"COMMONSENSE" ARITHMETIC REASONING

Reid Simmons

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

"Arithmetic reasoning" can range in complexity from simple integer arithmetic to powerful symbolic algebraic reasoning of the sort done by MACSYMA. We describe an arithmetic reasoning system of intermediate complexity called the *Quantity Lattice*. In a computationally efficient manner the Quantity Lattice integrates qualitative and quantitative reasoning, and combines inequality reasoning with reasoning about simple arithmetic expressions, such as addition or multiplication. The system has proven useful in doing simulation and analysis in several domains, including geology and semiconductor fabrication, by supporting useful forms of reasoning about time and the changes that happen when processes occur.

1 Introduction

"Cogito Ergo Sum" — I think, therefore I add

"Arithmetic reasoning" denotes a broad class of inferences which range in complexity from simple integer arithmetic, such as "$1+1=2$," to complex symbolic algebra, such as "$\int x(2\log x+1)dx = x^2\log x + C$." We have identified a particular class of arithmetic reasoning which we believe to be very common in tasks such as simulation, planning and diagnosis. This class of arithmetic reasoning is intermediate in both expressive power and computational efficiency. This paper describes an implemented system called the *Quantity Lattice* and details its expressive power and potential range of applications. We also indicate where the Quantity Lattice fits in the spectrum of arithmetic reasoning tools.

Certain classes of arithmetic inferences seem to crop up frequently in everyday life. Some involve solely qualitative relationships :

- $A < B$ and $B \leq C$. Is $A < C$?
- Joe is taller than Amy and Jack is shorter than Amy. Is Joe taller than Jack?

Others involve mixed qualitative and quantitative information :

- $X \leq 1$ and $Y = 2$. What is the relationship between X and Y?
- New York is less than 120 miles away and Washington is 138 miles away. Which city is closer?

A large class of simple arithmetic reasoning problems combine qualitative relationships with arithmetic expressions:

- $X \leq 1$ and $Y = 2$. What is the relationship between $X+X$ and Y?
- I finish class at 3, then eat for at most 1 hour, and afterwards study for 2 to 3 hours, but have to be at a meeting by 6. Is there enough time to fit in a half hour nap?
- A geologic formation is eroded all the way down to sea level. Uplift follows. Is it possible for airborne erosion to affect that formation again?
- Two silicon wafers are oxidized at the same rate for the same amount of time. They are then etched for the same amount of time, but one wafer etches faster than the other. Which wafer will be thicker at the end?

These questions, and many more like them, can all be answered by reasoning about ordinal relationships ($>, <, =, \geq, \leq \neq$) between expressions and by reasoning about the value of simple arithmetic expressions ($+, *, -, /$). To handle cases where the values of the numbers are only partially specified, the reasoning must also be able to combine qualitative and quantitative information. There are several systems reported in the AI literature which handle various subsets of this class of inferences, including those which deal with time [Allen,Dean,Vere], space [Davis] and actions [Forbus,Simmons].

The Quantity Lattice is an arithmetic reasoning system which handles a wider class of arithmetic inferences than the systems referred to above. The remainder of this paper describes the Quantity Lattice and details the inferences it supports, discusses why the Quantity Lattice is useful and compares it with other arithmetic reasoning systems.

2 The Quantity Lattice

The primary significance of the Quantity Lattice[1] is that it smoothly integrates *relationships, arithmetic expressions, qualitative* and *quantitative* information, permitting it to handle a wide range of "commonsense" arithmetic inferences. By "integrates" we mean that adding one type of knowledge may constrain other types and thereby enable additional inferences to be made. For example, if we tell the system that "$Y = X + 5$" it will infer the additional *qualitative* constraint "$Y > X$," even though it does not yet know anything about the actual values of X or Y. If we now tell the system that "$X < 2$" it will deduce the additional *quantitative* constraint that "$Y < 7$."

The Quantity Lattice has been used for reasoning about time

[1] The name "Quantity Lattice" is historical and does not imply that the representation is a mathematical lattice.

and about the effects of processes [Mohammed,Simmons,Williams]. In temporal reasoning it has been used to maintain a consistent partial order of time points and to answer queries about relationships between time points and about the durations of intervals. Its main application, however, has been in reasoning about the effects of processes. Consider, for example, the geologic model which states "the height of a formation after uplift equals the height before uplift plus the amount of uplift" (from [Simmons]). If the only numeric information known is that the amount of uplift is positive, the Quantity Lattice can still infer that the height after uplift is strictly greater than the height before uplift. If we later tell the system that the height before uplift is at least 100 and the amount of uplift is at least 50, it can then infer that the height after uplift is at least 150. The Quantity Lattice supports such inferences in a computationally efficient manner.

An obvious question is "why implement an arithmetic reasoning system when existing symbolic algebra packages like MACSYMA can perform the same class of inferences and more?" The main answer is efficiency. The Quantity Lattice is designed to efficiently handle problems in which there are thousands of variables, expressions and inequalities, but where each expression contains only a small fraction of the total number of variables. The algorithms and data structures used by the Quantity Lattice are designed to take advantage of this type of arithmetic problem which is often encountered in doing commonsense reasoning such as in the geology or semiconductor manufacture domains.

Another major advantage of the Quantity Lattice is that it maintains justifications for all its inferences. This dependency information facilitates doing retraction and is used to generate explanations of how two quantities are related. The system described in [Mohammed] to diagnose failures in semiconductor fabrication depends in large part on the explanations generated by the Quantity Lattice to determine how attributes of the wafer relate to parameters of the manufacturing process.

2.1 Representation

The Quantity Lattice supports reasoning about ordinal relationships between expressions whose values are real numbers. An ordinal relationship is one of $>, <, =, \geq, \leq, \neq$. An expression is a simple expression, such as "A," a numeric expression, such as "5," or an arithmetic expression, such as "$B + 5$."

Expressions are represented as nodes in a digraph. The nodes are called *quantities* and the arcs of the graph are called *relationships*. An arithmetic expression is simply a quantity with an associated *formula*, a list of its operator and arguments.

Information is added to the Quantity Lattice by asserting or retracting relationships between expressions, such as "$A = B + 5$." The system constrains the value of an expression by reasoning about its position in the graph and, if it is an arithmetic expression, by the values of its arguments. It uses the assertions to infer relationships between expressions and to infer upper and lower numeric bounds on the values of expressions.

The upper and lower numeric bounds are represented by associating a real valued interval with each quantity. The interval indicates that the actual value of the expression falls somewhere within the interval range. For example, if the only constraint on **A** is that it is positive, **A** would have the interval $(0, \infty]$, denoted by $\mathbf{A} \in (0, \infty]$.[2]

[2]A parenthesis indicates a half-open interval, a bracket indicates a half-closed interval.

As a simple example, the two equations "$A = B + 5$" and "$B \geq 0$" are represented by five quantities : **A**, **B**, **5**, **0** and **(B + 5)**. The quantities **A** and **(B + 5)** are linked by an "=" arc in the graph and **B** and **0** are linked by a "$\geq$" arc. The quantity **5** has the interval $[5, 5]$ and the quantity **0** has the interval $[0, 0]$.

2.2 Inferences

Two types of inferences are performed by the Quantity Lattice :
(i) determining the relationship between two quantities,
(ii) constraining the value of an arithmetic expression.
These types of inferences are carried out by using five different reasoning techniques :

1. Determining relationships using graph search
2. Determining relationships using numeric constraint propagation
3. Constraining the value of arithmetic expressions using interval arithmetic
4. Constraining the value of arithmetic expressions using relational arithmetic
5. Constraining the value of arithmetic expressions using constant elimination arithmetic

These reasoning techniques are integrated in the sense that inferences performed by one technique can be used by another to perform further inferences. For example, the relational arithmetic technique infers ordinal relationships between an arithmetic expression and its arguments. These relationships can be used by the graph search technique to find new relationships between quantities.

2.2.1 Graph Search

There are two ways for the system to determine the relationship between the quantities **A** and **B** — one qualitative and one quantitative. The qualitative technique searches the graph of quantities using a simple breadth-first search to find a path between the quantities. Figure 2 presents a small graph in which we are trying to find the relationship between **A** and **B**. Each quantity is marked by the order in which it is searched and by its relationship to **A**. Relationships are found by using a simple transitivity table (see Figure 1). For example, since $A = C$ and $C \leq E$ we can infer that $A \leq E$ by finding the intersection of the column marked = and the row marked $\leq$ in the transitivity table. Notice that the search along the bottom branch does not proceed past **G** because its relationship to **A** is unknown.

The standard breadth-first search will find any path between

Figure 1: Transitivity Table for Ordinal Relationships

	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$
$<$	$<$	$<$	??	??	$<$	??
$\leq$	$<$	$\leq$	??	??	$\leq$	??
$>$	??	??	$>$	$>$	$>$	??
$\geq$	??	??	$>$	$\geq$	$\geq$	??
$=$	$<$	$\leq$	$>$	$\geq$	$=$	$\neq$
$\neq$	??	??	??	??	$\neq$	??

?? means that the relationship is unknown.

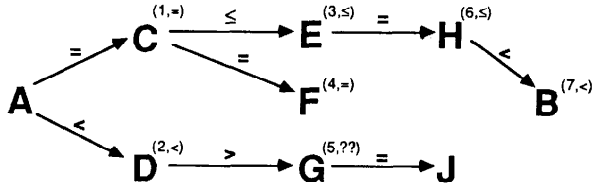

Figure 2: Graph Search of the Quantity Lattice

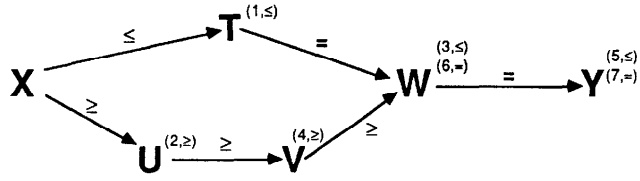

Figure 3: Combining Paths to Constrain the Relationship

the two quantities. However, in some cases there are multiple paths between two quantities with different paths yielding different relationships. Since we want to find the most constrained relationship between the quantities (where $<, >$, and $=$ are more constrained than $\leq, \geq$, and $\neq$) we need to modify the search slightly to combine the different relationships found via different paths.

For example, Figure 3 presents a graph in which we are trying to find the relationship between **X** and **Y**. The quantities are again marked with the order of the search and relationship found so far. Notice that **W** and **Y** are visited twice, and that the second time they are visited the relationship recorded on the quantity is the combination of the relationships found via the multiple paths. For instance, following the path **X**, **T**, **W** the relationship is $\leq$ but following the path **X**, **U**, **V**, **W** the relationship is $\geq$. The combination of $\leq$ and $\geq$ yields $=$, which is the most constrained relationship between **X** and **W**. Thus the most constrained relationship between **X** and **Y** is also $=$. In general, a quantity is revisited only if the relationships found via separate paths combine to yield a more constrained relationship.

Although this extension to the standard breadth-first search means that some quantities might be visited more than once, since any combination of unconstrained relationships yields a constrained one, they are visited at most twice. Thus the complexity of this algorithm is $O(R)$, where R is the number of relationships (i.e. arcs) in the graph, the same order of complexity as that of standard breadth-first search.

The result of a search is cached by adding a new relationship to the graph. This relationship is justified by a path between the quantities. There may actually be many equally constraining paths, but for efficiency only one is found and recorded as the justification.

2.2.2 Numeric Constraint Propagation

The other method of determining the relationship between two quantities is quantitative. The ordering between two quantities can be determined if the intervals associated with the quantities do not overlap, except possibly at their endpoints,[3] since the value of a quantity is constrained to lie within the interval. For example, if $\mathbf{A} \in (-\infty, 2]$, $\mathbf{B} \in [2, \infty)$ and $\mathbf{C} \in [5, 10]$ then we can infer that $A \leq B$ and $A < C$, but cannot infer anything about the relationship between B and C. In addition, equality can be inferred if both intervals are single points and they have the same value.

This quantitative method for determining relationships between quantities has two advantages over the graph search technique : (i) it is a constant time algorithm, and (ii) it can detect relationships not explicit in the graph, since the system implicitly "knows" the ordering of the reals (e.g. from "$A < 1$" and "$B > 2$," infer "$A < B$").

However, this method alone is not sufficient to answer questions of the form "if $A > 1$, $B < 0$ and $A < C$ what is the relationship between B and C?" because we first need to determine the upper and lower bounds of the intervals of **B** and **C**. This is done by performing a numeric constraint propagation whenever a relationship is asserted between two quantities. This propagation ensures that the intervals of all quantities are consistent with the assertion. For example, if we assert "$A < C$" the system constrains the upper limit of **A**'s interval to be less than the upper limit of **C**'s interval. Similarly, it constrains the lower limit of **C**'s interval to be greater than **A**'s lower limit. In turn, these constraints propagate to all quantities which are $<, \leq$, or $=$ to **A** and $>, \geq$, or $=$ to **C**. This constraint propagation algorithm has the same computational complexity as the graph search algorithm, for reasons similar to those presented in Section 2.2.1.

Both the qualitative and quantitative inference techniques described above perform consistency checking. When the user asserts a relationship between two quantities, the Quantity Lattice checks to see if the relationship is consistent. This involves searching the Quantity Lattice graph to make sure that the inverse relationship cannot be inferred from the relationships already present. Thus, asserting a relationship in the Quantity Lattice is of complexity $O(R)$. When performing numeric constraint propagation, the system checks to ensure that the upper bound of an interval is never less than its lower bound. If an inconsistency is found, an exception is raised which the user must handle. Typically, this entails finding the justifications underlying the inconsistency and retracting one of them.

2.2.3 Interval Arithmetic

One of the important features of the Quantity Lattice is that it combines reasoning about ordinal relationships with reasoning about arithmetic expressions. The Quantity Lattice maintains constraints between the two types of knowledge in order to provide a more expressive system.

As mentioned, an arithmetic expression such as "$B + 5$" is represented as a quantity with an associated formula. An arithmetic expression can be placed in the Quantity Lattice graph like any other quantity by asserting relationships between it and other quantities, such as "$A = B + 5$." Thus, the value of an arithmetic expression may be constrained by the values of other quantities as described in the previous section.

There are three other techniques used by the Quantity Lattice to constrain the value of an arithmetic expression further. One technique is quantitative (*interval arithmetic*) and the other two are qualitative (*relational arithmetic* and *constraint elimination arithmetic*). Interval arithmetic computes the value of an

[3] The implementation actually allows the intervals to overlap by some ϵ to compensate for the approximate nature of computer arithmetic.

$$[xl, xu] + [yl, yu] \equiv [(xl + yl), (xu + yu)] \qquad [xl, xu] * [yl, yu] \equiv \left[\begin{array}{l} \min(xl * yl, xl * yu, xu * yl, xu * yu), \\ \max(xl * yl, xl * yu, xu * yl, xu * yu) \end{array}\right]$$

$$[xl, xu] - [yl, yu] \equiv [(xl - yu), (xu - yl)]$$

$$-[xl, xu] \equiv [-xu, -xl] \qquad [xl, xu]/[yl, yu] \equiv \begin{cases} (-\infty, \infty) & \text{if } (yl < 0) \wedge (yu > 0) \\ \left[\begin{array}{l} \min(xl/yl, xl/yu, xu/yl, xu/yu), \\ \max(xl/yl, xl, yu, xu/yl, xu, yu) \end{array}\right] & \text{otherwise} \end{cases}$$

Figure 4: Interval Arithmetic Operators

arithmetic expression by applying the arithmetic operator of the formula to the endpoints of the intervals of its arguments. For example, "$[3, 6) + [-1, 5]$" yields "$[2, 11)$." The system maintains the most constrained interval by applying interval arithmetic when the arithmetic expression is first constructed and whenever the interval of an argument changes. Also, constraining the arithmetic expression through interval arithmetic may in turn constrain the other quantities related to the arithmetic expression via numeric constraint propagation.

Figure 4 presents some of the operators used by Quantity Lattice in doing interval arithmetic.[4] Although just five basic operations are shown, it is quite easy to add other arithmetic operators. In particular, the trigonometric and absolute value operators were added for the version of the Quantity Lattice used in the geologic reasoner of [Simmons].

As an example of interval arithmetic, consider the following set of constraints :

- $A \geq 3, A \leq 4$, (i.e. $\mathbf{A} \in [3, 4]$)
- $B \geq 1, B \leq 4$, (i.e. $\mathbf{B} \in [1, 4]$)
- $C = 2$, (i.e. $\mathbf{C} \in [2, 2]$)
- $D = (B * C)/(A + B)$

Using interval arithmetic, the system computes that $(\mathbf{B} * \mathbf{C}) \in [2, 8]$ and $(\mathbf{A} + \mathbf{B}) \in [4, 8]$ constraining $\mathbf{D} \in [0.25, 2]$. If we now assert "$B \geq C$" numeric constraint propagation will constrain $\mathbf{B} \in [2, 4]$ (see Section 2.2.2). The system will then recompute the arithmetic expressions, constraining $\mathbf{D} \in [0.5, 1.6]$.

Unlike many constraint propagation systems, for efficiency reasons constraints in the Quantity Lattice are not bi-directional. They have a preferred direction — constraints are propagated up to an arithmetic expression from its arguments. To achieve inferences in the other direction, the user must explicitly assert constraints for each argument of the arithmetic expression in terms of the expression and its other arguments. For example, given the expression "$(A + B)$" one would assert "$B = (A + B) - A$" and "$A = (A + B) - B$."

[4]For presentation purposes, the axioms ignore whether the intervals are open or closed.

2.2.4 Relational Arithmetic

Interval arithmetic has some serious limitations. First, interval arithmetic will often compute intervals which are larger than commonsense dictates. For example, suppose we know that $\mathbf{A} > \mathbf{B}$, $\mathbf{B} \in [0, 1)$ and $\mathbf{A} \in (0, 1]$. Interval arithmetic computes that $(\mathbf{A} - \mathbf{B}) \in (-1, 1]$ but we should be able to infer that $(\mathbf{A} - \mathbf{B}) \in (0, 1]$ since $\mathbf{A}$ is greater than $\mathbf{B}$. The problem is even clearer when we realize that by using interval arithmetic we cannot determine, in general, that $\mathbf{A} - \mathbf{A}$ is zero. For example, if $\mathbf{A} \in [1, 2]$ then by interval arithmetic $(\mathbf{A} - \mathbf{A}) \in [-1, 1]$ since $[1, 2] - [1, 2] = [-1, 1]$. Only by knowing that both intervals refer to *the same* quantity can we infer that the answer is [0,0].

Another limitation is that often interval arithmetic cannot increase our knowledge at all. For example, if all we know is that "$X = Y + 5$," then $\mathbf{Y} \in (-\infty, \infty)$ and by interval arithmetic we can only constrain $\mathbf{X} \in (-\infty, \infty)$. Using interval arithmetic we gain no information about the relationship between $\mathbf{X}$ and $\mathbf{Y}$, although we know, in fact, that $\mathbf{X}$ is greater than $\mathbf{Y}$.

We have compensated for both these deficiencies in interval arithmetic by combining it with an arithmetic technique based on ordering relationships. *Relational arithmetic* maintains constraints on the qualitative relationship of an arithmetic expression to its arguments. The relationship depends on the relationship of the expression or its arguments to the identity value for the arithmetic operator of the expression.

Figure 5 presents axioms encoding this relational arithmetic technique. Using these axioms and the examples presented above, the system infers that since $5 > 0$ then $(\mathbf{Y} + \mathbf{5}) > \mathbf{Y}$ and therefore $\mathbf{X} > \mathbf{Y}$. Similarly, the system infers that since $\mathbf{A} > \mathbf{B}$ then $(\mathbf{A} - \mathbf{B}) > 0$. This inference, combined with numeric constraint propagation, constrains the lower bound of $(\mathbf{A} - \mathbf{B})$ to be greater than 0, while interval arithmetic constrains the upper bound to be less than or equal to 1. Thus integrating the two techniques constrains $(\mathbf{A} - \mathbf{B}) \in (0, 1]$ which is the smallest consistent interval for this problem.

The complexity of the relational arithmetic algorithm is $O(R)$. The algorithm includes three steps : (i) performing several comparisons of quantities to determine which axioms are applicable,

Figure 5: Axioms for Relational Arithmetic

For $rel \in \{<, \leq, >, \geq, =, \neq\}$

$$x \; rel \; 0 \Rightarrow (x + y) \; rel \; y$$
$$y \; rel \; 0 \Rightarrow (x + y) \; rel \; x$$

$$x \; rel \; y \Rightarrow (x - y) \; rel \; 0$$

$$(x > 0 \wedge y > 0) \Rightarrow (x \; rel \; 1 \Rightarrow (x * y) \; rel \; y) \wedge (y \; rel \; 1 \Rightarrow (x * y) \; rel \; x)$$
$$(x > 0 \wedge y < 0) \Rightarrow (x \; rel \; 1 \Rightarrow y \; rel \; (x * y)) \wedge (y \; rel \; -1 \Rightarrow (x * y) \; rel \; -x)$$
$$(x < 0 \wedge y > 0) \Rightarrow (x \; rel \; -1 \Rightarrow (x * y) \; rel \; -y) \wedge (y \; rel \; 1 \Rightarrow x \; rel \; (x * y))$$
$$(x < 0 \wedge y < 0) \Rightarrow (x \; rel \; -1 \Rightarrow -y \; rel \; (x * y)) \wedge (y \; rel \; -1 \Rightarrow -x \; rel \; (x * y))$$

$$(x > 0 \wedge y > 0) \Rightarrow ((x \; rel \; y \Rightarrow (x/y) \; rel \; 1))$$
$$(x > 0 \wedge y < 0) \Rightarrow ((x \; rel \; -y \Rightarrow -1 \; rel \; (x/y)))$$
$$(x < 0 \wedge y > 0) \Rightarrow ((x \; rel \; -y \Rightarrow (x/y) \; rel \; -1))$$
$$(x < 0 \wedge y < 0) \Rightarrow ((x \; rel \; y \Rightarrow 1 \; rel \; (x/y)))$$

$$
\begin{array}{rcl}
\text{For } rel \in \{<, \leq, >, \geq, =, \neq\} & & \\
x \; rel \; y & \Rightarrow & (x + z) \; rel \; (y + z) \\
\\
x \; rel \; y & \Rightarrow & (x - z) \; rel \; (y - z) \\
x \; rel \; y & \Rightarrow & (z - y) \; rel \; (z - x) \\
\\
x \geq 0 \wedge x \; rel \; y & \Rightarrow & (x * z) \; rel \; (y * z) \\
x \leq 0 \wedge x \; rel \; y & \Rightarrow & (y * z) \; rel \; (x * z) \\
\\
x \geq 0 \wedge x \; rel \; y & \Rightarrow & (x/z) \; rel \; (y/z) \\
x \leq 0 \wedge x \; rel \; y & \Rightarrow & (y/z) \; rel \; (x/z)
\end{array}
$$

Figure 6: Axioms for Constant Elimination Arithmetic

(ii) asserting the newly inferred relationship and (iii) performing a numeric constraint propagation. All of these steps are $O(R)$ although if one of the quantities being compared is numeric the comparisons can be done in constant time, such as in the case of the axiom "$x \; rel \; 0 \Rightarrow (x + y) \; rel \; y$."

2.2.5 Constant Elimination Arithmetic

All the above techniques are still not powerful enough to infer that $A > C$ if $A = B + X$, $C = D + X$ and $B > D$. To solve this problem, the system must be able to infer that if the same amount is added to two expressions then the results are related in the same way as the original expressions are. That is, $A \; rel \; B \Rightarrow (A + C) \; rel \; (B + C)$. Figure 6 presents axioms which enable the system to reason about relationships between two arithmetic expressions.[5] We call this *constant elimination arithmetic* because it gives the system the power of a very simple algebraic simplifier — one which can eliminate constants from expressions.

Note that these axioms extend the power of the axioms in Figure 5 which infer relationships involving only one arithmetic expression and one simple expression. In fact, the axioms of Figure 5 are only special cases of the ones in Figure 6. For example, by substituting $Y = 0$ we derive $X \; rel \; 0 \Rightarrow (X + Z) \; rel \; (0 + Z)$ which simplifies to the addition rule in Figure 5. However, for efficiency we have chosen to implement the special cases of Figure 5 separately.

Also, for efficiency, we apply the axioms in Figure 6 in a consequent manner — that is, only for those arithmetic expressions actually in the system. Otherwise we could create an explosion of arithmetic expressions of the form $A + q$ for all quantities q in the Quantity Lattice. Finally, we note that an even more general form of the axioms in Figure 6 are of the form $X \; rel \; Y \Rightarrow (X + Z_i) \; rel \; (Y + Z_j)$ for any $Z_i = Z_j$. We limit the expressive power for the sake of efficiency by insisting that $i = j$.

The algorithm for constant elimination arithmetic is $O(E * R)$, where E is the number of arithmetic expressions in the system. The term R arises because to determine whether the antecedent clause of each axiom is true the Quantity Lattice must be searched to determine the relationship between two quantities. The term E arises because when an expression of the form "$A \; op \; B$" is constructed, the appropriate axiom in Figure 6 must be applied for each existing expression of the form "$A \; op \; C$" or "$C \; op \; B$." In practice, however, the number of such expressions is usually a rather small percentage of E, and so the average complexity is much better than the worst case complexity.

Although the computational complexity of this technique is greater than that of the other four arithmetic reasoning techniques presented above, we included constant elimination arithmetic in the Quantity Lattice because the inferences it supports are often needed in the domains we are exploring. For example, in the semiconductor fabrication domain [Mohammed] we often need to make inferences like "two silicon regions which start out the same thickness will end up the same thickness if they are oxidized at the same rate for the same amount of time."

[5] There are also permutations of the axioms for the commutative operators + and *.

3 Results

Combining the five reasoning techniques of (i) graph search, (ii) numeric constraint propagation, (iii) interval arithmetic, (iv) relational arithmetic and (v) constant elimination arithmetic enables the Quantity Lattice to perform a large range of "commonsense" arithmetic inference in a fairly efficient manner. It can, for instance, handle all the questions listed in the introduction in $O(R)$ time except for the last question which takes $O(E * R)$.

The Quantity Lattice has been tested in several domains, including geology [Simmons], semiconductor fabrication [Mohammed] and reasoning about temporal constraints [Williams]. In both the geology and semiconductor fabrication domains the Quantity Lattice is used to support qualitative simulation — it maintains and reasons about a partial order of time points which represent when processes occur and when objects are created and destroyed, and it helps in reasoning about the changes produced by processes. For example, the effect of the geologic process "uplift" would be represented by equations stating that the height of a formation after the process equals the height before the process plus some positive quantity "uplift-amount." From this information the Quantity Lattice would infer that the new height is greater than the old height.

To measure the performance of the Quantity Lattice, we used an example from the semiconductor fabrication domain and simulated the fabrication of a pair of resistors using the models of [Mohammed]. The simulation involved 27 processing steps and took 384 seconds of CPU time on a Symbolics 3600. Of this, 112 seconds or 29% was used by the Quantity Lattice. The simulator asserted 3125 relationships among 1357 quantities, taking an average of 0.01 seconds per assertion. It constructed 660 arithmetic expressions, of which about one-third were binary additions.

The simulator queried the Quantity Lattice to determine the relationship between quantities over 23,000 times taking an average of 0.003 seconds per query. Of this, the vast majority were for relationships between time points and most were relationships that the system already knew or had already inferred and cached. In fact, of the 23,507 queries 14,736 were already known to the system and were answered taking an average of 0.0005 seconds per query and 751 were determined quantitatively in constant time (see Section 2.2.2) taking an average of 0.002 seconds per query. The remaining 8020 were determined using graph search taking an average of 0.0075 seconds per query. Of the 8020 graph searches, 1884 new relationships (paths) were found between quantities.

In the geologic domain both a qualitative and quantitative simulation are performed [Simmons]. The qualitative simulation is done using the same simulator as in the semiconductor fabrication domain and the performance of the Quantity Lattice is

similar to that described above. For the quantitative simulation, much more emphasis is placed on constructing arithmetic expressions and determining real values for the parameters of processes. Thus the numeric constraint propagation and interval arithmetic techniques are used more heavily than for the qualitative simulation.

Using a 7 step geology simulation example, the quantitative simulation constructed 718 arithmetic expressions, more than was constructed for the 27 step semiconductor fabrication example, while asserting less than half as many relationships as for the semiconductor example. Constructing the arithmetic expressions consumed 45% of the time spent in the Quantity Lattice, as opposed to only 19% for the qualitative simulation. When relationships were asserted between quantities 51% of the time was spent propagating numeric constraints and 29% was spent checking for consistency. These figures are reversed for the qualitative simulation in which 56% of the time was spent doing consistency checking with only 26% needed for constraint propagation.

4 Relation to Other Work

The Quantity Lattice was designed as a compromise between expressive power and computational complexity. Efficiency of operation was gained by taking advantage of the expected structure of the problem — many loosely connected variables and expressions. This is in contrast to a system like MACSYMA which is designed to handle sets of equations where each equation involves most of the variables, that is, the resulting coefficient matrix will be dense rather than sparse. This expectation leads one to use more powerful algebraic techniques like solving systems of equations, which are polynomial in complexity, rather than using the techniques described above which are mostly linear in the number of equations.

There are symbolic algebra algorithms which make use of the structure of the domain to achieve performance comparable to the Quantity Lattice. However, these algorithms are not actually used in MACSYMA because it is designed to solve systems of equations in general. For example, the types of domains handled by the Quantity Lattice are amenable to solution by setting up the equations in band matrix format and representing the matrix of equations as a linked list so that inequalities can be easily inserted into the correct row to preserve the band matrix format.

When one has only a few expressions and inequalities, solving systems of equations as MACSYMA does is not too expensive. When there are thousands of expressions and inequalities, as in our simulation domains, making inferences by symbolically solving equations becomes computationally infeasible. On the other hand, there are many inferences which MACSYMA handles that the Quantity Lattice cannot. For example, the Quantity Lattice does not do simplification. Thus, in general, it cannot deduce that $X = (X + Y) - Y$. The appropriate strategy is to have the problem solving system reason about the class of inferences it needs to make — using the computational efficiency of the Quantity Lattice for simple "commonsense" inferences and doing the more complex (and computationally inefficient) problems using a symbolic algebra package.

On the other side of the spectrum from general purpose symbolic algebra systems there are systems which perform some subset of the inferences provided by the Quantity Lattice. Like the Quantity Lattice, they use specialized representations to make the inference algorithms more computationally efficient.

The temporal reasoning system of [Allen] uses a representation similar to the one used to store qualitative relationships in the Quantity Lattice. Although Allen's system uses time intervals and we use time points, the basic difference is really in the implementation. Where Allen's system computes the transitive closure of the relationships every time an assertion is made, the Quantity Lattice infers a relationship only upon demand. Although it might seem to be more efficient to compute the closure, in practice we have found that the closure algorithm infers many more relationships than are actually needed, and is thus less efficient overall. For example, in the semiconductor fabrication simulation example presented in Section 3 there are 1357 quantities and therefore over 1.8 million potential relationships between quantities. However, during the simulation only 5009 (0.27%) of those relationships are actually needed.

In designing problem solvers which use the Quantity Lattice, we have found it useful to be able to tell the system "I am interested in the relationship between **A** and **B** — let me know if it ever changes." This feature, also used by [Dean], is implemented with a mechanism which associates *demons* with relationships in the Quantity Lattice graph. If the relationship changes, then the demon is fired.

The main problem with this scheme is that in order to be complete, the system must explicitly check all relationships which have demons to see if they have changed whenever any constraint is added. This would involve one graph search for each such relation and is clearly not reasonable computationally. A compromise position is to check only those relationships which are reachable by a path length of N or less from the relationship was added. Although this scheme does not necessarily cover all the relationships which logically might have changed, surprisingly an N of only 1 has been found to be sufficient in practice for the domains which we have explored. This same scheme has also been used by the time-map manager of [Dean] which uses an N of greater than 1.

Several researchers have incorporated some degree of qualitative and quantitative numeric reasoning. The DEVISER planning system [Vere] maintains qualitative temporal relationships in the form of a plan network, but associated with each plan node are numeric intervals which indicate the range of start and end times for the node. The interpretation of these intervals is identical to that of the Quantity Lattice — the real value lies somewhere in the interval. DEVISER uses techniques similar to interval arithmetic and numeric constraint propagation described in Section 2.2.2 to maintain the constraint that *start time* = *end time* + *duration*, where *duration* is a real number. Inconsistencies in a plan's schedule are detected if the upper bound of an interval is constrained to be less than its lower bound.

The Quantity Lattice can perform the same inferences with two major advantages. First, the duration of a plan step can be represented as an arbitrary expression, such as "$B + 5$." In DEVISER the duration must be a real number and the temporal constraints cannot be applied until the duration is known exactly. Second, the Quantity Lattice integrates qualitative and quantitative knowledge in such a way that new qualitative relationships can be inferred as more quantitative information is known (see Section 2.2.2). This integration is lacking in Vere's temporal reasoner. The system of [Allen] represents the quantitative duration of time intervals, but does not allow durations to be added together — something which is necessary to achieve at least the level of performance that Vere's system reaches.

A system which approaches the Quantity Lattice in expressive power is the fuzzy spatial reasoner of [Davis]. A similar fuzzy number representation is used in [Dean], but it only does addition and subtraction and the intervals are bounded by integers, not reals. As in the Quantity Lattice, [Davis] represents the value of expressions using intervals and performs constraint propagation to narrow the intervals. However, evaluation of arithmetic expressions is done using a Monte Carlo technique rather than interval arithmetic. This technique overcomes some of the disadvantages of pure interval arithmetic, but it is rather expensive computationally. The representation of qualitative relationships is handled by placing one quantity in a *local frame of reference* of another quantity. However, it is not clear whether a quantity can be placed in more than one local frame of reference, that is, whether qualitative partial orders can be represented. In any event, there seems to be no facility for inferring new qualitative relationships as can be done using relational arithmetic techniques, so the range of inferences performed is still smaller than the Quantity Lattice.

5 Conclusions

"Commonsense" arithmetic reasoning is an important form of reasoning. We have presented the Quantity Lattice, a system which performs many of these commonsense arithmetic inferences. We believe that the Quantity Lattice offers a reasonable balance between expressive power and computational complexity.

The range of inferences performed by the Quantity Lattice was carefully chosen by observing the types of arithmetic reasoning used in doing the qualitative and quantitative reasoning tasks needed in the domains of geologic interpretation [Simmons] and semiconductor fabrication diagnosis [Mohammed]. The algorithms used by the Quantity Lattice are designed for the range of assertions and queries commonly found in these and similar real-world domains. The resulting system smoothly integrates qualitative and quantitative information, ordinal relationships, and arithmetic expressions. The various types of knowledge constrain one another to enable more powerful inferences to be performed.

At the same time, the computational complexity is quite modest. The worst case for each assertion or inference is $O(E * R)$ while, in practice, the average case is much better as only small portions of the Quantity Lattice need to be traversed for each operation. Finally, all the inferences performed by the Quantity Lattice are recorded along with their justifications which facilitate retraction and the generation of explanations.

I would like to thank Randy Davis, Walter Hamscher, Dan Carnese, Mark Shirley and Jeff Van Baalen for thoughtful suggestions for improving this paper. Thanks to Rich Zippel for his insights on MACSYMA. I also thank Brian Williams and John Mohammed for their suggestions gained through using the Quantity Lattice.

References

[Allen] Allen, James. "Maintaining Knowledge About Temporal Intervals," CACM, vol. 26, no. 11, 1983.

[Davis] Davis, Ernest. "Representing and Acquiring Geographic Knowledge," Yale University Research Report 292, 1984.

[Dean] Dean, Thomas. "Temporal Imagery : An Approach to Reasoning about Time for Planning and Problem Solving," Yale University Research Report 433, October 1985.

[Forbus] Forbus, Kenneth. "Qualitative Process Theory," AI Journal, vol. 24, 1984.

[Mohammed] Mohammed, John; Simmons, Reid. "Qualitative Simulation of Semiconductor Fabrication," AAAI-86, Philadelphia, PA.

[Simmons] Simmons, Reid. "Representing and Reasoning About Change in Geologic Interpretation," MIT AI Technical Report 749, December 1983.

[Vere] Vere, Steven. "Planning in Time : Windows and Durations for Activities and Goals," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-5, no. 3, May 1983.

[Williams] Williams, Brian. "Doing Time : Putting Qualitative Reasoning on Firmer Ground," AAAI-86, Philadelphia, PA.