditional simulators. Traditional simulators tell you what the values of each quantity are but not how they came about.

In addition to constructing the justification histories, it is important to record the dependence of each maximal episode on the justification episodes used to construct it; that is, both that the justification episodes have the same value and that their extents are adjacent or overlapping.

Having the propagator operate directly in terms of concise histories is essential. Suppose the sequence of deduced episodes were propagated without summarization. In this case, each successive episode with the same value will be propagated forward separately, rather than propagating forward a single maximal episode which encompasses them. In Section 11 we see that, in the worst case, feedback could cause an infinite number of consecutive episodes with the same value to be created. In addition, this propagation will introduce a number of irrelevant temporal distinctions into the predicted behavior. These result from the fact that each propagation of a set of non-maximal episodes requires an ordering between their ends. If the propagator cannot infer the required ordering, then it must either halt without predicting the rest of the behavior, or split cases on each of the orderings possible, as we saw in section 3. The use of maximal

episodes solves this problem.

Earlier we assumed that a cell is used as an output cell of at most one rule. In general several rules may deduce values for the same cell. Thus for a particular cell we must 1) record a justification history for every rule which uses it as an output, and 2) record the dependence of the value history on each of its justification histories. In addition, we must also make sure that the values deduced in the different justification histories are consistent. To accomplish this we construct a procedure which walks forward along the episodes of each justification history, constructing the concise value history by creating successive maximal value episodes and associating justification episodes with each value episode. If a justification episode, *je*, has a value different from its immediate predecessor, then a maximal episode, *ve*, with this value is added to the value history such that t-(ve) is constrained to be equal to t-(je). If *ve* already exists, then the function checks that *ve*'s value and t-(ve) are in agreement with *je*.

Earlier we also assumed that each function is complete. If a function is partial then it could produce gaps in its justification history where a behavior cannot be predicted. During the extent of this gap the same value may be maintained, or may change several times. Thus, before incorporating into the value history any justification episodes following a gap, we must make sure that 1) the extent of the gap is covered by a combination of episodes from other justification histories for that cell, and 2) these episodes are incorporated into the value history. If the episodes on either side of a gap have different values, then the maximal episode containing the justification episode immediately preceding the gap must end somewhere within the gap.

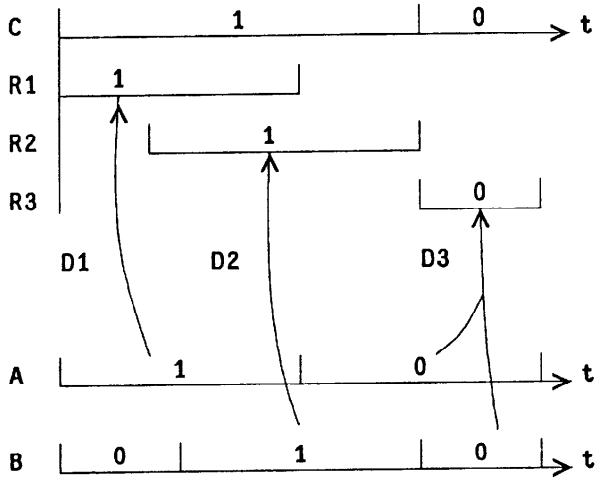
Consider the example consisting of three cells: A,B and C, and the constraint: $C = A$ OR B. This constraint is modeled with three rules, each being a partial function:

R1: If $A = 1$ then $C = 1$

R2: If $B = 1$ then $C = 1$

R3: If $A = 0$ and $B = 0$ then $C = 0$

The following shows the values and justification histories deduced for C, given inputs for A and B. Note that the justification histories for $R1$, $R2$ and $R3$ overlap in places, each contains gaps, and together they cover the two value episodes for C:



To review briefly, in general a constraint propagator carries out four basic operations: 1) select a constraint and set of values, 2) apply the constraint to deduce a new value, 3) record the new value, and 4) check consistency. Section 8 discussed selecting constraints, Section 7 discussed applying constraints, and Section 9 discussed recording values and checking consistency. Note also that the overall goal of a constraint propagator is to tell us *what* will happen (i.e., compute values), *why* it will happen (i.e., record justifications) and to spot inconsistencies.

10 The Time Box

Whenever TCP has a question about the relationship between the extents of different episodes it consults the time box. Separating inferences about time from behavioral prediction produces a system which is more easily extensible and conceptually clear.

The demands placed by constraint propagation on the time box can be characterized in terms of 1) the types of questions asked, 2) the temporal information available, and 3) the inference necessary to answer these questions.

TCP asks questions about temporal relations when applying a constraint, or when incorporating newly deduced episodes into a value history. Questions asked by the temporal constraint propagator have been of the form "Does this episode have a non-empty extent?", "Which of these episodes begins/ends first?", "Are these episodes adjacent or overlapping?", or "Are these relations consistent?" Each of these questions can be reduced to a question about the ordering ($<, =, >, \leq$ or $\geq$) between two or more events. In addition, the events being ordered are either parts of 1) value histories for cells participating in the same constraint, or 2) justification histories for the same cell. Thus no global ordering is required – *all temporal interactions are local to the constraints.*

Information about temporal relations is provided both externally and by the constraint propagator. Information from the constraint propagator is of the form: "episode A is the intersection of the following episodes", "A is contained in B", "these two episodes begin/end at the same time" or "these two episodes meet." Each form, except intersection, can be expressed as a conjunction of endpoint orderings. Intersection is more complex and is discussed later in this section.

Information provided externally is problem-dependent. For many digital and quantitative problems, precise information is available about the exact times that events occur in the input histories. This information might be provided in terms of precise numerical values, upper and lower bounds, algebraic relations (e.g., A occurs 20 seconds after B), or a total ordering. At the other extreme, qualitative reasoning makes as little commitment about temporal relations as possible; i.e., only when they affect the predicted behavior. At this extreme commitments about temporal relations are required by the propagator only when no further inferences can otherwise be made.

Finally, the time box must use the temporal information provided to check consistency and answer queries. Inconsistent information leads to wasted effort exploring wrong paths; thus new information should be checked for consistency before it is used. The number of relations queried is small relative to the number of relations deducible; thus relations should be deduced on demand.

The inference algorithms used in the time box depend on the types of temporal information supplied and on properties of the constraint network. The simplest case occurs when the constraint network has no feedback loops, and the end points of the episodes for each input are specified quantitatively. In this case

the extent of each justification episode can be determined precisely, as the intersection of its support, using simple arithmetic. In addition, the extent of each deduced value episode can be determined from its justification episodes before it is propagated. Thus, answering a query involves a simple arithmetic comparison. In this case TCP acts like an event driven simulator which records dependencies.

A second case occurs when the system has no feedback, but the endpoints are specified qualitatively, through inequalities or upper and lower bounds. This case is similar to the first except that, to answer queries the time box must combine inequality reasoning (e.g., transitivity of > and =) with reasoning about simple arithmetic expressions. In addition, the minimum/maximum of a set of events (used to determine intersection) are determined by querying the ordering between each of the events. Several systems are available with this capability[7,2].

The remaining case to consider involves analyzing systems with feedback using a qualitative representation for time (i.e., specifying only partial orders on events). This is the most interesting class for qualitative reasoning and is the topic of the next section.

11 Modeling Time for Feedback Systems

Feedback is both pervasive and important. Even trivial systems, such as two component circuits and simple harmonic oscillators, exhibit feedback. Without feedback, physical systems would not have state or memory. Properties such as damped oscillation and bistability depend critically on feedback.

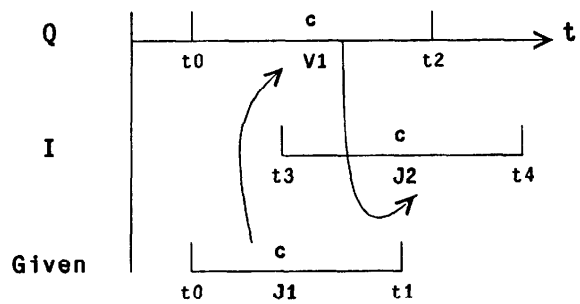
Reasoning with feedback is more difficult than the two cases addressed in the previous section. The problem lies in determining the extent of a maximal value episode. If a system has feedback, then the delay through the feedback loop may cause a quantity to maintain its value (e.g., as in a flip flop). Thus the extent of the episode will depend on itself. This is the important difficulty. If the constraint propagator waits until the extent of the value episode is determined before propagating it, then it cannot determine this cyclic dependency and the extent will not be determined. This is one of the reasons why temporal reasoning systems such as the one described in reference [2] cannot handle feedback.

One solution to this problem is to drop the restriction that episodes be maximal, and instead construct the value histories directly from the result of each propagation.[3] In this case propagating a value episode around the feedback loop will allow a new episode to be deduced, whose extent is shifted by the loop delay. This new episode is also propagated around the feedback loop, deducing another episode and so forth. Thus the propagator begins to count in increments of the loop delay; the problem is that this can continue indefinitely. A second way to deal with this problem is for the constraint propagator to use maximal episodes, but make assumptions about the persistence of particular episodes; this, however, can quickly lead to an explosion in the combinations of assumptions possible.

Fortunately, many forms of feedback can in fact be reasoned about without making any assumptions, simply by having a more robust representation and inference mechanism. TCP handles many cases of feedback by allowing a value episode to be propagated forward before its full extent is determined. Propagation can occur as soon as the episode's existence is known. As each episode is propagated the time box is told about its relation to other justification and value episodes. End points of episodes are represented symbolically and are constrained by specifying relations with other points. Instead of computing each point's value immediately, they are only solved for as needed. Thus constraints between points can be updated with no cost.[4]

To see how this is useful, consider a simple feedback example where a quantity Q is a function of *only* itself (i.e., $Q = f(Q)$). In addition, assume the function is an identity function I and there is some delay d from input to output. From this mathematical model the propagator should be able to deduce that, due to the delay, Q will *never* change its value (i.e. Q has inertia).

Initially we are given that $Q = c$ over an interval $(t0, t1)$ where $t1 - t0 > d$ and c is a constant. This fact is recorded by constructing a value episode $V1$ for Q with value c and extent $(t0, t2)$. This is supported by justification episode $J1$ with extent $(t0, t1)$ and value "Given." $V1$ must at least include the extent of $J1$ thus $t1 < t2$. Next I is invoked on $V1$, producing $J2$ with extent $(t3, t4)$. We know that $t0 < t3$ and $t2 < t4$ because I has delay. Also, $J2$ overlaps $J1$ (since the delay through I is less than the extent of $J1$), thus $J2$ is used to support $V1$ as well. This implies that $t2 \geq t4$, because the extent of $V1$ at least includes the extent of $J2$. This however is inconsistent with $t2 < t4$. The only explanation consistent with these constraints is that $V1$ *never changes*. This situation is depicted below:



Consider what requirements this type of problem places on the time box. The type of argument given above can only be made if 1) the constraint propagator generates these relations for the time box, 2) the time box can recognize inconsistencies, and 3) the time box is given and can manipulate symbolic endpoints. Consider the relations which TCP must record; there are two cases. First, when a justification episode, J, is incorporated into a value episode, V, we use inequalities to record that the extent of J is a subset of V (i.e., t-(V) $\leq$ t-(J) and t+(V) $\geq$ t+(J)). Second, when a justification episode, J, is deduced from a set of value episodes, $V1, V2, ..Vn$, we record that the extent of J is equal to the intersection of episodes in the set (i.e., t-(J) = Max[t-(V1), t-(V2) , ... t-(Vn)] and t+(J) = Min[t+(V1), t+(V2) , ... t+(Vn)]). Min and Max, in turn, can be expressed using inequalities and disjunction. For example, $a = Min(b, c)$ becomes "$a \leq b$, $a \leq c$, and ($a = b$ or $a = c$)". Thus, to reason fully about the temporal relationships deduced during propagation, the time box must be able to handle both inequalities and disjunction. Notice, however, that every relation in the disjunction produced by an expression x = Min[...] mentions the point

[3] This is the approach taken by event driven logic simulators, which use a quantitative representation for time.

[4] If we are interested in recognizing all inconsistencies immediately then we must pay the cost of testing a temporal relation for consistency when recorded.

x.[5] Thus the time box need only deal with a limited form of disjunction, rather than the general case.

As part of this research, a polynomial time algorithm has been developed which answers questions of validity and consistency about relations involving inequalities, given expressions involving inequalities and the limited form of disjunction described above. More specifically, given a set of R relations including D disjuncts, determining whether or not a particular relation logically follows from this set takes worst case time $O(D * R)$.[6] A number of techniques are used for guiding constraint propagation which significantly reduces this time in practice, without sacrificing the completeness or soundness of this algorithm. However, space does not permit a detailed discussion of the algorithm or these techniques here.

Even with a powerful time box, modeling systems with feedback is still a difficult problem. For example, TCP inherits the well known limitation of local constraint propagators in that it is not a complete constraint satisfaction system. A number of existing techniques[10,8] can be used in TCP to solve this problem, depending on the representation for values being used.

12 Qualitative Reasoning Revisited

Given the framework described above, incorporating qualitative reasoning about physical systems into TCP involves 1) mapping time to the reals (e.g., specifying intervals to be open and closed), 2) adding principles of continuity and integration[9], 3) providing a set of constraints which support a qualitative representation and algebra, and 4) expressing the laws of physics in terms of these constraints. Instead of describing each of these steps we demonstrate the approach with a familiar example. Consider a harmonic oscillator, consisting of a mass, M, and a spring, S. Let x denote the position of the mass with respect to its rest point. The spring is extended from its rest point ($x > 0$) and released at time $t0$ with zero velocity. The position of the mass then oscillates back and forth, extending and compressing the spring. The initial part of the oscillation can be explained as follows:

> At time $t0$ the spring is extended from its rest point ($x > 0$) and its velocity is zero. The positive position produces a force on the spring and mass, causing an immediate acceleration. This acceleration causes the spring to begin to move inward towards its rest point immediately after $t0$. Because of the increasing velocity, the spring eventually reaches its rest length ($x = 0$) after a finite interval of time.

We would like to use TCP to generate a prediction corresponding to the behavior explained above. To do this the system is described in terms of the state variables, position (x), velocity (v), acceleration (a) and force (f). The qualitative representation used consists of the sign of each quantity ($+, 0, or -$), and equations used to describe the system are:

(E1) $f_s(t) = kx(t)$ Hooke's Law
(E2) $f_s(t) = -f_m(t)$ Conservation of Force
(E3) $f_m(t) = ma(t)$ Newton's First Law

where the subscripts s and m on force denote mass and spring respectively. k and m are assumed to be positive and finite. For simplicity of presentation we rewrite E1-E3 as:

(E4) $a(t) = -x(t)k/m$

In addition to these basic constraints we incorporate a few special rules. A detailed discussion of the principles underlying these rules is presented in [9] and [10]. We know from continuity that (C1) a quantity moving through an open/closed interval of space takes an open/closed interval of time. Thus, a quantity, Q, will be in the open interval "positive" ($0 < Q < inf$) for an open interval of time and in the closed interval "zero" for a closed interval (possibly an instant). From principles of rates of change (integration) we know that: (I1) if a quantity, Q, is 0 at some instant, tq, and dQ/dt is negative over an interval immediately following tq, then Q will be negative at least over that interval,[7] (I2) if a quantity, Q, is positive and its derivative is bounded above by a finite negative value as long as Q is positive, then Q will become zero in a finite amount of time (although it might remain so for only an instant), and (I3), if a quantity, Q is negative and its derivative is non-positive over an interval, then Q is bounded above by a finite negative value over that interval.

The following naming and notational conventions are used. A pair $\langle v, i \rangle$ is used to denote an episode with value v and interval i. (a, b) denotes the open interval from a to b, and $[a, b]$ denotes a closed interval. A justification episode's value is a list consisting of the rule and value episodes used in the deduction. The nth value episode for state variable x is named "xvn" and the nth justification episode is named "xjn". In the explanation, if two points are equal then the same name is used for both points. Each relation between points is labeled "Rn" where n is an integer. The remainder of this section shows each sentence of the English explanation given above and the corresponding prediction made by TCP. While constructing a complete dependency record is important for many tasks, presenting these details here would obscure the example, so they are omitted.

The analysis begins with the spring extended but not moving ($x > 0$ and $v = 0$ at $t0$):

xv1:	$\langle x = +, [t0, t1) \rangle$	
xj1:	$\langle (\text{Given}), [t0, t0] \rangle$	justifying xv1
R1:	$t0 < t1$	the extent of xv1 contains xj1
vv1:	$\langle v = 0, [t0, t2] \rangle$	
vj1:	$\langle (\text{Given}), [t0, t0] \rangle$	justifying vv1
R2:	$t0 \leq t2$	the extent of vv1 contains vj1

The t- of $xv1$ and $vv1$ are both closed because analysis begins at $t0$. t+ of $xv1$ is open by $C1$ since x is positive. Likewise, t+ of $vv1$ is closed by $C1$ since v is zero. $R1$ holds, since x is positive at least as long as the extent of the given $xj1$; likewise for $R2$. In addition, $R1$ is a strict inequality since the end of $xv1$'s extent is open.

The positive position ($xv1$) produces a force on the spring and mass, causing an immediate acceleration ($av1$). By E4, x being positive over $[t0, t1)$ causes a to be negative over that interval:

av1:	$\langle a = -, [t0, t3) \rangle$	
aj1:	$\langle (\text{E4}, \text{xv1}), [t0, t1) \rangle$	justifying $av1$
R3:	$t1 \leq t3$	the extent of $av1$ contains $aj1$

This acceleration causes the spring to begin to move inward immediately after t0. Specifically, v is 0 at $t0$ ($vv1$) and its derivative, a, is negative immediately following $t0$ ($av1$), so by I1, v becomes negative immediately following $t0$, and remains so as long as a is

[5] Conceptually, if we view each relation as an edge in a graph, and each point as a vertex, then all the edges mentioned in a disjunction are connected to a common vertex.

[6] Answering the same question, but disallowing disjunction takes worst case time $O(R)$.

[7] I1 is actually a special case of the constraint: $[Q(t + epsilon)] = [Q(t)] + [dQ(t + \epsilon)/dt]$, where ϵ is an infinitesimal delay and the notation $[x]$ denotes the sign of x.

negative:

vv2:	$\langle v = -, (t0, t4)\rangle$	
vj2:	$\langle (I1, vv1, av1), (t0, t3)\rangle$	justifying vv2
R4:	$t3 \leq t4$	the extent of vv2 contains vj2
R5:	$t2 \leq t0$	vv2 follows vv1

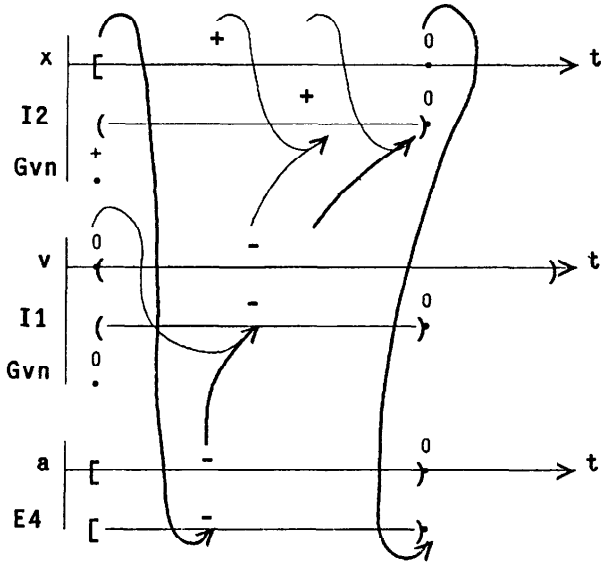
vv1 ends when v is no longer zero and v is negative over (t0,t3) thus vv1 ends at or before t0 (R5).

Because of the increasing velocity, the spring eventually reaches its rest length ($x = 0$) after a finite interval of time. Specifically, 1) x is positive (xv1), 2) v is bounded by a finite negative value over $(t0, t3)$ by I3, since v and its derivative are negative over this interval (vv2 and av1), and 3) v is bounded as long as x is positive, since t+(xv1) = t1 $\leq$ t3 = t+(vv2). Thus by I2, $x = 0$ at $t1$:

xv2:	$\langle x = 0, [t1, t5]\rangle$	
xj2:	$\langle (I2, xv1, vv2, av1), [t1, t1]\rangle$	justifying xv2
R6:	$t1 \leq t5$	extent of xv2 contains xj2

The analysis continues, identifying a deceleration immediately following $t5$, and eventually predicting the oscillation.

The deductions made above are summarized in the following figure:



A causal explanation can be constructed by tracing forward along the dependencies through each justification episode whose start coincides with the beginning of the value episode it supports. In the figure above the chain of dependencies drawn with a thick line corresponds roughly to the causal explanation printed above in italics.

While we have used the oscillator for simplicity of presentation, our system is in fact capable of dealing with considerably more complex devices involving partially ordered, time varying inputs. It has, for example, predicted the bistable behavior of an SR-latch built from cross-coupled NOR gates, accurately modeling the positive feedback that is crucial for latching values.

13 Summary and Research Status

Predicting behavior involves describing both what happens and why. Thus, the propagator must provide a clear description of both a quantity's values and their justifications. We have seen that concise histories are crucial in describing values and their justifications, as was demonstrated in the issue of feedback. TCP provides a clear separation between inferences about time and behavioral prediction. This allows a variety of temporal representations to be used without modifying the propagator itself. Finally, by avoiding unnecessary commitments the resulting predictions are more broadly applicable.

A prototype of TCP was implemented in the fall of 1984, using Simmons' Quantity Lattice[7] as the time box. The power of this approach has been demonstrated on a number of examples taken from digital electronics and arithmetic with time varying inputs. A second time box has been developed that incorporates disjunction. This, along with principles of qualitative reasoning are currently being incorporated into TCP. Plans for the near future include: 1) augmenting TCP to perform envisionment as well as simulation, 2) incorporating techniques for abstracting or approximating constraints used during analysis, and 3) adding a more robust control strategy for guiding this process.

14 Acknowledgements

I would especially like to thank Randy Davis for many hours of help in clarifying these ideas. I would also like to thank the following people for their advice and support during this research: Johan de Kleer, Ron Brachman, Walter Hamscher, Mark Shirley, Reid Simmons, Jeff Van Baalen and Leah Ruby Williams. I would also like to thank AT&T Bell Labs for support and the use of their equipment during the writing of this paper.

References

[1] Allen, J., "Maintaining Knowledge About Temporal Intervals," *Comm. ACM,* 26 (1983), 832-843.

[2] Dean, T., "Planning and Temporal Reasoning Under Uncertainty," *IEEE Workshop on Principles of Knowledge-Based Systems,* Denver, CO, (December, 1984).

[3] de Kleer, J., and Brown, J.S., "A Qualitative Physics Based on Confluences," *Artificial Intelligence,* 24 (1984), 7-84.

[4] de Kleer, J., and Williams, B. C., "Reasoning about Multiple Faults," *Proceedings National Conference on Artificial Intelligence*, Philadelphia, Penn., August, 1986.

[5] Forbus, K., "Qualitative Process Theory," *Artificial Intelligence,* 24 (1984), 85-168.

[6] Kuipers, B., "Commonsense Reasoning About Causality: Deriving Behavior From Structure," *Artificial Intelligence,* 24 (1984), 169-204.

[7] Simmons, R., " 'Commonsense' Arithmetic Reasoning," *Proceedings National Conference on Artificial Intelligence*, Philadelphia, Penn., August, 1986.

[8] Sussman, G.J., and Steele, G.L., "CONSTRAINTS: A Language for Expresssing Almost Hierarchical Descriptions," *Artificial Intelligence,* 14 (1980), 1-40.

[9] Williams, B.C., "The Use of Continuity in a Qualitative Physics," *Proceedings National Conference on Artificial Intelligence,* Austin, TX, (August, 1984).

[10] Williams, B.C., "Qualitative Analysis of MOS Circuits," *Artificial Intelligence,* 24 (1984), 281-347.