Numbers Representable in Pure QSIM

A. C. Cem Say

Department of Computer Engineering, Boğaziçi University, Bebek 80815, İstanbul, Turkey

say@boun.edu.tr

Abstract

The standard number representation for qualitative reasoning is shown to be more powerful than commonly thought. We prove that this scheme, based on ordered lists of symbols for critical values and *corresponding value* tuples fixing points on the curves relating variables, is sufficient for representing the equality of a landmark to any specified number in the radical extension of the set of rational numbers, as well as the numbers π and e. This suggests that the perceived "weakness" of qualitative reasoners stems not necessarily from their representational setup, but from the nature of the algorithms they employ on their input models.

Introduction

The *quantity space* representation for numbers is a standard feature of qualitative reasoning [8,2] programs. This representation, in which the value of a particular quantity is shown either as a landmark (a symbol standing for an interesting threshold) or an interval between two consecutive landmarks, is commonly regarded [1] as a quite weak (i.e. abstract) one. The only obvious things one can say about a value in the quantity space framework are its sign and its ordinal relationships with other points or intervals in the same quantity space. One also employs *corresponding values* (tuples of landmarks satisfying a known relation between system variables) to represent additional knowledge about the curves of such relations, and, in the same time, the relative magnitudes of the landmarks involved. The ability of qualitative reasoners to deal with systems for which only this kind of incomplete information is available is considered to be one of their strong points, so this perceived degree of abstraction is suitable in most cases. However, the qualitative reasoning community has recently come under criticism [5] for the supposed inadequacy of a typical example of their programs in performing "expert reasoning," and this problem has been attributed to the employed "modeling language, behavior representation, and algorithm." Furthermore, certain applications require number representations which allow somewhat more exact statements and deductions to be made about the values of the variables in the system under consideration. In some cases, one possesses more knowledge about the numerical values of some individual landmarks, or the ratios of two landmarks of the same variable, and would like these facts to be taken into account by the reasoner. In other cases, one might want to inspect the output of a qualitative reasoner to see whether new information of this sort about the landmark values has been produced. Several "extended" schemes, which allow the representation of these kinds of information by using "mixed" qualitative-quantitative approaches, have been devised. [3,4,11,2]

In the rest of this paper, we show that a well-known and widely used quantity space- and corresponding value-based representation, namely that of Kuipers' original QSIM [2] algorithm, is powerful enough to represent the equality of a landmark to any specified rational number, as well as elements of a sizable subset of irrational numbers.

Table 1. The qualitative constraint types

`(add X Y Z)`	$\equiv X(t) + Y(t) = Z(t)$
`(mult X Y Z)`	$\equiv X(t) \,.\, Y(t) = Z(t)$
`(minus X Y)`	$\equiv Y(t) = -X(t)$
`(d/dt X Y)`	$\equiv \dfrac{d}{dt}X(t) = Y(t)$
`(constant X)`	$\equiv \dfrac{d}{dt}X(t) = 0$
`(M+ X Y)`	$\equiv Y(t) = f(X(t)), f' > 0$
`(M- X Y)`	$\equiv Y(t) = f(X(t)), f' < 0$

For a detailed explanation of QSIM, see [2], which is the standard reference. In the following, we use only the "pure QSIM" constraints shown in Table 1. "Multivariate" constraints are not considered.

Numbers in QSIM

In this section, we will show that information at a great level of detail about the numerical values of landmarks is expressible in QSIM. We start by showing that landmark equality across quantity spaces is expressible, and end up by proving that one can represent the equality of a landmark to any specified number within an important subset of **R**. As will be seen, most of the proofs in this section are in an algorithmic form, telling one what components (variables and constraints) to add to an already existing QSIM model, so that a specific item of information will be "embedded" in the final, augmented model. (In the exposition below, equations like $x = y$ for two landmarks x and y are supposed to mean that the real numbers represented by these two symbols are equal; such equations do not assert that the two symbols are one and the same.)

Proposition 1. Let x and y be landmarks in the quantity spaces of variables X and Y, respectively. The information $x = y$ is expressible in QSIM.

Proof. Define a new variable P with the basic quantity space $\{-\infty, 0, \infty\}$. Insert the constraint `(add X P Y)` with the corresponding value (CV) tuple (x,0,y) to all operating regions. Since this is equivalent to saying that $x + 0 = y$, $x = y$ has been shown.

Proposition 2. Let x and y be landmarks in the quantity spaces of variables X and Y, respectively. The information $x = -y$ is expressible in QSIM.

Proof. Define a new variable P with the basic quantity space $\{-\infty, 0, \infty\}$. Insert the constraint `(add X Y P)` with the CV tuple (x,y,0) to all operating regions. Since this is equivalent to saying that $x + y = 0$, $x = -y$ has been shown.

These very simple facts are the building blocks used to represent a variety of relations between two landmarks in the same quantity space.

Proposition 3. Let x_1 and x_2 be landmarks of a variable X. The information $x_1 = -x_2$ is expressible in QSIM.

Proof. Define a new variable P with the quantity space $\{-\infty, 0, p, \infty\}$. Assume, without loss of generality, that, of x_1 and x_2, x_1 is the negative one. Use the method of Proposition 2 to express $p = -x_1$. Now, if we can say that x_2 is equal to p, we will have expressed $x_1 = -x_2$. But Proposition 1 already established that this is possible, so the fact that two landmarks in the same quantity space are negatives of each other is expressible in pure QSIM's notation.

Proposition 4. Let x_1 and x_2 be landmarks of a variable X. The information $x_2 = 2*x_1$ is expressible in QSIM.

Proof. We will show how to express this fact for the case where x_1 and x_2 are positive. The method used when they are negative is trivially similar. Define a new variable P with quantity space $\{-\infty, 0, p, \infty\}$. Insert the constraint `(add P P X)` with the CV tuple (p,p,x_2) to all operating regions. This is equivalent to saying that $p + p = x_2$. Now, using the method of Proposition 1, we can say that $p = x_1$, which yields $2*x_1 = x_2$.

This means that there is an easy way of expressing $x_2 = k*x_1$ for cases where $\mathbf{k} = 2^i$, where i is any positive integer. For the more general case where k is any integer, several approaches are possible, including the following one.

Proposition 5. Let x_1 and x_2 be landmarks of a variable X. For any given integer k, where

$|k|>1$, the information $x_2 = k * x_1$ is expressible in QSIM.

Proof. We show how to express this for the case where x_1 and x_2 are positive. Extension to other cases is straightforward, making use of an additional variable and the method of Proposition 2. We start by defining three new variables named P, S, and U. In addition to the basic set, the quantity space of P includes k positive landmarks named $p_1, p_2, ..., p_k$, where the ordering is so that, of two landmarks, the one with the greater subscript is greater. The quantity space of U similarly includes k positive landmarks named $u_1, u_2, ..., u_k$, where the ordering rule is the same as above. S has the quantity space $\{-\infty, 0, s_1, \infty\}$. One first uses Proposition 1 to express that $p_1 = s_1 = u_1$. Then we insert the constraint `(add P S U)` to all operating regions. The CV list of this constraint is formed by executing the following loop:

FOR c:=2 TO k DO
  BEGIN
    add the CV tuples $(p_c,0,u_c)$ and (p_{c-1},s_1,u_c)
      to the CV list
    (* i.e. establish that $p_c = u_c = c*u_1$. *)
  END

At this point, we have expressed the fact that $p_k = u_k = k*u_1$. (Note that, by construction, any two landmarks whose subscripts are consecutive integers are one "unit" away from each other in every quantity space. Furthermore, this unit is the same for all the variables and is equal to the magnitude of the landmark with subscript 1.) It remains to use Proposition 1 to say $u_1 = x_1$ and $u_k = x_2$.

Once we have the power of representing integer multiples, it is easy to express that the ratio of two landmarks is a given rational number.

Proposition 6. Let x_1 and x_2 be landmarks of a variable X. For any given rational number q, such that $q = \frac{n}{d}$, where $|n|$ and $|d|$ are the smallest integers which satisfy this equation, and neither $|n|$ nor $|d|$ are equal to 1, the information $x_2 = q * x_1$ is expressible in QSIM.

Proof. We show how to express this for the case where $0 < x_1 < x_2$, the methods for the other cases make simple use of additional variables, Proposition 2, and the technique that will be described here. Define a variable **P** with quantity space $\{-\infty, 0, p_1, p_d, p_n, \infty\}$. (Note that $0 < x_1 < x_2$ and $x_2 = \frac{n}{d} * x_1$ together imply that $d < n$.) Use the method of Proposition 5 to express $p_n = n * p_1$ and $p_d = d * p_1$. Clearly, $p_n = q * p_d$ has been expressed. It remains to use Proposition 1 to say $p_d = x_1$ and $p_n = x_2$.

Having come this far making use of only the `add` constraint, we now turn to the `mult` constraint. It is easy to see that one can express equations of the form $x_2 = \sqrt[n]{x_1}$ for integer n using "cascades" of `mult`'s. But this constraint also allows us to express the equality of a landmark to the number 1, which is essential if we want to express information about the real numerical values of individual landmarks, rather than the ratios of two landmarks, as we have done until now. We first provide a definition to make the following statements more concise.

Definition 1. A real number r is said to be *representable in QSIM* if a QSIM model (finite set of quantity spaces, constraints and corresponding values) containing a landmark **x**, from which x's equality to r can be deduced, can be constructed.

Proposition 7. All rational numbers are representable in QSIM.

Proof. We will provide a method for expressing the equality of any given rational number to a landmark. Call the rational number in consideration q. Define a variable named X. If $q = 0$, the standard landmark 0 is by definition equal to it, and the proof is over. Otherwise, introduce a landmark x which has the same sign as the number q into X's quantity space. Define new variables P and S with quantity spaces $\{-\infty, 0, p, \infty\}$ and $\{-\infty, 0, s, \infty\}$, respectively. Express the fact that $p = s$ using the method of Proposition 1. Define a new variable U. In addition to the basic set, U will have a positive landmark called "1". Add the constraint `(mult U P S)` to all operating regions. This constraint will have the CV tuple ("1", p, s). Obviously, the numerical value of "1" has been expressed to be 1. If $q = 1$, express the fact that $x =$ "1", and the proof is over. If $q \neq 1$, the variable U will have another landmark called "q". U's landmarks 0, "1" and "q" will have the same relative ordering as the numbers 0, 1, and q. Using the appropriate one of Propositions 3, 5, or 6, express the fact that "q"= q*"1" = q, and finish by expressing x = "q".

In order to specify the power of our representation precisely, it will be convenient to state some "closure" properties:

Proposition 8. If r_1 and r_2 are two given real numbers representable in QSIM, their i) sum, ii) difference, iii) product, and iv) quotient (if $r_2 \neq 0$) are also representable in QSIM.

Proof. Assume that landmarks x and y of variables X and Y have been shown to be equal to r_1 and r_2, respectively. For each case, define another variable P. Note that, since r_1 and r_2 are known, the signs of their sum, difference, etc. are unambiguously known. For each case, let P have a landmark p with the appropriate sign. Do the following depending on the operation to be applied:

i) To represent $r_1 + r_2$, define the constraint `(add X Y P)` with CV tuple (x,y,p),

ii) To represent $r_1 - r_2$, define the constraint `(add P Y X)` with CV tuple (p,y,x),

iii)To represent $r_1 * r_2$, define the constraint `(mult X Y P)` with CV tuple (x,y,p),

iv)To represent $\frac{r_1}{r_2}$, define the constraint `(mult P Y X)` with CV tuple (p,y,x).

We now expand our attention to a new "operator," namely, $\sqrt[n]{}$

Proposition 9. If r is a given real number representable in QSIM, then the number $\sqrt[n]{r}$ (n a positive integer) is also representable in QSIM if it is real.

Proof. First, some well-known facts of algebra: Each nonzero real number r has n distinct complex n^{th} roots, at most two of which can be real. All positive real numbers have two real n^{th} roots differing only in sign for even n, and a single positive real n^{th} root for odd n. Negative real numbers have no real n^{th} roots for even n, and a single negative real n^{th} root for odd n. For odd n and positive r, $\sqrt[n]{-r} = -\sqrt[n]{r}$. So we can prove this proposition for positive r and with the understanding that the $\sqrt[n]{}$ "operator" always yields a positive value without any loss of generality; the method of Proposition 2 can be used to establish equality to the number's negative, if needed. Define the new variable S_1 with quantity space $\{-\infty, 0, s_1, \infty\}$, ($s_1$ will be shown equal to $\sqrt[n]{r}$) and execute the following loop:

```
FOR c:=2 to n DO
  BEGIN
    Define new variable Sc with quantity space
        {−∞, 0, sc, ∞};
    Define the constraint (mult Sc-1 S1 Sc)
        with CV tuple (sc-1,s1,sc)
  END;
```

By construction, landmark s_n of variable S_n is equal to $s_1{}^n$. Now express equality of s_n to r to complete the chain.

Propositions 7 to 9 can be reformulated together as follows:

Proposition 10. Any real number in the radical extension of the set of rational numbers is representable in QSIM.

In other words, any real number for which we can write a finite expression consisting only of integers, parentheses, and the symbols +, -, *, /, and $\sqrt[n]{\ }$, where n is an integer, is representable in QSIM.

Let us now examine the problem of representing transcendental numbers. The `d/dt` and `M+` constraints, which have been absent from our discussion so far, will be useful in this regard.

Proposition 11. The number *e* is representable in QSIM.

Proof. Consider the variables X, Y, DXT, and DYT. X has the quantity space {−∞, 0, "1", "e", ∞}. Y has the quantity space {−∞, 0, "1", ∞}. Assume that the model has been augmented with additional variables and constraints so that both the landmarks named "1" have been expressed to be equal to 1. Assume further that the constraints

`(d/dt Y DYT)`,

`(d/dt X DXT)`,

`(mult X DYT DXT)` and

`(M+ X Y)` with CV tuples (0, −∞), ("1", 0), ("e", "1"), (∞, ∞)

have been asserted as well. From this combination of constraints, "e"'s equality to *e* can be deduced. The `mult` constraint, when rearranged, simply means that

$$\frac{dY/dt}{dX/dt} = \frac{1}{X}$$

Since, by the definition of the `M+` constraint, Y is a differentiable function of X, the chain rule for the derivative of a composite function allows us to replace the left hand side by dY/dX:

$$\frac{dY}{dX} = \frac{1}{X}$$

Integrating, we get

$$Y = \ln X + c,$$

where c is a constant.

The corresponding values enable us to determine that c is zero, so

$$Y = \ln X.$$

So "e" is the number whose natural logarithm is 1, namely *e*.

The method we used in this proof suggests that surprisingly detailed information about the shape of the monotonic functions can be represented in pure QSIM as well; see [6] for a detailed discussion. In particular, it is easy to see that this ability to represent the natural logarithm function and its inverse enables us to establish a new "closure" rule, which states that any number of the form x^y is representable in QSIM whenever both x and y are representable in QSIM and x is positive.

Proposition 12. The number π is representable in QSIM.

Proof. Consider the variables X, Y, U, DXT, DYT, X2, and P. Y has the quantity space {−∞, "-pi/2", 0, "pi/2", "pi", ∞}. Assume that "-pi/2" has been shown to be -"pi/2" by the method of Proposition 3, and "pi" has been shown to be 2*"pi/2" by the method of Proposition 4. U is fixed at its positive landmark "1", which has been expressed to be equal to 1. The constraints

`(d/dt Y DYT)`,

`(d/dt X DXT)`,

`(mult X X X2)`,

`(add X2 U P)`,

`(mult P DYT DXT)`, and

`(M+ X Y)` with CV tuples (−∞, "-pi/2"), (0,0), (∞, "pi/2")

have been asserted. From this model, proceeding as we did in the previous proof, we obtain

$$Y = \arctan X.$$

Since $\lim_{X\to\infty} \arctan X = \pi/2$, the value of "pi" has been expressed to be π.

Clearly, making use of the "closure" rules mentioned earlier, we can obtain an immensely rich set of numbers, including transcendental as well as algebraic ones, which are representable in QSIM.

Note that all the algorithms in this section were designed so that the variables and constraints that they add do not further constrain the original system model. This issue is best explained by an example. Consider Proposition 2. A naive way of expressing $x = -y$ for landmarks of variables X and Y would be to assert simply the constraint `(minus X Y)` with CV tuple (x,y). This seems more economical than the method we used, since it does not involve the creation of a new variable. (In QSIM, it is desirable to have as few variables as possible for efficiency reasons.) But this approach is wrong, since `(minus X Y)` carries a much stronger statement than $x = -y$, it means (see Table 1) that at all times t, $X(t) = -Y(t)$. The addition of such a constraint *changes* the system model; specifically, it could prune some or even all of the behaviors that QSIM would have predicted for the original model. The algorithms presented here contain additional, otherwise unconstrained variables to avoid such situations. They also refrain from adding landmarks into quantity spaces about which quantitative information is being expressed, providing a neat separation of "real" model items and those created by the quantitative information incorporator.

Discussion

In [1], Forbus presents a scale of "abstraction" for the various ways of representing numbers used in the field of qualitative reasoning, and puts quantity spaces "high in this structure, almost up to sign values." In this paper, we have shown that quantity spaces, when used in conjunction with corresponding values, provide a representation which is actually very strong in the sense that one can express the equality of a landmark value to any specified number in an important subset of the real numbers employing the vocabulary of this representation only.

Table 2. A spurious QSIM behavior prefix

time	t_0	(t_0, t_1)	t_1
P	0, inc	$(0,p_1)$, inc	p_1, std
U	0, inc	$(0,u_1)$, inc	$(0,u_1)$, inc
S	0, std	$(0,s_1)$, inc	$(0,s_1)$, inc
Z	0, dec	$(-\infty,0)$, dec	$(-\infty,0)$, dec
R	0, inc	$(0,r_1)$, inc	r_1, std
X	0, inc	$(0,x_1)$, inc	$(0,x_1)$, inc
Y	0, std	$(0,y_1)$, inc	$(0,y_1)$, inc
Z2	0, dec	$(-\infty,0)$, dec	$(-\infty,0)$, dec
D	0, inc	$(0,d_1)$, inc	d_1, std

Although it may possess knowledge represented in this manner, pure QSIM does not make use of such embedded quantitative information for pruning spurious behaviors which may contain "quantitative inconsistencies." This is because, although the QSIM *representation* is strong enough for this job, the qualitative arithmetic routines of the pure QSIM *algorithm* are not adequate for using such data to help disambiguate certain operations which lead to branchings in the state tree. (In fact, as Wellman [9] points out, the filter QSIM employs for checking the consistency of `(add X Y Z)` constraints does not even distinguish between

$$Z(t) = X(t) + Y(t),$$

that is, the relation that it is supposed to enforce, and the weaker

$$Z(t) = g(X(t), Y(t)),$$

where g is a continuous and differentiable function, and both $\partial g/\partial X$ and $\partial g/\partial Y$ are positive throughout g's domain.)

The idea presented in this paper can thus easily be used to define a class of spurious behaviors; namely, those which can be generated by QSIM but contain inconsistent items of embedded quantitative information. One example of such a spurious behavior is exhibited by the following model and presented in Table 2: (All variables except Z and Z2 have a single positive finite landmark.)

`(mult P U S)` with CV tuple (p_1,u_1,s_1),

`(add U Z S)` with CV tuple $(u_1,0,s_1)$, `(mult R X Y)` with CV tuple (r_1,x_1,y_1),

`(add X Z2 Y)` with CV tuple $(x_1,0,y_1)$,

`(add P D R)`.

The state at t_1 is spurious, since the multiplication and addition constraints' corresponding values can be used to show that

$$p_1 = \frac{s_1}{u_1} = \frac{u_1 + 0}{u_1} = 1 = \frac{x_1 + 0}{x_1} = \frac{y_1}{x_1} = r_1,$$

and thus

$$D(t_1) = R(t_1) - P(t_1) = r_1 - p_1 = 0 \neq d_1,$$

so there is an inconsistency.

Modifying the algorithm to make use of information embedded *à la* Propositions 1-12 to handle such problems is a nontrivial task, and the resulting program would not necessarily be better in terms of efficiency or conceptual clarity than hybrid reasoners like Q2 [3] and Q3 (see [2]). In particular, the new variables, which have to be in the model only to establish arithmetic relationships among landmarks, and which do not "mean" anything from the point of view of the system being modeled, would cause "chattering" and other problems. See [7] for a discussion of these and proposals for tackling some of them.

One may also consider writing an "extraction" algorithm to inspect all the quantity spaces and CV lists for a model to see if any quantitative knowledge item can be found embedded in them. Unfortunately, this task requires much more than just inverting the methods of the previous section, since there are an infinite number of ways to express information of each form described in that section. Consider the following two constraints:

`(add A B C)` with CV tuple $(a_1,b,0)$

`(add A D B)` with CV tuple $(a_2,0,b)$

Here, it is easy to recognize the construct used in the proof of Proposition 3, and one can deduce that $a_1 = -a_2$. However, the same information $a_1 = -a_2$ is embedded in

`(mult A A E)` with CV tuples (a_1,a_1,e), (a_2,a_2,e)

in a different manner. Furthermore, it is easy to see that, using only, say, the transitivity of = or the properties of negation, one can construct arbitrarily large sets of constraints involving huge numbers of variables which imply the same piece of information about a_1 and a_2 in such a manner that no proper subset implies it. Therefore, an algorithm which would simply try to match constructs in the output with those mentioned in the proofs of the previous section could possibly miss some quantitative information deducible from the output. Clearly, a powerful qualitative algebraic reasoner with capabilities similar to those of [10,11] has to be incorporated into our algorithm if we wish to squeeze the most from a given constraint-quantity space set.

One promising direction for future work is the integration of such a quantitative information extraction scheme with a method of quantitative propagation like that of [3]. This could improve the overall power of semi-quantitative reasoners like Q2 by allowing them to utilize any available embedded items of numerical information, as well as those provided explicitly by the user.

Acknowledgements

I thank the anonymous reviewers for their helpful comments. This work was supported by the Boğaziçi University Research Fund. (Grant no: 96HA0121)

References

[1] K. D. Forbus, "Qualitative physics: Past, present and future," in: D. S. Weld and J. de Kleer, eds., Readings in Qualitative Reasoning About Physical Systems. Los Altos, CA: Morgan Kaufmann, 1990.

[2] B. J. Kuipers, *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge* (MIT Press, Cambridge, MA, 1994).

[3] B. Kuipers and D. Berleant, "Using incomplete quantitative knowledge in qualitative reasoning," in: Proceedings AAAI-88, Saint Paul, Minn., 1988, pp. 324-329.

[4] B. Kuipers and D. Berleant, *A smooth integration of incomplete quantitative knowledge into qualitative simulation*. Artificial Intelligence Laboratory, The University of Texas at Austin, AI90-122, 1990.

[5] E. P. Sacks and J. Doyle, "Prolegomena to any future qualitative physics," Computational Intelligence, vol. 8, pp. 187-209, 1992.

[6] A. C. C. Say, "Functions representable in pure QSIM," Proceedings of the Fifth Turkish Symposium on Artificial Intelligence and Neural Networks, İstanbul, 1996, pp. 251-255.

[7] A. C. C. Say and M. Doyuran, "The representative power of quantity spaces and correspondences," in: Proceedings of the Third Turkish Symposium on Artificial Intelligence and Neural Networks, Ankara, Turkey, 1994, pp. 205-214.

[8] D. S. Weld and J. de Kleer, eds., Readings in Qualitative Reasoning About Physical Systems. Los Altos, CA: Morgan Kaufmann, 1990.

[9] M. P. Wellman, "Qualitative simulation with multivariate constraints," in: Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, 1991, pp. 547-557.

[10] B. C. Williams, "MINIMA: A symbolic approach to qualitative algebraic reasoning," in: Proceedings AAAI-88, Saint Paul, Minn., 1988, pp. 264-270.

[11] B. C. Williams, "A theory of interactions: unifying qualitative and quantitative algebraic reasoning," Artif. Intell. vol. 51, pp. 39-94, 1991.