Improved Filtering for the QSIM Algorithm

A. C. Cem SAY and Selahattin KURU

ABSTRACT

We have identified a source of spurious predictions inside Kuipers' qualitative simulation algorithm QSIM. Our proposed solution involves the use of *interval corresponding values* for filtering inconsistent states and does not require any additions or restrictions in the input set, unlike the other approaches to the spurious prediction elimination problem. The time and space complexities of the algorithm are not affected by the modification.

I. INTRODUCTION

The QSIM algorithm for qualitative simulation [1,2] is widely used in the field of qualitative reasoning. QSIM takes as input a description of a physical system in terms of constraints representing relations between the system's parameters. In addition to this model, the qualitative values of the parameters at the initial time point are also given. The algorithm involves creating all the possible qualitatively different successors of this state and expanding newly created states in the same manner, therefore building a tree of states, each branch of which corresponds to a future behavior of the system.

Along with all the possible behaviors that can be exhibited by systems described by the input model, the QSIM algorithm produces, for some inputs, *spurious* behaviors; behaviors which no such system would have. Research has been going on [3-5] on extensions to QSIM which will reduce the number of spurious behaviors in the algorithm's output. As a result of these studies, new types of constraints have been proposed. The incorporation to the algorithm of some of these constraints requires addition of extra items of information to the input set (like system-specific knowledge) or restrictions on the possible relationships between the system's parameters (like the sign-equality assumption [3,4].) The non-intersection constraint [5], making use of the qualitative phase space framework, is currently only applicable to second-order systems.

We introduce a modification to the algorithm which allows it to detect a class of spurious

behaviors. Our method requires no changes to the input set and its area of applicability is the same as that of the original algorithm, since it makes use of information already possessed but not used by the "pure" QSIM. Examples of the utility of the improved algorithm and an evaluation will be presented.

II. QSIM PRELIMINARIES

The *parameters* of a system modeled in QSIM are continuously differentiable functions of time. Each parameter has a *quantity space*; a totally ordered collection of symbols (*landmarks*) representing important values that the parameter can take. QSIM has the ability of discovering new landmarks during simulation. The points and intervals in its quantity space make up the set of possible *qualitative magnitudes* of a parameter. The *qualitative direction* of a parameter is defined to be the sign of its derivative; therefore its possible values are: **inc** (+), **dec** (-) and **std** (0). A parameter's *qualitative state* is the pair consisting of its qualitative magnitude and qualitative direction. The collection of the qualitative states of its parameters make up the state of a system.

The "laws" according to which the system operates are represented by *constraints* describing time-independent relations between the parameters. At each step of the simulation, QSIM uses a set of transition rules to generate all possible "next" states of the parameters. The combinations of these states are filtered so that only the ones which constitute complete and legal states for the system remain. The constraints are used as predicates during this filtering; every constraint must still be satisfied by the newly proposed states of its parameters.

There are six types of constraints in the pure QSIM: ADD, DERIV, M+, M-, MINUS, and MULT[1]. Each type of constraint imposes a different kind of relation on its arguments. For example, if we have the constraint M-(A,B), any combination of parameter states in which parameters A and B have the same (nonzero) sign in their directions will be filtered out. Sometimes, additional knowledge about the constraints allows further filtering. In the above example, if we know that the parameters A and B had the point magnitudes A^* and B^* at the same moment at some time in the past, we can eliminate all state combinations in which A and B have magnitudes both less (or both greater) than these landmarks. A^* and B^* are called *corresponding values* (CVs) of that constraint.

Each constraint in a model (except those of the DERIV type) can have such CV tuples. During simulation, if all the parameters involved in a constraint are at landmark values at a particular time, then

the tuple composed of these values is added to the CV list of that constraint and this information is used at the later stages of the simulation to check proposed parameter value combinations.

The CV consistency checks for the ADD and MULT constraints are accomplished as follows: For the ADD constraint, any newly proposed magnitude triple $< x, y, z >$ must satisfy the equation

$$(x - p) + (y - q) = (z - r) \qquad (1)$$

for all CV triples $< p, q, r >$ of that constraint. (The terms can take the values +, -, or 0.) The justification is that, $p + q = r$ is known to be true, and $x + y = z$ is required to be true. Therefore Equation 1 must hold if the new magnitudes are consistent. For MULT, similarly,

$$(\frac{x}{p}).(\frac{y}{q}) = (\frac{z}{r}) \qquad (2)$$

has to be satisfied. (See [1] for a discussion on these equations, as well as a much more detailed explanation of QSIM.)

III. THE PROBLEM

Let us begin with an example showing the kind of spurious behavior that will be dealt with. Consider a system consisting of a ball thrown upward in a descending elevator. The elevator is going down with constant speed in an underground shaft. The parameter representing the elevator's upward velocity, EL_V, is therefore fixed at a negative landmark value. The elevator floor's height, EL_H, is at the negative landmark elh_0 at the beginning of the simulation, t_0. Ground level corresponds to zero in EL_H's quantity space. We are interested in the ball's height relative to the elevator floor; the parameter REL_H represents this quantity. Two positive landmarks of REL_H are known; $relh_0$ is the relative height of the ball at t_0, and win_h is the height of the elevator's window ($relh_0 < win_h$). The parameter BALL_H is the height of the ball relative to the Earth; zero is ground level. Note that, as a result of these definitions,

$$\text{EL_H} + \text{REL_H} = \text{BALL_H}$$

at all times. BALL_H has two negative landmarks; blh_0 is its magnitude at t_0 (therefore, $elh_0 + relh_0 = blh_0$) and blh_1 is another landmark, such that $blh_0 < blh_1$. The ball's flight is governed by gravity. The upward acceleration of all falling bodies, including the ball, is the parameter ACC, which has the constant magnitude g (a negative landmark.) The velocities of the ball relative to the elevator, REL_V, and relative to the Earth, BALL_V, are also parameters.

This system is described by the constraints shown in Table 1. Note that various "realities" which would violate these constraints and are not relevant to our discussion have not been modeled. So the shaft is infinitely deep, the elevator has no ceiling which might interrupt the ball's flight, and we do not care what happens after the ball hits the elevator floor.

Now consider Table 2. This is part of one of the behaviors that pure QSIM predicts when started on this system with the initial state described above. Only the values and quantity spaces of the height parameters have been shown in the table, recall that these are bound by the constraint ADD(EL_H, REL_H, BALL_H). The quantity spaces of these parameters at t_4 in this behavior are presented in Table 3.

TABLE 1 Elevator/ball system model

CONSTRAINT	CVs
DERIV(EL_H, EL_V)	
DERIV(REL_H, REL_V)	
DERIV(BALL_H, BALL_V)	
DERIV(REL_V, ACC)	
DERIV(BALL_V, ACC)	
ADD(EL_H, REL_H, BALL_H)	(elh_0, $relh_0$, blh_0)
ADD(EL_V, REL_V, BALL_V)	

TABLE 2 Spurious behavior of elevator/ball system

EL_H	REL_H	BALL_H	time
<elh_0, dec>	<$relh_0$, inc>	<blh_0, inc>	t_0
<(-∞, elh_0), dec>	<($relh_0$,win_h),inc>	<(blh_0, blh_1), inc>	(t_0,t_1)
<(-∞, elh_0), dec>	<win_h, inc>	<blh_1, inc>	t_1
<(-∞, elh_0), dec>	<(win_h, ∞), inc>	<(blh_1, 0), inc>	(t_1,t_2)
<(-∞, elh_0), dec>	<(win_h, ∞), inc>	<$NewB$, std>	t_2
<(-∞, elh_0), dec>	<(win_h, ∞), inc>	<(blh_1, $NewB$), dec>	(t_2,t_3)
<(-∞, elh_0), dec>	<$NewR$, std>	<(blh_1, $NewB$), dec>	t_3
<(-∞, elh_0), dec>	<(win_h,$NewR$),dec>	<(blh_1, $NewB$), dec>	(t_3,t_4)
<(-∞, elh_0), dec>	<win_h, dec>	<(blh_1, $NewB$), dec>	t_4

TABLE 3 Quantity spaces of height parameters at t_4 in Table 2

PARAMETER	QUANTITY SPACE
EL_H	{-∞, elh_0, 0, ∞}
REL_H	{-∞, 0, $relh_0$, win_h, $NewR$, ∞}
BALL_H	{-∞, blh_0, blh_1, $NewB$, 0, ∞}

To see that this is indeed a spurious behavior, inspect the parameter values at t_1 and t_4. At t_1, the

ball is passing the level of the elevator's window in its upward journey, and at that moment, its "absolute" height is known to be blh_1. As expected, the ball rises for a while, and then starts descending, passing by the window for a second time at t_4. QSIM predicts that at that time, the ball's (and so the window's) absolute height is in the interval $(blh_1, NewB)$, in other words, higher than blh_1. But this does not make sense, since the elevator is known to be descending all along, its window cannot be higher at t_4 than at t_1. So the behavior of Table 2 is a spurious prediction; this sequence of events will never occur.

IV. THE CAUSE

The spurious behavior seen in the previous section is not a result of the "weakness" of the set of constraints, or a fundamental shortcoming of qualitative simulation. All the information required for identifying that the qualitative state at t_4 is inconsistent with the past behavior of the system is available to the algorithm: The constraints between the parameters are given as input, and the states up to t_4 are created by the algorithm itself. The problem is the algorithm's inability to utilize this information.

QSIM uses corresponding value information to check the consistency of newly proposed parameter magnitudes with previous knowledge about the arithmetic and functional relationships in the system. However, the tuple of magnitudes at t_1, which allowed us to recognize that something is wrong, is *not* recorded as CVs of the constraint by the algorithm. The reason for this is that only landmarks are allowed in CV tuples, and there is a parameter which has an interval magnitude at the time of interest. In fact, EL_H has an interval magnitude in each state after t_0 in all behaviors of the system, so no CV tuples will be recorded for any constraint involving it.

Moreover, it would not be of much use even if pure QSIM did record such CV tuples. Equation 1 is used as the ADD CV consistency check, which would, when comparing the values at t_4 and t_1, require that

$$((-\infty, elh_0) - (-\infty, elh_0)) + (win_h - win_h) = ((blh_1, NewB) - blh_1) \qquad (3)$$

be satisfied. This would reduce to

$$? + 0 = +$$

where "?" can be any one of +, 0, or -, so the consistency check would be satisfied, the state at t_4 would be generated, and the same spurious behavior would still be produced.

The reason for pure QSIM's insistence on landmarks as CVs may be the fact that neither qualitative subtraction nor (provided the signs are the same) qualitative division are ambiguous when one of

the values participating in them is guaranteed to be a landmark. When both operands are allowed to be intervals, one may be faced with a situation where one has to compare an interval with itself, as in Equation 3, which leads to the condition being satisfied trivially.

However, we have demonstrated that the use of only-landmark CVs causes the generation of spurious behaviors. What is needed is a way of recording *interval CVs* (*ICVs*), and an extension of the consistency filtering and qualitative arithmetic rules to handle this generalization.

V. THE SOLUTION

To enable it to use more of the available information about the system's arithmetic and functional relationships, we have made the following changes to pure QSIM, [1] resulting in the *improved* QSIM algorithm, as it will be called in the rest of this text:

1) Whenever a new landmark is discovered for parameter P, augment the CV lists of constraints in which P participates to reflect this new information.

2) Whenever a new state is created, add the parameter magnitudes appearing in all non-DERIV constraints as tuples to the CV lists of these constraints, regardless of whether any of the magnitudes are landmarks or not.

3) During the CV consistency filtering phase, the tuples of proposed magnitudes will be eliminated unless they have the following relationships with all of their constraints' CV tuples:

i. For MINUS and M- constraints, if the proposed magnitude tuple is (m_A, m_B) and the CV tuple is (p, q), the signs of (m_A - p) and (m_B - q) should be the negatives of each other.

ii. For M+ constraints, if the proposed magnitude tuple is (m_A, m_B) and the CV tuple is (p, q), the signs of (m_A - p) and (m_B - q) should be the same.

iii. For ADD and MULT constraints, equations 1 and 2 should be satisfied, respectively.

iv. When a subtraction, or , (in the case of MULT,) division operation required for the above controls has an ambiguous result, the predicate in question will be automatically satisfied for that CV tuple.

4) Qualitative subtraction will yield one of +, 0, or -, according to the ordinal relation of its operands. However, when this relation cannot be determined from the operands, (m_A - p) will yield

+, if both its operands are the same interval magnitude, the parameter has had this same magnitude since the recording of p, and its direction is **inc**,

-, if both its operands are the same interval magnitude, the parameter has had this same magnitude since the recording of *p*, and its direction is **dec**,

?, (the ambiguous value,) otherwise.

5) Qualitative division will yield one of **lto** (<1), **one**, or **gto** (>1), according to the signs and ordinal relation of its operands. However, when this relation cannot be determined from the operands, ($\frac{m_A}{p}$) will yield

gto, if both its operands are the same positive interval magnitude, the parameter has had this same magnitude since the recording of *p*, and its direction is **inc**,

gto, if both its operands are the same negative interval magnitude, the parameter has had this same magnitude since the recording of *p*, and its direction is **dec**,

lto, if both its operands are the same positive interval magnitude, the parameter has had this same magnitude since the recording of *p*, and its direction is **dec**,

lto, if both its operands are the same negative interval magnitude, the parameter has had this same magnitude since the recording of *p*, and its direction is **inc**,

?, (the ambiguous value,) otherwise.

As an example to (1), consider the behavior of BALL_H in Table 2. In (t_1,t_2), the ball is in the interval (blh_1, *0*). This fact is recorded in the CV list of the ADD, as modification (2) requires. At t_2, BALL_H stops at its new landmark, *NewB*. This tells us that the *previous* magnitude of BALL_H can be more correctly described as (blh_1, *NewB*), and the related CV tuple is changed to contain the new information. (Actually, this application of (1) does not contribute to the elimination of the spurious behavior in this example. A system where (1) is necessary to prune the spurious prediction will be presented later in this section.) Some CV tuples containing landmarks adjacent to newly discovered ones cannot be revised according to modification (1); suppose BALL_H had the value (blh_1, *0*) before t_0, note that we can not update this magnitude even after *NewB* 's discovery.

An interval in a CV tuple means that the corresponding parameter had a value in that interval at the time the tuple was recorded, that is, interval CVs contain the same kind of information as pure QSIM's landmarks-only CVs, albeit in a somewhat looser fashion. The tuple of t_1, for instance, simply means that there exists a point X in the interval (-∞, *E1*) such that $X + win_h = blh_1$. The generalization to tuples

with multiple interval values is straightforward, see Section VI for a further discussion.

The rationales for (3.i) and (3.ii) stem from the nature of inverse and direct proportionality, respectively. If M-(A,B), then for every (not necessarily all-landmark) CV tuple (*A_mag*, *B_mag*) of this constraint, A and B's magnitudes should always be at opposing sides of *A_mag* and *B_mag*, (unless they are both exactly "on" them,) because they always "go" in opposing directions. (When *A_mag* and *B_mag* are both landmarks, this is very easy to see, the rules of (4) and (5) allow it to be generalized to interval CVs.) For the same reason, parameters linked by M+s should always be at the same sides of their CVs.

The justification for the rules of (4) and (5) is as follows: For any reasonable function f, if $f'(t) > 0$ for all $t \in [t_{beg}, t_{end})$, then $f(t_1) < f(t_2)$ for all $t_1, t_2 \in [t_{beg}, t_{end})$ such that $t_1 < t_2$. (An analogous proposition holds when $f'(t) < 0$; then $f(t_1) > f(t_2)$.) That is, if a quantity is continuously increasing (decreasing) in an interval, its value at a later time in that interval will be greater (less) than its value at an earlier time in the same interval. Whether the parameter has had the same magnitude in all the time since the recording of the CV to the proposal of the new values can be checked using information in the CV list or the state tree.

Table 4 shows the calculations made by the improved algorithm when the spurious state at t_4 is proposed in the elevator simulation. Each new CV triple is added to the beginning of the list, so older values are used later in the check. As can be seen in the table, Equation 1 fails to be satisfied when the proposed tuple is checked against the CV of t_1. As a result of this, a state containing that combination of magnitudes will not be created, and the spurious behavior of Table 2 will not be predicted.

TABLE 4 CV triples used to test ADD(EL_H,REL_H,BALL_H) at t_4

proposed triple: ((-∞, elh_0),*win_h*, (blh_1, *NewB*))

CV triple	Equation 1	O.K.?
((-∞, elh_0), (*win_h*, *NewR*), (blh_1, *NewB*))	(-) + (-) = (-)	Yes
((-∞, elh_0), *NewR*, (blh_1, *NewB*))	(-) + (-) = (-)	Yes
((-∞, elh_0), (*win_h*, *NewR*), (blh_1, *NewB*))	(-) + (-) = (-)	Yes
((-∞, elh_0), (*win_h*, *NewR*), *NewB*)	(-) + (-) = (-)	Yes
((-∞, elh_0), (*win_h*, *NewR*), (blh_1, *NewB*))	(-) + (-) = (?)	Yes
((-∞, elh_0), *win_h*, blh_1)	(-) + (0) = (+)	NO

As promised above, we now present a spurious prediction which cannot be eliminated without modification (1). Consider a (decidedly very simple) model, consisting of three parameters A, B, and C, with a single constraint among them: ADD(A,B,C). Assume that the parameters have the quantity spaces shown in Table 5 at the beginning of the simulation. Table 6 is a spurious prediction of QSIM about this system. As can be seen, a new landmark named *NewB* is inserted between *b1* and *b2* in B's quantity space at t_2.

TABLE 5 Initial quantity spaces of A, B and C

PARAMETER	QUANTITY SPACE
A	{-∞, *0*, *a1*, *a2*, *a3*, ∞}
B	{-∞, *0*, *b1*, *b2*, ∞}
C	{-∞, *0*, *c1*, *c2*, *c3*, ∞}

TABLE 6 A spurious prediction

A	B	C	time
<*a3*, dec>	<*b2*, dec>	<*c3*, dec>	t_0
<(*a2*, *a3*), dec>	<(*b1*, *b2*), dec>	<(*c2*, *c3*), dec>	(t_0,t_1)
<*a2*, dec>	<(*b1*, *b2*), dec>	<*c2*, dec>	t_1
<(*a1*, *a2*), dec>	<(*b1*, *b2*), dec>	<(*c1*, *c2*), dec>	(t_1,t_2)
<(*a1*, *a2*), dec>	<*NewB*, std>	<(*c1*, *c2*), dec>	t_2
<(*a1*, *a2*), dec>	<(*b1*, *NewB*), dec>	<(*c1*, *c2*), dec>	(t_2,t_3)
<*a1*, std>	<(*b1*, *NewB*), dec>	<(*c1*, *c2*), dec>	t_3
<(*a1*, *a2*), inc>	<(*b1*, *NewB*), dec>	<(*c1*, *c2*), dec>	(t_3,t_4)
<(*a1*, *a2*), inc>	<(*b1*, *NewB*), dec>	<*c1*, std>	t_4
<(*a1*, *a2*), inc>	<(*b1*, *NewB*), dec>	<(*c1*, *c2*), inc>	(t_4,t_5)
<(*a1*, *a2*), inc>	<(*b1*, *NewB*), dec>	<*c2*, inc>	t_5

The last state at t_5 is spurious. To detect this, the information at t_1 must be considered together

with that of t_5. At t_5, the A magnitude is less than at t_1. The C magnitudes are the same. So B at t_5 has to be greater than B at t_1 for the ADD to hold. At first sight, it seems we cannot decide on the relative ordering of ($b1$, $NewB$) and ($b1$, $b2$), and the CV check will be satisfied as it is in ambiguous situations, but the fact that B stops at $NewB$ at time point t_2 after decreasing in ($b1$, $b2$) in the interval (t_0,t_2) tells us that B was actually in ($NewB$, $b2$) in that time period. Now it is clear that the tuple of t_5 is inconsistent. Without modification (1), the algorithm cannot detect this inconsistency.

VI. CORRECTNESS AND COMPLEXITY

To prove that improved QSIM is actually better and not worse than pure QSIM, we have to show that:

i) Improved QSIM does not predict some of the spurious behaviors that pure QSIM predicts,

ii) Improved QSIM does not predict any spurious behaviors that pure QSIM does not predict, and,

iii) Improved QSIM is sound, i.e., it does not fail to predict any actual behavior.

(i) has already been proven by demonstration in the previous section. It is also easy to show (ii), i.e., improved QSIM causes no extra spurious behaviors: Consider the changes that we made to the algorithm. They cause it to record and check more CVs than before. CV tuples which contain only landmark values (i.e., the only kind present in the pure version,) are handled in the same way as before, so all the spurious behaviors to whose eliminations they contribute in pure QSIM are also eliminated in improved QSIM. The only difference that improved QSIM has is that it checks proposed states against interval corresponding values; a control totally absent in pure QSIM. If no inconsistencies are detected in this extra checking, improved QSIM will give the same output as pure QSIM. Otherwise, it will predict less behaviors. In no case can it predict some behavior that pure QSIM does not predict. So (ii) has been proven.

To prove (iii), we have to show that the filtering criteria given in the previous section eliminate only those tuples which do not satisfy their constraints, i.e., that the improved filters are *conservative* [4].

For the ADD and MULT constraints, this is already proven, since we use Kuipers' equations (1 and 2.)

For the M+, M-, and MINUS constraints, our filtering criteria subsume those of Kuipers, and are still conservative. Consider M+(B,A), the arguments for M- and MINUS are similar. We will give a geometric interpretation of the rules. When intervals are allowed in CV tuples, each tuple defines either a

point, a segment, or a rectangle in the plot of the two parameters against each other. It is known that the plot of A vs. B passes through these CV "areas." Each proposed magnitude tuple also defines such an area, call it PA. For M+ to hold, some point in PA has to be "higher" than some other points in all the CV areas to the "left" of PA, and "lower" than some points in all the CV areas to the "right" of PA. That is because we know that the plot will be "rising" as one goes from left to right in the graph (whichever way you look at it; A vs. B or B vs. A.) Our requirement that both proposed magnitudes should be "on the same sides" of the values in all CVs embodies this necessity: *Both* coordinates of each point in the graph of an M+ function are either less or greater than the coordinates of other points. The filter eliminates tuples only when it is certain that they do not fulfill this condition. So no magnitudes which truly satisfy the M+ are eliminated. For M- and MINUS, the reasoning is similar, with the slope of the curve now negative. In all cases, only values which violate the constraints are filtered out, so the filters are conservative.

Having shown the correctness of the improvement, let us consider the complexity of the resulting algorithm. If c is the number of constraints and t is the current length of the behavior being generated, constraint filtering takes $O(ct)$ time in pure QSIM [1]. In the improved version, the task of determining whether there is an unbroken string consisting of the same magnitude between a proposed magnitude and its recorded counterpart can be done in parallel with the previously present scanning of the CV list. This means that, although the improved version is using more CVs, the time required for using them is still on the order of ct. Updating the CVs after landmark discovery takes time linear in c. The worst-case time requirement of the algorithm is, again, caused by successor generation, and is exponential in the number of parameters. The modification can be implemented with almost no extra memory requirement, since QSIM already keeps the path in the state tree leading from the root to the current node for cycle detection purposes, and all the necessary CV information exists in that sequence of states.

We ran our implementations of both versions of the algorithm on various input systems; famous examples from literature, and those of our own. In cases where it is able to prune additional behaviors, improved QSIM also shows a clear time and space advantage, naturally. In one case (reported in [6],) the modified algorithm runs 19% faster than pure QSIM. When restricted to produce the tree only until t_4, pure QSIM predicts 147 behaviors of the elevator/ball system, four of which are spurious ones that are eliminated by the improved version. Systems for which no extra elimination is observed take, on the average, 3% more time to simulate. For further details, consult [6].

VII. CONCLUSION

We have identified a source of possible spurious behaviors in the original QSIM algorithm. This class of behaviors result from the fact that QSIM does not record magnitude tuples as corresponding values unless all members of the tuple are landmarks. This causes the algorithm to forgo future consistency checks which could detect the spurious behavior. Our solution is to record all parameter value combinations of constraints in the system's history as CVs. This also requires modifying other parts of the algorithm to perform a slightly more complex kind of qualitative arithmetic. The utility of the method has been demonstrated on example models. The modifications do not change the overall time complexity.

Many well-known cases of spurious behaviors (for example, those of the spring system [1]) are not caused by the feature of QSIM discussed here. Therefore, they cannot be eliminated by our modification. The methods mentioned in Section I have been successful to a large extent for a wide area of systems. Used in conjunction with the modified algorithm, they would provide an even tighter simulation and a corresponding increase in the level of complexity of systems that can be handled by QSIM. The fact that our modification does not necessitate any changes in the input and output formats or requirements of QSIM means that the new algorithm can be used to replace QSIM easily in any integrated application, leading to more correct results from the final product.

REFERENCES

[1] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, pp. 289-338, 1986.

[2] B. Kuipers, "Qualitative reasoning: Modeling and simulation with incomplete knowledge," *Automatica* , vol. 25, pp. 571-585, 1989.

[3] B. Kuipers and C. Chiu, "Taming intractible branching in qualitative simulation," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, California, pp. 1079-1085, 1987.

[4] B. Kuipers, C. Chiu, D. T. Dalle Molle and D. Throop, *Higher-order derivative constraints in qualitative simulation*. University of Texas at Austin, Artificial Intelligence Laboratory, AI TR 89-117, October 1989.

[5] W. W. Lee and B. J. Kuipers, "Non-intersection of trajectories in qualitative phase space: A global

constraint for qualitative simulation," *Proceedings of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann, Los Altos, California, pp. 286-291, 1988.

[6] A. C. C. Say, *Qualitative System Identification.* Ph. D. Dissertation, Bogaziçi University, Computer Engineering Department, Istanbul, 1992.

FOOTNOTES

Affiliation of authors: Computer Engineering Department, Bogaziçi University, 80815 Bebek, Istanbul, Turkey

[1] M+ and M- are the *qualitative function constraints*; they are used when there is a monotonically increasing (in the case of M+) or decreasing (M-) function relating the two parameters involved.