"STEPS TO IMPROVE QUALITATIVE SIMULATION"

Andrew F. Toal
A.I.M.G.
E.A.P.S. 3
University of Sussex
Brighton (0273) 606755 extn 4270
England e-mail:paddy@uk.ac.sussex.cvaxa {JANET}
March 1988

[Submitted to "second workshop on qualitative physics" Paris 1988]

This paper reports on two elements of work on qualitative simulation which extend on the basic Qsim algorithm [Kuipers 85]. The work has grown from attempts to examine the limits of current qualitative simulation techniques, specifically a reimplemented Qsim, when applied to models of the heart. The first 'Pragmatics to control simulation' presents ideas which extend the concept of Quality Space which is used in the Qsim system, and argues for a module to control the explosion of state space which occurs when no consideration is given to the meaning of splitting the existing quality space by adding a new landmark. 'Procrastination as a solution to the Qualitative simulation frame problem' presents work on better techniques for navigating the qualitative search space and 'Time to control simulation', reports on work to enhance the basic algorithm by use of temporal information, which is made available as states are generated and by examining previous state chains.

Terminology:

The qualitative simulator used here is a full reconstruction of Qsim in Prolog [Toal 87], the work proceeds with the dual purpose of constructing a large cardiac model to examine the potential for qualitative models as deeper representations in Medical Knowledge Based systems [Hunter 84] and to develop better qualitative reasoning systems. Qsim (and its Q derivative) is a qualitative simulator which generates trees of possible state sequences. A system is described by parameters. A state is a vector of parameter values and successor states are created by generating all legal successor values for each parameter then reconstructing all legal pairings of values. Legal pairings between parameters are expressed by constraints.

- Define, qs(P, t) = the qualitative state of parameter P at t, as having the form: <Value,Dir>
- Define, qs(P, t, t1) = the qualitative state of parameter P during time range <t,t1>

Parameters take values from there own 'landmark' list which is the set of interesting single values it may hold, rather than sets of possible values held as ranges. Value: is chosen as being at one of these landmarks or within a range of two adjacent landmarks. Dir : Direction of change or derivative is one of

- inc : if the parameters derivative is > 0
- std : if the parameters derivative is = 0
- dec : if the parameters derivative is < 0

ie jug_level: quality space = < 0 full inf> may be

- <0,std> : empty and not changing
- <0:full,inc>: filling
- <full, dec> : begining to empty

States go from time points (where at least one parameter reaches a landmark value) to time ranges (which are lengths of time where values are held within ranges) to the next time point. For a full description of the algorithm see [Kuipers 85].

A "Behavior" for the Parameter P is a sequence of states of P: qs(P, t0), qs(P, t0, t1)....., qs(P, tn) A "Behavior" for the System containing P, is the union of the behaviors of all its parameters.

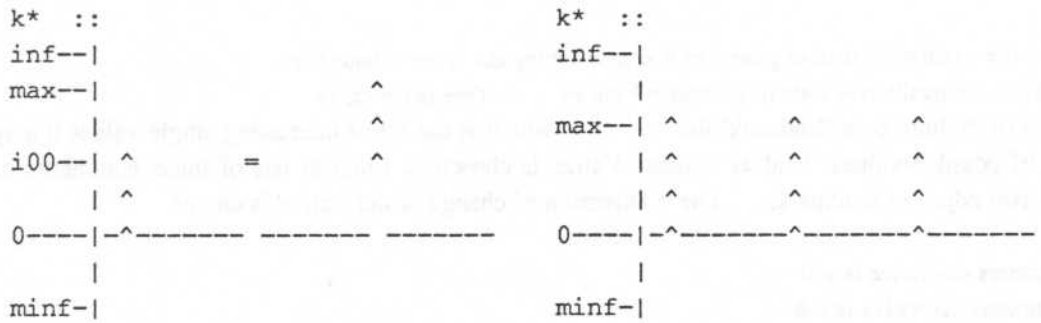
Basic Constraints supplied with the system are:

- add(a, b, c) : Arithmetic operators reimplemented
- sub(a, b, c) : for ranged values
- mult(a, b, c) :
- M0+(a, b) M0-(a, b) M+(a,b) M-(a,b): Monotonic functions exist between parameters constraining how they change
- deriv(a, b) : sign of 'b' defines the direction of change of 'a'

Pragmatics to control Simulation:

Qualitative simulation should operate by the considering parameters which vary among value spaces which reflect the same quality of behavior. The Qsim papers correctly stress the notion that one strength lies in its ability to dynamically split quality spaces by the discovery of new landmark values. This is both the power and curse of the system as the simplicity of the basic algorithms use of landmark creation, causes unwanted proliferation of states generation. The Qsim algorithm first generates potential successor states then filters these with various global filters (ie to check for cyclic behaviour). Having examined and interpreted the results of several simulations we suggest some additional filters (called not very originally 'deamons') which examine the system output and prune out state chains which are syntactically valid under the Qsim transition rules, but are in fact irrelevant, time wasting epiphenomena with no useful information content. The basic arguments for the operation of the deamons follow:

a) Qsim navigates paths through a space of qualitative values held by a parameter over time. Two degrees of freedom are given, magnitude and derivative. In many cases we cannot define both of these values. Consider a parameter k* with the following branched behavior:



```
time-|        t3              t4    time-|        t3              t4
state|1       2       4       5    state|1       3       6       7
```

The purpose of discovery of newlandmarks is to give the simulation a greater expressive capacity. The selection of new ranges to consider during simulation should help constrain the mappings between values, gather new information, but also the system must only make distintions which are qualitatively interesting. If we examine the behaviour around the time point t3 above, the system branches on the possibility of a point of inflection, where k* has a newlandmark. If we consider the behaviour either side of such a time point where we initially have a shared state, which is held for some time range, and then a multiplicity of branches, we have no use for nonunique state sequences, which simply split the landmark space because it is possible, but with no interesting effect on the behaviour information. Consider the two examples above - unless the new landmark for k* is the only option we have the question "Why is it interesting enough to generate a new state branch, when an equivalent branch, but without the newlandmark, is also being expanded?" The answer is, if this is the case, it is not interesting enough to be expanded. It is typical of a class of options, where the general quality of behavior is equivalent (since the final transition options are the same), but the system is generating all possible paths; not the simplest.

Defined in a prolog format:

```
cutBranches(StateA):-

   follow(StateA,[StateB1, StateB2|Others] ),
   forAll(P,parameters), value(stateB1, P, Value1),
                  value(stateB2, P, Value2),
                      Value1 =/= Value2,
                  follow(StateB1, StateC1),
                  follow(StateB2, StateC2),
                  new_Value_Transitions(StateC1, P, NEWPOSS1),
                  new_Value_Transitions(StateC2, P, NEWPOSS2),
     different_by_newlandmark_in_ranges(NEWPOSS1,NEWPOSS2),
                  keep_simplest([StateC1, StateC2]).
```

If we compare state sequences from a shared state and different paths finally generate the same options for some time point, choose the simplest path to keep and throw the options with unnecessary new landmarks. Also Kuipers proof of the Qsim algorithm allows for the fact that landmarks may be passed over several time prior to being discovered. This filter does not effect his completeness proofs, but does present more succinct state sequences where we minimise the unexpressive generation of landmark values.

b) This leads to the general maxim:

"dont assert a new landmark unless it gives useful new information to the system simulation"

Another possible deamon then presents itself, if we consider the information content of a state. Given we have no direct mapping of landmark values onto real numbers, and that one ordinal position is expressive as another, since we are representing the multiple possible mappings from class of functions which we are only expressing qualitatively. We can inhibit generation of states which are simply replicating an existing structure. To explain if

n-parameters are linked by constraints, and k of them generate newlandmarks at timepoint tk, and then generate a second set of newlandmarks at timepoint tk+1, while n-k of them are constant, the second time point state may be syntactically possible but contains no new information. Where one such landmark can exist, we can generate several, which reflects the uncertainty about the specific shape of the relations we are specifying. A system which has to interpret and use the output of these simulations should be aware of this, but enumerating them all serves no purpose and will often just create an infinite branch to the simulation tree.Lets examine this with an example:

Stuttering Tanks Examples:

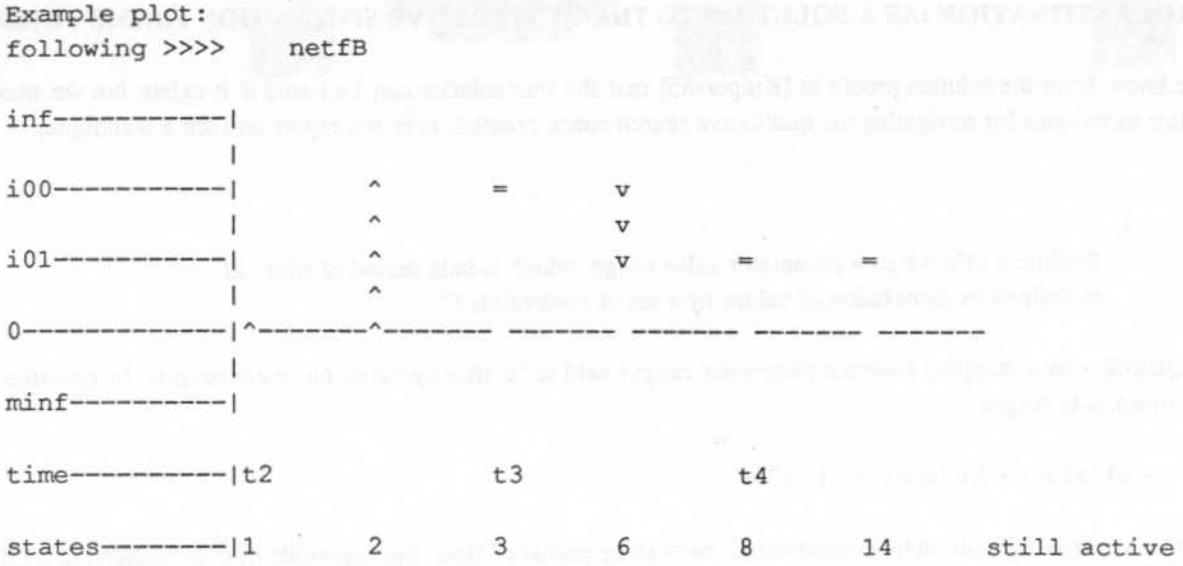
Two tanks with outflow holes, A above B, the

inflow to A is constant, the outflow from A becomes the inflow to B.

```
  .        Parameters: Initial State:      Constraints:
  .
|. | A    inflowA.                    mplus(pressA, amtA).
|__|      [0,if1].   [if1,std].        mplus(pressB, amtB).
  .        pressA.    [0,inc].        mplus(outAinB, pressA).
 |.  | B   amtA.      [0,inc].        mplus(outflowB, pressB).
 |__|      outAinB.   [0,inc].          sub( inflowA,  outAinB, netfA).
  .        netfA.     [ range1(0,inf),dec]  sub( outAinB, outflowB, netfB).
  .        pressB.    [0,inc].        deriv( amtA, netfA).
           amtB.      [0,std].        deriv( amtB, netfB).
           outflowB.  [0, std].
           netfB.     [0,inc].
           constant( inflowA, [if1,std]).
```

a) Basic Qsim Output:

```
?- tree.
state1  with valset  valRSet0  is an start state
doing  state1
O --- o --- o --- o --- o  state4   [marked, incont]
            \-- o --- o --- o  state11   [still, active]
                    \-- o  state12   [still, active]
                     \- o  state13   [still, active]
                \- o --- o --- o  state14   [still, active]
                    \-- o  state15   [still, active]
                     \- o  state16   [still, active]
                  \-- o --- o --- o  state18   [quiet, [state17]]
                   \- o  state10   [quiet, [state6]]
```



so examining the stuttering behaviour shown in the stuttering tanks example, here we have the system generating the newlandmark value i00 for netfB, this is the only option and can mark useful space. However when the system expands to t4/state8 it generates another landmark value i01 (this is allowed because both the inflow and outflow of tankB are increasing and so qualitative arithmetic does not constrain the derivative of the netflow). However this second landmark option tells the system nothing, it knows that some landmark exists at t3 which is the netflow defined by the in/out flow being in ranges. Then at t4 it tells us the same. Unless some connected parameter also changes then the distinction of these quality spaces is spurious and uninformative.

c) Another possible filter is under investigation which is similar to
Kuipers cycle filter, but this requires states to be all landmarks, I
have been examining the information content of systems where the 'same'
states reoccur in the state graph, this may be expanded on later.

The effect of implementation of these deamons has, while not curing the problems of qualitative simulation, has at least cut down the branching problems in a general and intuitive manner on all systems tried. Examine the following example which is taken from Kuipers IJCAI-87 paper [Kuipers 1987], and considers the problem his system has in dealing with stutter in a pair of cascaded tanks. Kuipers basic Qsim trys to build an infinite tree and he devotes half his paper to methods of dealing with this by both ignoring selected derivatives or incorporation of higher order derivatives (although the h.o.d. work is not applied to this example). However running the system with these deamons present gives a finite and understandable envisionment with two branches (the tanks filling together or the higher tank filling first).

These arguments for what appear simple filters are in fact steps in the development of a better language within which to consider the expressive powers of states and using this to control the state generation. From them a more general technique has been developed:

PROCRASTINATION :AS A SOLUTION TO THE QUALITATIVE SIMULATION FRAME PROBLEM:

We know from the solution proofs in [Kuipers85] that the true solution can be found if it exists, but we need much better techniques for navigating the qualitative search space created, here we report on such a technique.

- Define: a QBlock as a parameter value range, which is held period of time, as is defined by association of values by a set of constraints C'.

A QBlock - as a mapping between parameter ranges held to be true byone or more constraints. Ie consider adding the magnitude ranges

$$< a1, a2 > + < b1, b2 > = < c1 , c2 >$$

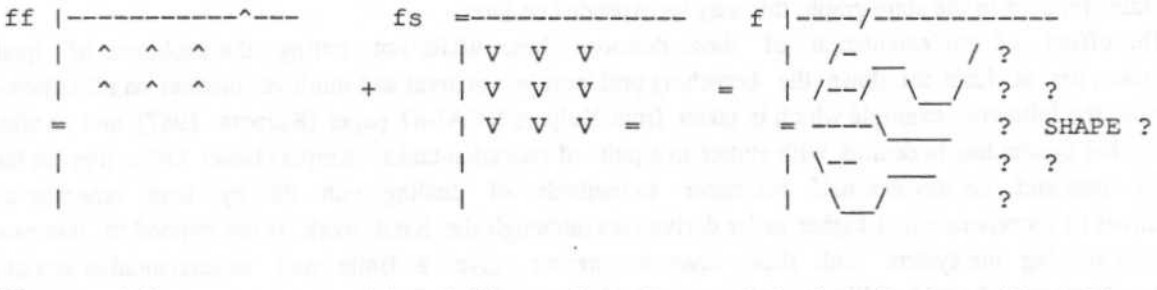
While A and B maintain there ranged values over some period of time, themagnitude of C is bounded in a QBlock.

- Define: a weak-QBlock as a QBlock, where the associated values do not define the derivative completely, with the form [<Val1, Val2>, ???].

Ie addition where one parameter increases while another decreases doesnot define the derivative.

<ValueA, inc> + <ValueB, dec> = < ValueC, {?inc,std,dec?} >

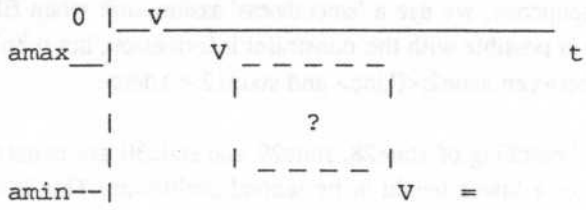
Weak-QBlocks are the cause of several problems in qualitative simulation.Consider the following addition, taken from a model of a damped spring.



The sum of the two parameters is bounded in magnitude, but any derivative value is considered. Since Qsim grows all syntactically possible branches and new 'std' values generate new landmark values, new branches, new world models and more confusion in interpretation. We can consider many paths across these blocks,:

- Define: path(P, Value1, Value2) as a possible sequence of values generated for parameter P between two landmarks values.

Consider this trace of parameter 'a':



With no well defined derivative the number of paths the parameter make take across the QBlock represented by the dotted box, are infinite. Qsim attempts to enumerate this infinite set of paths, and branches in direct proportion to this choice.

Here we can see the Qsim equivalent of the frame problem. When it is growing its infinite set of paths it has no general idea of the structure of the transitions it is growing. It blindly forward chains a single step at a time, rather like simple 'gps' planners stacking and unstacking blocks and never getting any closer to a solution to a problem.

Our solution to this problem is procrastination.
- If we can only bind parameter values to weak-QBlocks do so.
- If we have nondeterministic derivative values, process them as such and wait until deterministic solution allow choices to be made
- If we consider a parameter to be within a qualitative block, then suspend generation of new value sequences for the parameter, offer block transitions instead.

Block transitions:
- if a parameter remains within a qualitative block, is offers the same qualitative information to the system.
- A weak-QBlock [<V1,V2>, ???] can can offer several options
  - it may become a QBlock, one of the defining constraints may become deterministic
  - it may be exited at its maximum or minimum value
  - deterministic magnitude information may alter the range
- Consideration of new 'std' values is complex and can generated a set ofpossible labels.

The basic loop of the system:

i) Run Qsim step
ii) Fuse the confused states, where branching is made on nondetermanistic derivatives
iii) Run the next state ? <QBlock transitions applied automatically>
iv) Reconstruct where possible, either by derivative or magnitude
v) loop.

The Qsim system is enhanced with an extra filter which ensures the consistent use of QBlocks. The mechanism of this and the specifics of the transitions will be detailed in a longer report. Instead let us examine the technique by example. Attached you will find a procrastinated version of Kuipers 'Spring with Damping Model'.

Notice the dramatic difference in branching generated with the procrastinating algorithm. A state sequence grown to equivalent depth with the Qsim system would offer a complicated set of hundreds of options. As can be seen from state0/time t0, on the graph, this system is a spring stretched to some initial length 'xint' and released. The graph shown is the correctly predicted damped case. The other 'quiet' state chain is the other perfectly damped (ie

the frictional force is so strong the spring just returns to a stable rest position). Both of these would be generated by the simple qsim system, but selecting them as correct solutions from the very dense tree shown, is not trivial. When we wish to interpret the meaning of the qualitative sequence, we use a 'smoothness' assumption when filling in the blank state paths (ie state2-state8-state9). Any path is possible with the constraint information, but rebuilding with maximum smoothness would generate a single peak between state2:<0,inc> and state12:<0,dec>.

The problem is, as ever, not completely solved. The branching of state28, state29 and state30 are caused since the system cannot decide if it cycles, or reaches higher or a lower height in its second oscillation. This is exactly the problem Kuipers finds with extended running of his simple perfect (ie non-damped) spring model. Our proposed line of research to solve this problem is based on Qsims ignorance of temporal durations and comparative derivatives. These will be discussed in the next section, since they are not artifacts open to solution by procrastination.

The Simple ball system has single result:

```
?- tree.
state0 with valset valRSet0 is an start state

doing state0
>>>> o --- o --- o --- o --- o --- o state5   [marked, complete]
```

Slowball System: : Simulation of the ball system with the ball given a slower initial velocity. The system has the landmarks and corresponding value information from a 'simple ball throw'. Note qsim has not got the reasoning power to cope with the simulation.

```
Initial Landmarks:     Start State:    Constraints:
y.
[minf,0,ymax, inf].      [ 0 , inc ].       deriv( y, v).
v.                                        deriv( v, a).
[minf,0, vmin,vmax,inf]. [ vmin, dec].
a.                                        constant( a, [g, std ] ).
[minf,g,0,inf].          [ g, std ].

RUNNING THE SYSTEM:
?- tree.
state0 with valset valRSet0 is an start state
doing state0
>>>> o --- o --- o --- o --- o --- o state11   [marked, complete]
            \-- o --- o --- o --- o --- o --- o --- o state16 [still, active]
             \- o --- o --- o --- o state13   [marked, complete]
```

A three way branch occurs when starting with smaller initial velocity, as the system cannot decide if the ball reaches same/less/greater height (sound familiar in the light of spring systems discussed by Kuipers).

Time to control simulation:

Our work has shown the need for multiple knowledge sources and representations in Qualitative reasoning, (for instance earlier work on an electrical model for the heart which is not best represented as a Qsim constraint network, but must integrate with such a model of the hearts structure to reason about cardiac activity [Toal 88]). Here we investigate the use of reasoning with Time. This is a much shorter level of analysis than was intended at this time, and is placed here to show one of the areas from which we hope to find a solution to the type of branching we are still left with, even when using procrastination as a search heuristic.

The weakness of the Qsim algorithms use of temporal information and a solution to some of the problems is easily constructed. Examine the example above, it is another ball example, only in this case the ball is thrown a second time, but slower. What is the predicted result? What does Qsim learn from the initial ball throw?
The single ball output has only one state sequence (up then down). Now examine the output of 'slowball*'. This model is the same ball thrown up again only with a slower initial velocity. The qsim system has grown new landmarks but these do nothing but cloud the value space. Notice the 3 way branching as qsim considers behaviours where the ball gets as high/higher or stops lower down! Obviously if we throw a ball with less initial velocity it cannot reach as high - but qsim cannot make this deduction.

The change of a parameter over time is related to its derivative. Indeed we can consider a qualitative version of integration, where the change in a parameter over time is equal to the sum of parts of its derivative over the same time period. We will currently restrict consideration to function varying between known landmark values:

- Define: Qualitative parameter segment -
  qf(Parameter(timepoint1), Parameter(timepoint2)), to represent the
  segment of the continuous qualitative function which is Parameter(t),
  bounded by the two known landmark values:
  Parameter(timepoint1) and Parameter(timepoint2).

- Define: qf-duration(t1, t2)
  to be the duration of a qualitative time segment defined by
  timepoints t1 and t2. Its value is the difference between the two timepoints

- Define: Sum of qualitative values of a segment of a parameter:
  sum(qf(P(t1), P(t2)), qf-duration(t1, t2))
  To be the summation of the values held by P over the duration given.

- Define: change(P, t0, t1) to be the change in parameter P between two timepoints
  it may be represented as initial and final value <P(t0), P(t1)> or as
  a magnitude of change | P(t1) - P(t0) |.

- Define: P' to be the parameter related to P such that the constraint deriv(P, P')
  holds

- Define: dR(P, t0, t1) to be the range of values of P', the derivative of the parameter P, between time t0 to t1, inclusive.

We can now relate the change of a parameter to its derivative P' between two timepoints t0 and t1.

change(P, t1, t2) = sum(qf(P'(t1), P'(t2)), qf-duration(t1, t2))

For example the change in the distance travelled is the sum of the velocity over theduration being considered.

We can compare the relative magnitudes of these sums, to decide on the relative magnitudes expected of final states for a system which is run from more than one start state. If we examine the early Kuipers model of a bouncing ball it is unable to decide if the second bounce is as high, higher or lower than the first. It would be hoped that reasoning about the final magnitudes of parameters by considering the derivatives and durations involved may be a step towards a solution. It is obvious that such reasoning is beyond the simple Qsim algorithm since it holds time points only as a simple sequence, with no representation for the durations between time points. We can compare relative changes, without having to calculate them explicitly.

Consider comparing the difference in the net change in a parameter 'x', caused by two different qualitative functions of its derivative 'v' :

a) qf(v(t0), v(t1)) let Da = qf-duration(t0, t1)

b) qf(v(ta), v(tb)) let Db = qf-duration(ta, tb)

change(Y, t0, t1) = sum(qf(v(t0), v(t1)), Da)
change(Y, ta, tb) = sum(qf(v(ta), v(tb)), Db)

We have the trivial case where: t0=ta and t1=tb and the changes are the same since we are looking at the same action in time. But how else can we compare the relative sums? The system so far specified leaves a search space bounded by the relative magnitudes of the main parameters

Time:	v(t0):::v(ta)	v(t1):::v(tb)	v(t0):::v(tb)	v(t1):::v(ta)
Da > Db	v(t0) > v(ta)	v(t1) > v(tb)	v(t0) > v(tb)	v(t1) > v(ta)
Da = Db	v(t0) = v(ta)	v(t1) = v(tb)	v(t0) = v(tb)	v(t1) = v(ta)
Da < Db	v(t0) < v(ta)	v(t1) < v(tb)	v(t0) < v(tb)	v(t1) < v(ta)

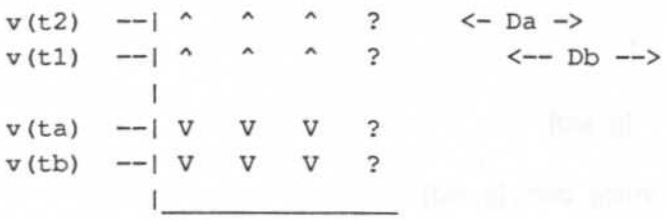
In a paper as short as this I do not intend to give complete consideration to all combinations of these, but we can make deductions about some. Some are impossible, and this becomes a typical consistent graph labelling problem. Other situations are well defined:

IF v(ta) < v(t1) and v(tb) < v(t1) and v(t1) < v(t0) THEN

sum(qf(v(t0), v(t1)), Da) < sum(qf(v(ta), v(tb)), Db)
WHEN Da < Db OR Da = Db

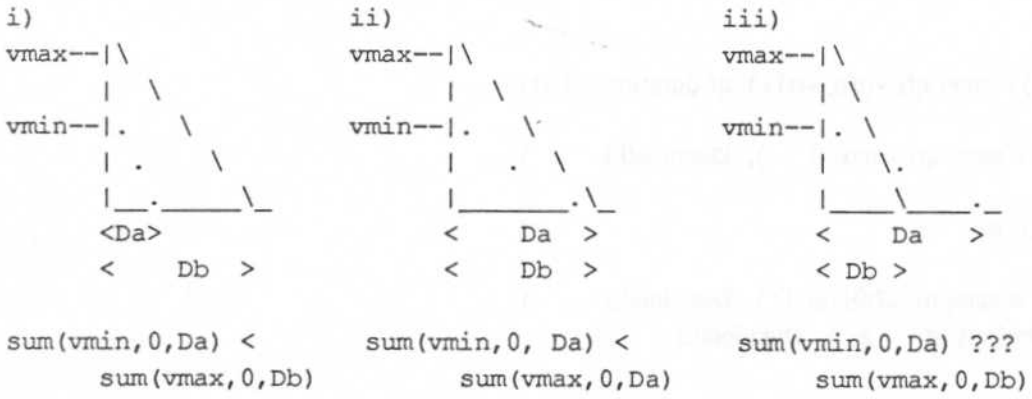
AND is unknown without further analysis when Da > Db

This is because we are considering two functions, such as those depicted below, which never overlap. Since one is always greater than the other, the sum over the same (or less time) of smaller elements, must always be the smaller. It the time span of the smaller segments is greater then its relative magnitude may not be defined at this stage.



Instead of considering all the cases, some of which are nondetermined. Consider the cases, where we know the end two values are the same, and less than the start values.

```
v(t1) = v(b)   ; for this case say 0.
```




```
sum(vmin,0,Da) <        sum(vmin,0, Da) <        sum(vmin,0,Da) ???
      sum(vmax,0,Db)           sum(vmax,0,Da)           sum(vmax,0,Db)
```

However we can decide which case we have by comparing the derivatives of the curves. This can unravel even some complex cases, since starting from different values the derivative of the higher curve must be greater than lower, for some section, for them to intersect. Again the possible cases are numerous and it is my intention only to indicate the technique by examination of the slowball example. Also applying the change rule recursively gives

change(v, t0, t1) = sum(qf(v'(t0), v'(t1)), qf-duration(t0,t1))

When we have constant derivatives: If dR(v, ta, tb) = dR(v, t0, t1) then we must have case 1, since it takes longer to cause a larger change in magnitude, with the same derivative.

Examine the output from the normal ball example:

ball1: [Y V A]

```
t0: state0   [0, inc],          [vmax, dec],          [g, std]
      |
  : state1   [range1(0, inf), inc], [range1(0, vmax), dec], [g, std]
      |
t1: state2   [ymax, std],       [0, dec]              [g, std]
```

Compare this to the output sequence from the slowball:

```
ball2:  [   Y          V          A   ]

ta:state0    [0, inc],          [vmin, dec],          [g, std]
   |
  state1     [range1(0, ymax), inc], [range1(0, vmin), dec], [g, std]
  | | \
tb:| \ state2 [ymax, std],      [0, dec],             [g, std]
tb:|  state3  [ymax, inc],      [range1(0, vmin), dec], [g, std]
tb:state4    [y00, std],        [0, dec],             [g, std]
```

When comparing these we have some coincidence of values, the final magnitude for velocity is the same [0] in two of the cases, and it is known that the initial velocity for the second case is less than the first. Also both cases share the same initial value for Y: [0,inc] and have the same acceleration function A.

From

change(Y, t0, t1) = sum(qf(v(t0), v(t1)), qf-duration(t0, t1))

<0, ymax> = sum(qf(vmax, 0), Duration0.1)

Also since deriv(v, a):

change(v, t0, t1) = sum(qf(a(t0), a(t1)), Duration0.1)
= sum(qf(g , g), Duration0.1)

change(Y, ta, tb) = sum(qf(v(ta), v(tb)), qf-duration(v , ta, tb))

Case1: change(Y, ta, tb) = <0, ymax> = sum(qf(vmin, 0), DurationA.B)
change(v, ta, tb) = <vmin, 0> = sum(qf(g, g) , DurationA.B)

case 1.1 DurationA.B = Duration0.1
Since qf(a(ta), a(tb)) = qf(a(t0),a(t1)). If durations are the same
change(v, ta, tb) = change(v, t0, t1)
<vmin, 0> = <vmax,0 >
INCONSISTENT!

case 1.2 DurationA.B > Duration0.1
Since qf(a(ta), a(tb)) = qf(a(t0),a(t1)). If DurationA.B is greater

change(v, ta, tb) > change(v, t0, t1)
<vmin, 0> > <vmax, 0>
INCONSISTENT!

case 1.3 DurationA.B < Duration0.1
Since qf(a(ta), a(tb)) = qf(a(t0),a(t1)). If DurationA.B is smaller
change(v, ta, tb) < change(v, t0, t1)
<vmin, 0> < <vmax,0>

But: change(Y, ta, tb) = sum(qf(v(ta), v(tb), DurationA.B)
= sum(qf(vmin , 0), DurationA.B)
Since we know the v' values are equal we have case i) from above,
sum(qf(v(ta),v(tb)), DurationA.B) < sum(qf(v(t0),v(t1)), Duration0.1)
Thus: change(Y, ta, tb) < change(Y, t0, t1)
< 0, ymax> < <0, ymax>
INCONSISTENT!

Case2: change(Y, ta, tb) = <0, ymax> = sum(qf(vmin, 0), DurationA.B)
change(v, ta, tb) = <vmin, [vmin:0]> = sum(qf(g, g) , DurationA.B)
Is left as an exercise.

Only one consistent combination of durations and magnitudes can be found, that

change(Y, ta, tb) = <0,ymin>
change(v, ta, tb) = <vmin,0>
qf-duration(ta,tb) < qf-duration(t0,t1)

Which is one of the three options the weaker inference engine of Qsim leaves as an option. In the full paper these ideas were to have been applied to the more complex damped spring case, and used to show how we can reason about the sequences of parameter values to show that the velocity of the spring mass as it crosses the point of zero extension gradually decreases and how we can better control the potentially chaotic behaviour of the simulation. The management of the temporal information has been shown to be possible in the scope of work done within our group using Allens temporal logic [Allen 81], by Ian Hamlet and J.R.W. Hunter [Hamlet 87]. However the more complex case structure required for this is still under construction, although we still believe an amalgam of procrastination and improved temporal reasoning can solve several problems within the qualitative simulator.

Conclusion:

This report is a brief overview of work attempting to improve qualitativesimulation by considering techniques which help in the navigation of the search of a qualitative spacw. The meaning of quality, better use of temporal information and procrastination are considered and shown to be very useful.

Damped Spring Behaviour:

Physics: force = friction+ expansion tension

acceleration - relates_to - force

friction - relates_to - velocity * damping

tension - relates_to - stretch (x)

Constraints:

deriv(x,v).

deriv(v,a).

mplus0(f,a). ;;; net force (ma) and acceleration

mminus0(ff,v). ;;; friction force @ velocity

mminus0(fs,x). ;;; spring force @ extension

add(fs,ff, f).

BASIC QSIM: Massive Branching with this model:

?- tree.

state0 with valset valRSet0 is an start state

doing state0

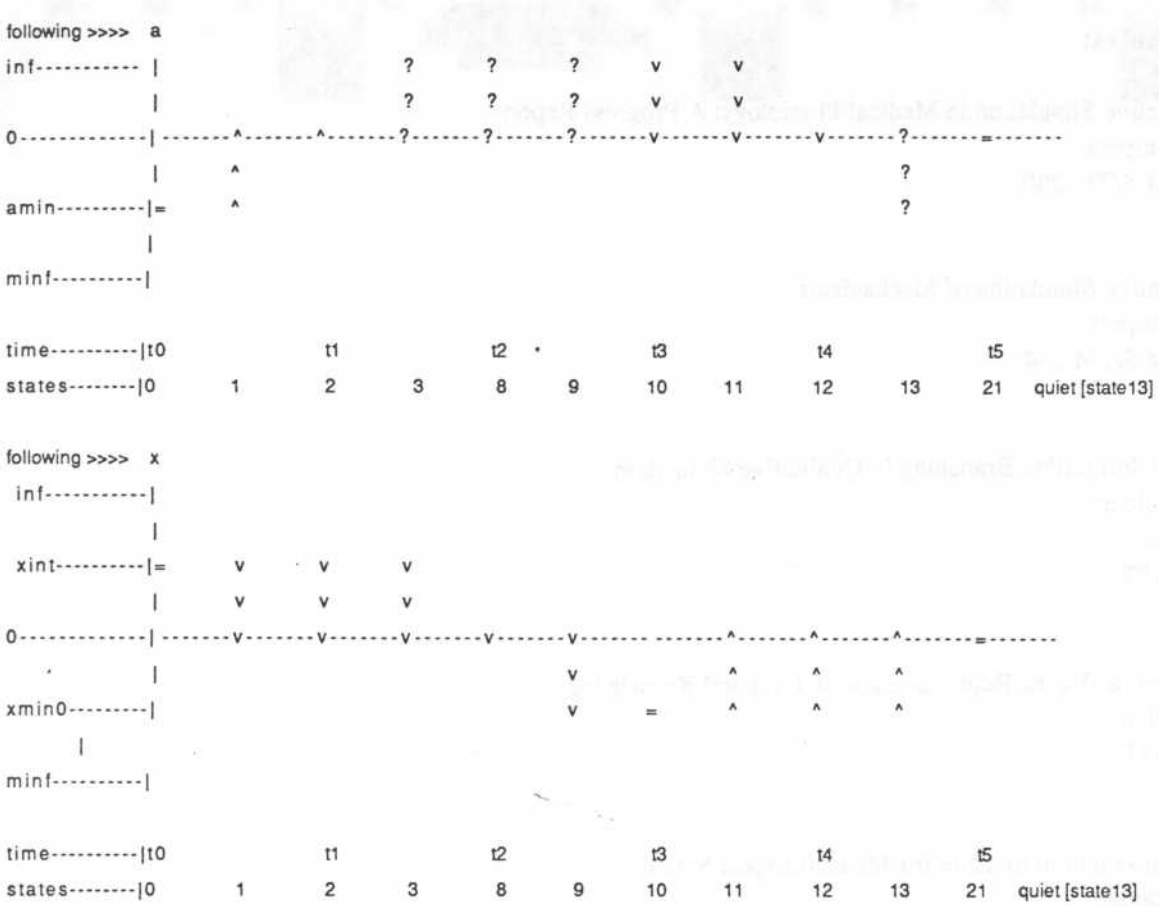
```
>--- o --- o --- o --- o state3  [marked, incont]
          \-- o --- o --- o state9  [marked, incont]
                \-- o --- o --- o state28  [still, active]
                      \-- o  .     .     .
                 . .  .  .  .     .     .
                \- o --- o --- o  .     .     .
                      \- o state51  [still, active]
```

A DAMPED SPRING RUN WITH THE PROCRASTINATING ALGORITHM:

state0 with valset valRSet0 is an start state

doing state0

```
>--- o --- o --- o --- o --- o state7  [quiet, [state3]]
          \-- o --- o --- o --- o --- o --- o --- o state21  [quiet, [state13]]
                      \-- o --- o --- o --- o --- o --- o state29  [still, active]
                                \-- o --- o state30  [still, active]
                                \- o state28  [still, active]
```


following >>>> a
inf----------
0----------
amin----------
minf----------
time----------|t0 t1 t2 t3 t4 t5
states--------|0 1 2 3 8 9 10 11 12 13 21 quiet [state13]
following >>>> x
inf----------
xint----------
0----------
xmin0---------
minf----------
time----------|t0 t1 t2 t3 t4 t5
states--------|0 1 2 3 8 9 10 11 12 13 21 quiet [state13]

References:

Qualitative Simulation in Medical Physiology: A Progress Report
B.J. Kuipers
MIT/LCS/TM-280
1985

Qualitative Simulation of Mechanisms
B.J. Kuipers
MIT/LCS/TM-274
1985

Taming Intractible Branching in Qualitative Simulation
B.J. Kuipers
C. Chiu
IJCAI-87
1987

An Interval-Based Representation of Temporal Knowledge
J.F. Allen
IJCAI-81
1981

A Representation of Time for Medical Expert Systems
I.M. Hamlet
J.R.W. Hunter
AIME-87
1987

An Intelligent Model Based System for diagnosis in
Cardiology - a research proposal
J.R.W. Hunter
N. Gotts
AIMG-7 Sussex University
1984

Qualitative Models within Medical Reasoning Systems
A.F. Toal <to appear in>
British Medical Informatics Society,
Medical Informatics 1988

"Qsim & Cardiology = ?"
A.F. Toal
A.I.M.G. Internal Report
Presented at: Workshop on Qualitative Modelling, Jozef Stefan Institute
ed: T. Urbancic I. Bratko IJS DP-5019 (1988)
Yugoslavia, 1987