Using Inter-Behavior Contradictions for Modeling and Diagnosis

A. C. Cem Say

Department of Computer Engineering
Boğaziçi University
Bebek 80815, İstanbul, Turkey
say@boun.edu.tr

Abstract

We present a technique for determining certain pairs of qualitative simulation predictions to be mutually contradictory. This leads the simulator to produce more informative outputs, which results in improved performance in reasoning tasks like model revision and diagnosis.

Introduction

Qualitative simulation programs, which enable the representation and use of incomplete knowledge, provide useful tools for analysis, design, and diagnosis of dynamic systems. (Kuipers 1994) The utility of such a program depends directly on the extent and correctness of its predictions about the considered system. We present a technique for making qualitative simulators give more precise information about the interrelationships among the predicted system behaviors. Our technique exploits the fact that alternative behavior pairs can sometimes be automatically proven to be mutually contradictory, meaning that they can not be exhibited by the same physical system. This knowledge can be used to improve performance in qualitative reasoning tasks like diagnosis and modeling.

Preliminaries

The qualitative simulator used in our work is a version of QSIM (Kuipers 1994) enhanced with the generalized corresponding value (Say and Kuru 1993) feature. A brief overview is given below.

QSIM's input is a system model and an initial state written in the qualitative format. The variables' values and certain relationships in the model are represented incompletely, usually corresponding to infinitely many real numbers and "quantitative" functions, respectively.

The "important" values that a variable may take are represented by *landmark* symbols in an ordered list (the *quantity space*). *Magnitudes* are points or intervals in the quantity space. The *direction* of a variable is the sign of its time derivative; i.e. – (*dec*), 0 (*std*), or + (*inc*).

The *state* of a system is the collection of the magnitudes and directions of its variables.

The system model consists of the quantity spaces and the *constraints* that represent the relationships among the variables. The examples in this paper contain the constraint types:

`(d/dt x y)` ≡ $x(t) = y(t)$

`(M+ x y)` ≡ $y(t) = f(x(t)), f' > 0$

`(constant x)` ≡ $x(t) = 0$

The constraints may possess associated tuples of *corresponding values* (CV's). A CV tuple is a collection of the magnitudes that can be taken on by the constrained variables at the same time. For example, the constraint

`(M+ liquid-amount pressure-at-bottom)`

describing a liquid tank would have the CV tuple <0,0>, since an empty tank has zero liquid pressure. Magnitudes appearing in CV tuples can be intervals as well as points. (Say and Kuru 1993)

During simulation, candidates for the next system state which do not satisfy the constraints are filtered out. Due to the incompleteness of the available knowledge, QSIM may predict several successor states for a given system state, building a tree of states as its output. The initial state is the root of this *behavior tree*. The paths from the root to the leaves are the *behaviors*. All mathematically possible behaviors are predicted. (Kuipers 1994)

New landmarks are introduced during simulation by inserting a new symbol to the quantity space when a variable "stops" while it has an interval magnitude. Similarly, new CV tuples are linked to the constraints as the sets of constrained variables are seen to have new collections of values as the simulation proceeds. So, in general, states which are nearer to the leaves of the tree have more CV tuples associated with them than states which are nearer to the root.

Identifying Contradictory Behaviors

The Idea

Consider the simple model M1 consisting of the constraints

```
((constant v))
((d/dt x1 v))
((M+ x1 x2) (inf inf) (minf minf))
```

where both *x1* and *x2* have the quantity space $\{-\infty, 0, \infty\}$, and *v* has a negative landmark called *v0*. Let the initial state be

(v = <*v0*, std>, $x1$ = <(0, ∞), dec>, $x2$ = <(0, ∞), dec>).

One system with this model is shown in Fig. 1: There is a long road built in the north-south direction on a desert. The position of an object on the road is measured

relative to a specific "zero point," north being the positive direction. We have a long wall on the eastern side of the road. At some point to the west of the road, a very powerful light source illuminates the whole area. A car is going south with constant speed. *x1* is its displacement and *v* is its velocity. *x2* is the displacement of the shadow of the car on the wall from the wall's zero point, which is at the same latitude as the road's zero point. We are not interested in what happens after the car or the shadow reaches the zero point.

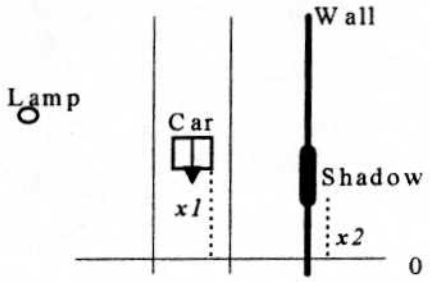

Figure 1: One possible car-shadow system (corresponding to prediction B3)

This simulation produces the predictions

B1: The car and its shadow reach zero simultaneously (meaning that the light source is at the latitude of the zero points.) The CV tuple <0, 0> has been appended to the M+ constraint.

B2: The car reaches the zero point before the shadow (meaning that the light source is to the south of the zero point.) The tuple <0, (0, ∞)> has been appended to the M+ constraint.

B3: The shadow reaches the zero point before the car (meaning that the light source is to the north of the zero point.) The tuple <(0, ∞), 0> has been appended to the M+ constraint.

As described in (Kuipers 1994), if QSIM makes *n* different predictions, namely B1, B2, ..., B*n*, given a model M and an initial state Qstate(t_0), this can be viewed as the proof of the theorem

$$M \wedge Qstate(t_0) \rightarrow (B1 \vee B2 \vee ... \vee Bn). \qquad (1)$$

But we can state more than a simple disjunction about the relation among the three behaviors of the car-shadow system, in the following sense: If we see that one actual "run" of this system from this initial qualitative state produces, say, B1, we can be certain that all other runs will also produce B1. This is because each behavior predicted by QSIM in this simulation entails a different assumption about a system property which is not given in the input model, namely the latitude of the light source relative to the zero point. The three leaves of the behavior tree correspond to three *different* specializations of the input model. If we have evidence that the actual system model is of the kind that produces one behavior, we can legitimately erase the others from our list of predictions for future runs. The alternative predictions here are *contradictory* to each other. In this sense, the theorem that can be proven for this particular system is

$$M1 \wedge Qstate(t_0) \rightarrow (B1 \vee B2 \vee B3) \wedge (B1 \rightarrow \overline{B2}) \wedge (B1 \rightarrow \overline{B3}) \wedge (B2 \rightarrow \overline{B3}).$$

(It should be noted that some QSIM users employ M+ constraints to stand for time-dependent interactions between the two variables. Such an interpretation would violate the "standard" assumption we make in the previous paragraph about the existence of a unique and time-invariant underlying function, and the stronger conclusion about the behavior interrelationships would not be derivable.)

Even when the simulation does not involve different model specializations in different branches, predictions may contradict each other. Consider the modified system M2:

```
((constant v))
((d/dt x1 v))
((d/dt x2 v))
```

The road now has two lanes, with one car on each lane. The cars go south with the same speed. *x1* and *x2* are the positions of those cars. (Fig. 2) The initial state is now

(v = <$v0$, std>, *x1* = <*x1init*, dec>, *x2* = <*x2init*,dec>), where *x1init* and *x2init* are positive landmarks.

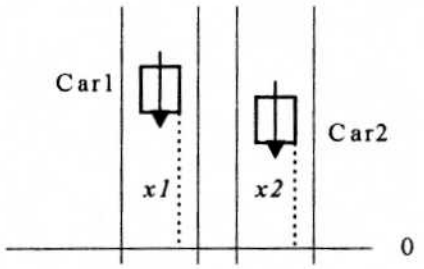

Figure 2: One possible realization of the initial state of the two-cars system

In this simulation, QSIM predicts three behaviors, describing the cases where Car1 reaches the zero point before, after, or at the same time as Car2. No branch of the tree contains any new CV's. So the model at each leaf is exactly the same as the model at the root. But it is easy to see in this case that each particular behavior rules out *all* the others when observed. Two behaviors with different orderings of the cars can not be exhibited in two different runs starting from this same initial state, which is an *exact* description of the world under consideration, without any intervals providing "slack." (Note that we are making the "standard" assumption that each landmark symbol in the input represents a single specific, albeit usually unknown, real number.) So QSIM can prove

$$M2 \wedge Qstate(t_0) \rightarrow (B1 \vee B2 \vee B3) \wedge (B1 \rightarrow \overline{B2}) \wedge (B1 \rightarrow \overline{B3}) \wedge (B2 \rightarrow \overline{B3}).$$

for this simulation.

If, on the other hand, the initial state of the two-cars system were

(v = <$v0$, std>, *x1* = <(0, ∞), dec>, *x2* = <(0, ∞), dec>), we could only say

$$M2 \wedge Qstate(t_0) \rightarrow (B1 \vee B2 \vee B3)$$

about this system, since all three possible orderings of the cars correspond to this qualitative state, and observing, say, Car1 "win" in a particular behavior would not justify us to conclude that it will always win when the system is started up in another "quantitative" state corresponding to $Qstate(t_0)$.

The Algorithm

Our method (Fig. 3) is based on the ideas introduced in the previous section: Checking the initial state to see if it is an exact description, and checking the list of CV tuples accumulated at each leaf of the behavior tree for consistency with the corresponding CV tuples of all the other leaves.

A qualitative state is *exact* if it can only match a single point in phase space. (Note that, in general, we would not be able to determine which particular point that is, given the information available to the reasoner.) A variable appearing in the first place of a `d/dt` constraint is a *phase variable*. The algebraic manipulator used in the exact state identification routine scans the entire constraint set and attempts to show that every variable in the model is either:

a) A phase variable,
b) A constant, or,
c) Another function that can be defined solely in terms of the phase variables.

If the model allows the derivation of such expressions for each variable, then an assignment of point values to the phase variables would completely determine the system state, since all other magnitudes and derivatives of any order are functions of the magnitudes of the phase variables. The value `true` is assigned to the Boolean variable `InitialStateIsExact` only when this expression derivation succeeds and all phase variables are seen to be at point magnitudes.

The Boolean function `CrossCheckOK` uses the QSIM algorithm's constraint filters to check each CV tuple of one behavior against all the tuples of that constraint in the other behavior. If an inconsistency is found, these two behaviors represent different specializations of the input model and can never be exhibited by the same physical system. (Note that CV inconsistencies can be detected on "precise" constraints like `add` or `mult`, (Kuipers 1994) as well as the inherently incomplete `M` functions. The difference between the specializations deemed to be contradictory in such cases is in the assumptions they entail about the relative magnitudes of the numbers represented by the landmarks in the CV's. (Say 1998)) The quantity spaces have to be passed as arguments to `CrossCheckOK` because all of the QSIM CV consistency control routines involve operations which compare two values of the same variable, and such comparisons can only be performed if the variable's quantity space is known. Since states belonging to different branches can have different sets of landmarks for the same variable, `CrossCheckOK`'s first job is to produce "unified" quantity spaces for each variable appearing in the constraint under consideration. New landmarks which cannot be placed unambiguously in the unified quantity space (this happens when both behaviors have discovered different landmarks lying in the same interval of the initial quantity space,) are replaced by the narrowest interval which contains them and which is bounded by nonproblematic landmarks before being passed to the comparison routine. For instance, assume that two predictions Bi and Bj have modified the initial quantity space $\{0, \infty\}$ of variable x to

$\{0, new\_x\_1, new\_x\_2, \infty\}$

and

$\{0, new\_x\_1, new\_x\_3, \infty\}$

respectively. When these behaviors are being cross-checked, the incomparable landmarks are eliminated to obtain the unified quantity space $\{0, new\_x\_1, \infty\}$. Any appearance of, say, *new_x_2* in a CV tuple will be replaced by (new_x_1, ∞) during the subsequent controls.

```
if InitialStateIsExact
 then Mark all prediction pairs as contradictory
 else
  begin
   (* The QSIM tree contains n behaviors *)
   for  i = 1  to  n-1
    for  j = i + 1  to  n
     for each constraint C∉{d/dt,constant} in the system model
      begin
       CL1 = C's CV list for the final state of Behavior i;
       CL2 = C's CV list for the final state of Behavior j;
       LL1 = the set of final quantity spaces for Behavior i;
       LL2 = the set of final quantity spaces for Behavior j;
       if not(CrossCheckOK(CL1,CL2,LL1,LL2))
        then Mark Behaviors i and j as contradictory; Jump to next j
      end
  end;
Use the contradiction marks to write the prediction formula
```

Figure 3: The contradiction detection algorithm

The contradiction detection algorithm outputs a Boolean expression (the *prediction formula*) describing the discovered interrelationships among the behaviors. See (Say 1998) for examples on the detailed execution of both stages of the algorithm.

Discussion

Correctness. Contradiction recognition based on exact initial state identification is a *sound* technique (i.e. the behavior pairs that it marks are indeed contradictory,) since, by our definition, an exact qualitative state corresponds to a single point in phase space, determining the future system behavior completely. (A rigorous proof would also include a demonstration of the correctness of the algebraic manipulator used to derive expressions for the non-phase variables.) Since this algebraic manipulator's capabilities are limited, this routine may "miss" some exact states by failing to come up with the necessary expressions for one or more variables, even when this could be achieved through a more complicated series of rewrites. Therefore, some contradictions may go unnoticed by this routine, meaning that it is an *incomplete* procedure.

The CV cross-checking technique, on the other hand, is both sound and complete: Two behaviors are marked as contradictory only if they contain CV tuples which fail to satisfy a QSIM constraint. (See (Kuipers 1994) for the correctness of the QSIM constraint filters.) No such CV contradiction can be missed, because of the exhaustive nature of the algorithm of Fig. 3.

Application Style and Complexity. Although we implemented the procedure of Fig. 3 as a postprocessor to QSIM, it can just as well be applied in an incremental manner to the *partial* behavior tree after each recording of a new CV tuple by a newly created state during simulation. (In that case, the algorithm has to be modified to skip behavior pairs whose prefixes have already been marked.) It is easy to find individual examples in which either of these methods outperforms the other one from the point of view of execution time, this being a matter of the tree's shape and CV structure.

Contradiction detection's computational complexity is quadratic in the number of leaves in the tree. Cross-checking a behavior pair takes $O(c.d^2)$ time, where c is the number of constraints, and d is the depth of the tree, which is the worst-case number of CV tuples.

Intractable Branching and Contradiction Detection. One common complaint about qualitative simulators in general, and "landmark generation" approaches like QSIM in particular, is that they generate "too many" predictions. Since the contradiction finder makes use of CV tuples discovered during simulation, one may be led to think that it is fundamentally a landmark generation technique, and intractable branching could overwhelm any gains that would be expected from it as the problem size is increased. Fortunately, this is not the case: It is possible to record and use CV tuples even when the simulator is not able to generate any new landmarks. In fact, no behavior contradiction in any of the examples in this paper involves a CV tuple containing a newly generated landmark. We claim that our approach is a positive contribution because it utilizes previously unused information in the simulation environment and improves the predictive accuracy.

We examined how the number of discovered contradictions (and hence the "use" to which they can be put, as will be explained in the next section) grows in relation to the number of behaviors by incrementally increasing the number of landmarks causing branching in the ball-shadow system of Fig. 4 and running the program. (This simulation branches depending on the order in which landmarks in the quantity spaces of the ball and shadow position variables are crossed. See (Say 1998) for the details.) As Table 1 indicates, the *percentage* of the prediction pairs marked as contradictory actually increases as the tree becomes bigger: The method seems to be more productive when there happens to be more branching.

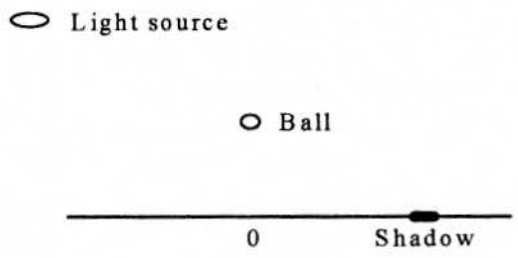

Figure 4: The ball-shadow system

Applicability

The contradiction identification applications proposed below only make use of the CV cross-checking technique. Since *exact* matching of a collection of observed real values to a previously recorded one is impossible in practice, the identification and use of exact qualitative states is not included in the following discussion.

System "Complexity" (Number of nonzero landmarks in input)	Number of Predictions	Number of Contradictions	Percentage of Pairs Marked as Contradictory
"Basic" system with one landmark each	11	32	58
An extra landmark for one variable	23	176	70
An extra landmark for each variable	61	1536	84

TABLE 1. Behavior and Contradiction Counts in Successively More Detailed Runs of the Ball-Shadow System

BEHAVIOR # 1

c(Na,P)	*c(ADH,P)*	time
cn eq,dec	*ca eq*,dec	t_0
(*cn**, *cn eq*),dec	(*ca**,*ca eq*),dec	(t_0, t_1)
*cn**,dec	(*ca**,*ca eq*),dec	t_1
(0,*cn**),dec	(*ca**,*ca eq*),dec	(t_1, t_2)
(0,*cn**),dec	*ca**,dec	t_2
(0,*cn**),dec	(0,*ca**),dec	(t_2, t_3)
new cn 1,std	*new ca 1*,std	t_3

BEHAVIOR # 2

c(Na,P)	*c(ADH,P)*	time
cn eq,dec	*ca eq*,dec	t_0
(*cn**,*cn eq*),dec	(*ca**,*ca eq*),dec	(t_0, t_1)
(*cn**,*cn eq*),dec	*ca**,dec	t_1
(*cn**,*cn eq*),dec	(0,*ca**),dec	(t_1, t_2)
*cn**,dec	(0,*ca**),dec	t_2
(0,*cn**),dec	(0,*ca**),dec	(t_2, t_3)
new cn 2,std	*new ca 2*,std	t_3

Figure 5: Two behaviors of the hypothalamus-pituitary gland subsystem during the onset of SIADH

Diagnosis with Multiple Experiments

Kuipers (1994) describes how the contrapositive of the "theorem" proven by QSIM can be used to refute candidate system models in a diagnosis application: (1) can be transformed to

$$\text{Qstate}(t_0) \wedge \overline{\text{B1}} \wedge \overline{\text{B2}} \wedge \ldots \wedge \overline{\text{B}n} \rightarrow \overline{\text{M}}. \qquad (2)$$

So we conclude that M is not a correct model of our system if an observed behavior starting with Qstate(t_0) does not match any of the predictions B1 through B*n*. If each observed trajectory of the system is matched with at least one prediction, M cannot be overruled, since the antecedent of (2) is false in this case.

The additional information contained in the prediction formulae written by the contradiction finder can be used for improved diagnostic reasoning: We know that each contradictory behavior pair <Bi,Bj> embodies a statement of the form

$$(\text{M} \wedge \text{Bi} \rightarrow \overline{\text{Bj}}).$$

When we rearrange this implication, we obtain

$$(\text{Bi} \wedge \text{Bj} \rightarrow \overline{\text{M}}). \qquad (3)$$

So if we make two separate observations Obs_1 and Obs_2 of two behaviors of the same system at different time intervals, and if these observed trajectories uniquely match the distinct predictions Bi and Bj from the QSIM output for this system, *and* if Bi and Bj are shown to be contradictory by the postprocessor, then we can soundly deduce that either our observation record or the presumed model for this system is wrong. Note that the original QSIM-based diagnosis framework is unable to recognize a faulty model when it observes two or more trajectories, each matching a prediction. The contradictory QSIM predictions under consideration do not even have to be the simulation results of the same initial state in our approach, since Qstate(t_0) does not appear in the implication that we use.

We will now give an example for this kind of diagnosis. A model of the healthy water intake/excretion balance mechanism in the human body (see (Kuipers 1987) for the details) contains the constraint

`(M+ c(Na,P) c(ADH,P))`,

representing the hypothalamus' function of sensing the concentration of sodium in the blood plasma, and "commanding" the pituitary gland to secrete a commensurate amount of antidiuretic hormone (ADH). The disorder known as the Syndrome of Inappropriate Secretion of AntiDiuretic Hormone (SIADH) gradually disrupts this `M+` constraint, eventually resulting in a constant value for *c(ADH,P)*. Assume that, in addition to their healthy equilibrium values, the variables *c(Na,P)* and *c(ADH,P)* have one extra landmark each, named *cn** and *ca** respectively. QSIM predicts (among others) the two behaviors shown in Fig. 5 for the healthy system's response to suddenly increased water intake (e.g. a transfusion). The variables decrease and simultaneously settle on new landmarks in both behaviors. The only difference from the point of view of the hypothalamus-pituitary gland subsystem between these behaviors is the order in which the landmarks *cn** and *ca** are crossed.

This behavior pair is contradictory with the constraint `(M+ c(Na,P) c(ADH,P))`, as would be noticed by a cross-checking of their generalized CV's. If we make two observations of this system, responding to separate transfusions, which uniquely match these two behaviors respectively, we can drop the healthy model and consider the fault models in our library.

Note how we stress that each observed trajectory has to be matched to a single prediction for diagnostic reasoning to be valid. Consider the behavior Bi appearing in (3). If a single observation matches both Bi and another prediction Bk, (because of incomplete information provided by the observation procedure,) we only know that (Bi $\vee$ Bk) is true, and there is insufficient information for the application of (3).

Modeling

Model revision is the task of modifying a qualitative model in response to undesirable features of its simulation output. This issue is becoming increasingly important as qualitative simulators are being used for bigger problems, where the complexity of the behavior tree may be a negative factor for many users. Learning how to prepare "good" models is not very easy, but there are simulation tools that provide help for model revision. As Clancy, Brajnik, and Kay (1997) point out, one way of decreasing the ambiguity of a simulation is to incorporate CV tuples in an appropriate manner in the model. Contradictory behavior information can be used to intelligently determine *where* a new correspondence is necessary: Intuitively, each contradiction found by the `CrossCheckOK` routine indicates that a constraint in

the system model lacked some CV tuples whose addition would eliminate some behaviors from the simulation output. Since no system which can exhibit two contradictory behaviors can exist, it is meaningful to try to enrich the model until no such contradictory pairs remain in the simulation output. The contradiction finder can therefore be viewed as a modeling aid that suggests which parts of the model under construction need to be improved. (The program specifies the constraint that caused the contradiction for each pair in its output.)

The qualitative modeling scenario would then look like this: The modeler prepares a QSIM model of the observed (or planned) system according to her initial understanding of its mechanism. She then runs QSIM (augmented with the contradiction detection postprocessor) with this model once for each distinct initial state that matches the observations (or specifications.) Examining the behavior tree and the contradiction information, the modeler can incorporate the appropriate CV tuples that would eliminate some of the contradictory behaviors to the model if she has justification to select one alternative over another. If the CV tuples are missing because of a genuine lack of information about the system, new observations or "experiments" aimed at discovering a more precise description of the relationship among the variables involved in the problematic constraint can be arranged. Such observations may even lead to the refutation and removal of the constraint under consideration, as discussed earlier.

Qualitative system identification (Say and Kuru 1996) is one way of automating the model construction task. Qualitative system identification programs take as input a list of possible qualitative behaviors of the system under consideration, and present a constraint model for the system. The capability of checking two separate behaviors for consistency, given a model, has interesting implications for these programs as well. All state-of-the-art qualitative system identification programs (see the list in (Say and Kuru 1996)) fail to recognize that the system exhibiting the behaviors of Fig. 5 can *not* have the constraint `(M+ c(Na,P) c(ADH,P))` in its model. A contradiction detector working on the combined input/output sets of such a program would notice the CV inconsistency when it attempted to cross-check these two behaviors.

Related Work and Conclusion

Clancy, Brajnik, and Kay (1997) present a set of tools and methods to help users of qualitative simulation perform model revision. Their techniques include using temporal logic trajectory constraints for restricting the simulation to a subset of behaviors having a desired property, focusing the view on a specified subset of the system variables, and analyzing why certain branches occur in the produced tree. Contradiction detection fits nicely in this framework, pointing at correspondences that could have been "turned on" in the initial model.

Falkenhainer and Forbus' (1991) use of *compositional modeling* allows their program to intelligently turn on or off certain fragments of the model to efficiently give relevant responses to queries. Our technique works at the much lower level of individual CV's and simply checks the exhaustive list of model specializations produced by the simulator against each other.

Our discussion of diagnosis presumed the existence of a routine for processing sensor data to observe trajectories and matching them to the predictions. This *measurement interpretation* task is an important problem by itself; see (DeCoste 1991) for a detailed account.

Dvorak's (1992) MIMIC program for monitoring dynamic systems uses template (2) to refute a hypothesized system model when observation of a single trajectory fails to match any of that model's predictions. The originality of our approach to using the prediction formula for diagnosis is that it utilizes contradictions between *several* observations to refute the model.

Shults and Kuipers (1997) describe how QSIM trees can be queried about high-level behavioral features using temporal logic. The extension of their language for accommodating queries about contradictory behaviors of various specializations of the model is on our agenda.

Having been implemented in the QSIM framework, the contradiction finder is easily integrable with the recent technical advances in qualitative simulation, (e.g. the complete solution of the problem of chatter (Clancy and Kuipers 1997)) improving the power and applicability of the overall reasoner.

The address say@boun.edu.tr can be contacted to obtain source codes of our programs described here.

Acknowledgments

I thank Ivan Bratko and the anonymous referees for their helpful comments and suggestions. This work was partially supported by the Boğaziçi University Research Fund. (Grant no: 99A101)

References

Clancy, D. J.; Brajnik, G.; and Kay, H. 1997. Model Revision: Techniques and Tools for Analyzing Simulation Results and Revising Qualitative Models. In *Proc. Eleventh Int. Workshop on Qualitative Reasoning*, Cortona, Italy. 53-65.

Clancy, D. J., and Kuipers, B. J. 1997. Static and Dynamic Abstraction Solves the Problem of Chatter in Qualitative Simulation, In *Proc. Fourteenth Nat'l Conf. Artificial Intelligence*.

DeCoste, D. 1991. Dynamic Across-Time Measurement Interpretation. *Artificial Intelligence* 51:273-341.

Dvorak, D. L. 1992. Monitoring and Diagnosis of Continuous Dynamic Systems Using Semiquantitative Simulation. Ph.D. diss., University of Texas at Austin.

Falkenhainer, B. and Forbus, K. 1991. Compositional Modeling: Finding the Right Model for the Job. *Artificial Intelligence* 51:95-143.

Kuipers, B. J. 1987. Qualitative Simulation as Causal Explanation. *IEEE Trans. Systems, Man, and Cybernetics* 17:432-444.

Kuipers, B. J. 1994. *Qualitative Reasoning: Modeling*

and Simulation with Incomplete Knowledge. Cambridge, Mass.: The MIT Press.

Say, A. C. C. 1998. Identifying and Using Contradictory Behaviors in Qualitative Simulation., Technical Report, CMPE-98-QR1, AI Lab., Boğaziçi University.

Say, A. C. C., and Kuru, S. 1993. Improved Filtering for the QSIM Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(9):967-971.

Say, A. C. C., and Kuru, S. 1996. Qualitative System Identification: Deriving Structure from Behavior. *Artificial Intelligence* 83:75-141.

Shults, B. and Kuipers, B. 1997. Proving Properties of Continuous Systems: Qualitative Simulation and Temporal Logic. *Artificial Intelligence* 92:91-129.