USING QUALITATIVE REASONING TO UNDERSTAND FINANCIAL ARITHMETIC

Chidanand Apté and Se June Hong

IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Abstract

This paper describes a general mechanism for the qualitative interpretation of simple arithmetic relations. This mechanism is useful for the understanding and reasoning about domains that can be modeled by systems of simple arithmetic equations. Our representation attempts to model the underlying arithmetic in its complete detail. Reasoning from these forms provides the completeness and consistency that cannot be always guaranteed by a pure production-rule based system. We describe an experimental architecture for Equation Reasoning (ER), and illustrate its applicability using examples from the financial domain.

1 Introduction

One popular form of representation for building expert systems is production rules. This representation has been found to be highly successful for encoding empirical knowledge directly elicited from human experts. This empirical knowledge is normally based on the human expert's experience at solving specialized problems. Many times the underlying basis for this knowledge is hard to formulate, and rules are then the most expedient way to encode the problem solving knowledge.

A class of problems exist for which it is possible to develop an underlying model of problem solving. Expert problem solving rules in such domains usually turn out to be specialized *pre-compiled* statements that are, in fact, derivable from the underlying theory. For building computer based problem solvers, it would then seem more advantageous for these systems to directly represent the underlying theory and reason directly from this representation just as a human expert would in the absence of pre-compiled rules. Intuitively, we can see the advantages of circumventing the problems of completeness and consistency which arise when one attempts to directly transfer a human expert's situation specific compiled rule knowledge into a computer program. However, on a more practical level, underlying theories, possibly well defined, often prove to be computationally expensive, and the trade-off between expediency and accuracy usually leans towards the use of production rules, even if it implies painstaking knowledge engineering efforts to ensure maximal coverage by the knowledge base.

There do exist some problem domains for which the underlying model is no more complex than simple arithmetic equations. It is an interesting research issue as to whether we can draw upon principles and mechanisms of causal and qualitative modeling for representing these simple underlying models, and use them as augmented problem solvers or support tools for knowledge acquisition and explanation generation.

Human experts rarely reason from underlying models, however simple, only because they have a truly vast store of applicable surface rules. On the other hand, it is not impossible for one to use a "reasoning from first principles" approach in the absence of such rules, when given a problem that can be abstracted down to its underlying mathematical form. When we try to examine, understand, and solve problems that are governed by numerical formulations, it is possible for us to use our *knowledge about numerical expressions* to understand how we can make the correct assumptions and approximations to get a quick feeling for the expected behavior of the unknowns in the solution. We may then progressively refine or restrict our approach to obtain more precise and accurate solutions. Many times, exact values of variables required for solving the problem are hard to come by, and inexact or estimated values need to be used for obtaining qualitatively reasonable solutions.

For a computer program to be able to use this approach, we need an explicit representation for the generic knowledge that can reason about numerical relations. Mechanisms are required for applying this knowledge to a situation that is modeled by equations. Strategies need to be developed for combining this domain independent equation based reasoning with the domain environment, which may include heuristic reasoning for solving problems that are not entirely based on pure numerical considerations.

1.1 Motivation

Our interest in this mechanism stems from a desire to build knowledge based automatic reasoners in the domain of finance. Financial planning and analysis is centered around a couple of dozen of arithmetic equations. Expert finance specialists seem to use a combination of numerical computations and heuristic rules. Upon closer examination, it turns out that a major portion of these heuristics are in fact derivable from exactly the same equation set (that are used in the computations), using a qualitative reasoning approach. It then seems natural to use a core representation for this basic equation set, and implement both quantitative and qualitative problem solvers that work off the same single representation.

The financial application is an extremely appealing problem from the viewpoint of building AI systems. There are sufficiently complex yet not impossible problems in this domain that merit closer inspection. Other than problem solving mech-

anisms, there is potential for applications of advanced knowledge representation techniques, problem solving control mechanisms, natural language processing, and many other AI notions. Knowledge based systems for business and finance may prove to be a rich arena for testing the fusion of many diverse AI mechanisms and concepts.

1.2 Approach

Assume an artifact whose behavior is governed by the equation $a + b^2 - c = 0$. What can we say about the behavior of the variables a, b, and c? Very little, given no further particulars. However, if we are told that typically a is extremely small compared to b , and that both a and b are typically greater than 1, and then asked how a small change in a would affect c , it would be relatively easy to answer. Because of the relation between a and b, a significant change in c will be only noticed if there is a change in b, and not otherwise. What we just did was apply our qualitative knowledge about arithmetic relations to the above equation, and made a statement of effect based on certain domain specific knowledge about the variables in the equation.

How should a computer based system solve a similar problem? The quick way is to build a rule based system that will incorporate rules like *"If 'a' is extremely small compared to 'b', and 'a' is changed by a small amount, then 'c' will not change significantly* or some variation thereof. We could write countless rules just for the equation $c = a + b^2$ for covering different combinations of conditions. If our system is modeled by some dozens of different equations then we need to repeat these countless rules for each of the equations. The folly in the rule based approach is obvious; the clean way to do this is to separate the equations and their domain specific knowledge into a distinct representation, and use domain independent knowledge about arithmetic relations to reason about what is represented.

Our architecture is based on what we view as the clean way. All equations that are available to model a problem or parts of it are stored in an Equation Base (EB). A special purpose problem solver for Equation Reasoning (ER) can inspect, manipulate, and reason about the equations in EB using its expert rules about arithmetic relations and expressions, in conjunction with domain specific Situational Knowledge (SK) about the current states of the variables involved. ER can be asked to determine the consequences on a variable(s) in a specified state, via a query. As a side effect, ER may be also queried for a symbolic solution for a specified variable (that can be used in computing the value(s) of that variable). ER is itself completely domain independent, all it knows about is simple arithmetic operations and their effects. It is a domain dependent problem solver that has to transform problems from the domain to specialized queries in the equational domain, and transform answers in the equational domain to inferences in the application domain.

1.3 Related Work

The use of a central "base" of equations to perform intelligent problem solving can also be seen in the work of [Kosy and Wise 84], where an equation base is used as a self-explanatory financial planning model. The use of equations in this work is to mainly give the system's spreadsheet like computation facility an *a posteriori* reasoning capability to generate explanations for computed values from the underlying equations. Our effort is more in the development of an *a priori* reasoning mechanism to solve for and generate problem solving steps or "rules" as and when required.

More recent attempts in the use of qualitative reasoning for solving financially related applications can be seen in [Hart et al. 86]. The major difference here is that this approach is considering more abstract level representations of financial activities as a basis to reason with, and not attempting to represent the underlying "arithmetic", as we wish to.

2 Representation and Reasoning about Equations

Consider a system to be reasoned about that is modeled by a set of equations. In the course of problem solving, we have to perform both qualitative inferences and numerical computations. Then given a problem specification, one way to solve that problem is to use an available production rule statement or a computable function to produce a required answer. What if the rule or the function is not available? Since we know that the system under consideration is modeled by a set of equations, we transform the domain situation problem into a query that consists of a goal accompanied by a constraint set on the variables. We then perform some symbolic manipulation and qualitative reasoning on the equations, to derive an answer to the query. The derived answer is then transformed back into a domain inference or value. This mode of operation is illustrated in Figure 1.

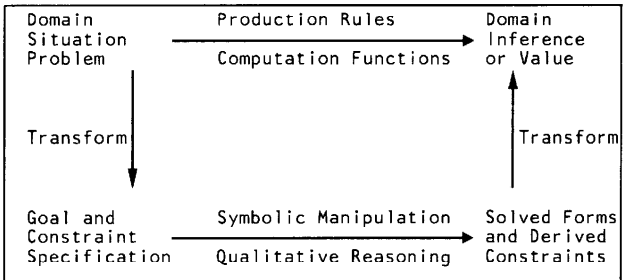

Figure 1. Reasoning from equations: *In the absence of available rules or functions, reasoning or solving may be done at the level of the underlying equations*

We represent all equations in their unfocused prefix notation in an *Equation Base*. For example, the equation $x = y \div z$ can be rewritten in its unfocused form as $x \times z - y = 0$. Corresponding to this equation, the form stored in the equation base would be *(- (× x z) y)*. To reason about or compute a variable that belongs to this equation set may require the solving of these equations to first obtain a focused symbolic solution for the variable. Classical computer based systems for symbolic manipulation of algebraic forms (e.g. MACSYMA, SMP, SCRATCHPAD II [Wolfram 85]) incorporate advanced general purpose algorithms for operations such as polynomial manipulation and symbolic integration. For the simple arithmetic world, there are more restricted but efficient approaches [Derman and Van Wyk 84], [Hansen and Hansen 85]. We use a simple *Solve* facility, based on the latter, for the purpose of solving for a variable of interest from a given set of equations in the equation base. This capability is required for two pur-

poses, for solving for a variable so that its value may be computed, and for solving for a variable so that its behavior may be reasoned about, in a focused manner, under weakly specified conditions.

The major components of our problem solving mechanism are:

- An equation base (EB), for storing all the available equations in their prefix unfocused notation.
- An equation reasoner (ER), that uses specialized and qualitative knowledge about arithmetic for solving and reasoning about the contents of EB.

Figure 2 illustrates the architectural organization of the overall equation reasoner. Various components will be appropriately described, although the focus in the remainder of this paper will be upon the inference capabilities of the qualitative equation reasoner (ER) and the knowledge it employs.

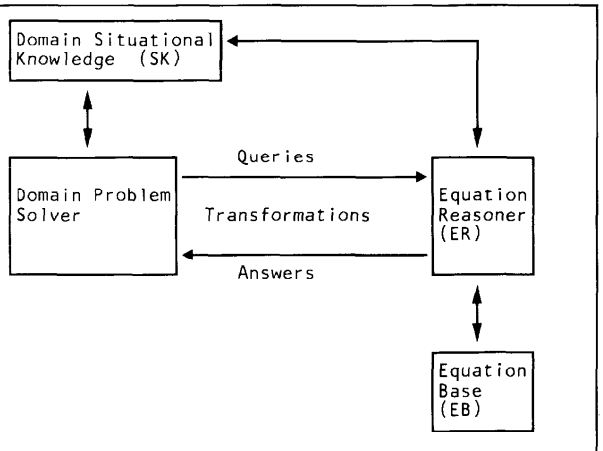

Figure 2. Overall organization of an Equation Reasoner: *Indicates how a single equation base may be used for both numerical computation and qualitative reasoning activities*

Typically, a domain problem solver will send off a request to ER for a computable function for a certain variable, or for finding out a qualitative set of consequences on variable(s) under certain situation specific constraints. These requirements are specified as part of a *query* statement that is composed by the domain problem solver. Some distinct types of queries that ER can produce answers for are:

- What is the function for computing the value of the domain variable V?
- Given a variable V and a constraint set on it $c(V)$, what are the implied constraints on one or more of the variables $(x_1, ..., x_n)$?
- Given variables $(x_1, ..., x_n)$, and constraints on them $c(x_1), ..., c(x_n)$, what are the implied constraints on the variable V?

Producing an answer for the first query is just a matter of simple symbolic solving for producing a computable expression for a focused variable V. A domain problem solver would typically request ER to produce this if it didn't already have available to it a function for computing a value for the variable under question. Of more interest is the way ER handles other types of queries. In a sense, producing answers for them also requires the symbolic solving for a focused variable V. In addition, ER can apply its general knowledge about arithmetic functions to *situation specific* and *query specific* domain knowledge about the variables involved to come up with some qualitative solutions. Thus, ER can solve for a variable V to obtain a form $V = f(x_1, x_2, ..., x_n)$, and use weakly specified constraints on x_1 ... x_n to derive some qualitative descriptions about V. ER can also apply this reasoning in the reverse direction, i.e., given some weakly specified constraint requirements on V, it can derive some qualitative statements about the states of the variables x_1 ... x_n.

A query posed to ER includes two set specifications, a *in-terms-of variable set*, and a subset of this, a *controllable variable set*. Exactly one member of the controllable set is identified as a *focused variable*, and one or more members of the in-terms-of set are associated with a *constraint list*. Determining the relevant members of this set is done by the domain problem solver using the situational knowledge base (SK). ER will solve for the specified focused variable, and in the process, will only inspect and keep track of qualitative effects on the controllable set variables. Any other variables that may come up in the symbolic solution will be considered invariant in the current situation. The in-terms-of variable set is a specification of variables in terms of which the focused symbolic solution is to be computed. Often, EB may contain multiple independent solutions for a variable, in which case specifying a in-terms-of set helps to restrict the number of solutions that are to be inspected. In some other cases, though EB may have a unique solution for a variable, a in-terms-of variable set specification is used to prevent the substitution of sub-expressions in a symbolic solution to finer resolutions. This is indeed the case in many situations when sub-expressions directly correspond to some domain concepts of current interest and the reasoning is to be done in terms of these concepts and not anything finer.

ER's general knowledge includes what we view as the qualitative knowledge about arithmetic relations. Currently, ER's knowledge is limited to the more common arithmetic notions, including *Addition, Subtraction, Multiplication, Division, Exponentiation, Summation expressions, Binomial expressions, Rounding, Integer parts, Fractional parts, Sign, Magnitude.* ER knows properties about these more common arithmetic relations. For example, given a relation x^y, ER can use its knowledge about exponentiation to make various inferences. The exponentiation knowledge includes rules that combine notions of even powers, odd powers, sign and magnitude of base, etc. An example of a qualitative rule about exponentiation is *If the base is a positive real fraction less than 1, then the result will decrease as the exponent increases.* A related rule would be *If the base is a positive real number greater than 1, then the result will increase as the exponent increases.*

Which of these rules to apply to a given situation is very much dependent upon the assumptions that can be made. An important additional ingredient that is therefore used by ER is simplifications and approximations based on assumptions that can be made about the operands involved in a relation. The ability to make these assumptions relies very much on domain and environment specific facts that can be gathered about these oper-

ands. These facts may be thought of as a combination of domain knowledge that is partly query dependent and partly query independent situational knowledge (SK) that can be accessed by ER during its reasoning process. The types of facts that ER will attempt to assess for a query using the domain situational knowledge base (SK) include *Is the operand a domain concept?, Is the operand's domain/situation value available?, Is the operand's domain/environment value available?, Does the operand have a domain/situation typical/default value?, Do the operands have typical/default relative magnitudes?.* Available data in SK regarding these questions in combination with ER's general knowledge about arithmetic make it possible to infer and propagate qualitative assessments about an arithmetic relation.

Many frequently occurring patterns in arithmetic expressions possess special properties that can be used for making quick qualitative interpretations. For example, the quadratic form $a \times x^2 + b \times x$ is constrained in its behavior by the special role x plays in the relation (think of the quadratic curve). However, if the terms $a \times x^2$ and $b \times x$ are inspected independently, we may possibly make locally correct but globally incorrect interpretations. One way to avoid this problem is to look for patterns matching a library of known special relations, before breaking into sub-expressions. To make this pattern match problem easy, the equation base (EB) keeps some such special relations (that are expected to frequently occur) in a pre-parsed form so that a special relation can be detected just as any standard arithmetic operator. For example, the domain that we wish to apply ER in has one very frequently occurring pattern $(1 + y)^{-z}$. When an equation containing this form is entered into EB, a preprocessor replaces the form by *(binomial1 y z)*. From the viewpoint of numerical computation, all required information is preserved too (e.g. a function *binomial1* can easily compute $(1 + y)^{-z}$ from its two arguments y,z).

Assume that EB contains three equations *(- (V (× x w)))* (corresponding to $V - x \times w = 0$), *(- (× w (- 1 (binomial1 y z))) y)* (corresponding to $w \times (1 - (1 + y)^{-z}) - y = 0$), and *(- (x (× s t)))* (corresponding to $x - s \times t = 0$). Note that EB represents a binomial term not in its fine details but rather as *(binomial1 y z)*. Assume ER receives a query that requires reasoning about the focused variable V, with a in-terms-of variable set specification of *(x,y,z,V)* and a controllable set specification of *(x,y,V)*. ER's solution for V would be $(x \times y) \div (1 - (1 + y)^{-z})$ (note that x is not further solved for because it is specified as a in-terms-of variable). The representation ER uses for building this solution is a variation of the standard prefix notation for storing expressions. For one thing, certain recognizable arithmetic concepts (e.g., a binomial term, in this case) are preserved as such, and not expanded into finer sub-trees. Also, appended with each operation and its operands is a property list that is gathered from an active domain specific situation environment.

In its most general form, the symbolic solution for a focused variable is of the type *(concept operand1 &optional operand2 ...)* where *concept* is an arithmetic operator or special form and each *operand* is itself an internal focused variable or a domain variable. ER makes two passes through this solution, once top-down for propagating qualitative assessments to each of the controllable variables, and once bottom-up for aggregation of individual controllable variables' qualitative constraints to their parent focused variable. For example, if the above query comes in with a constraint on V, *"decrease"*, then ER would use its two-pass propagate-aggregate mechanism to come up with the consequent constraints on x, *"decrease"*, and on y, *"decrease"*.

ER can also detect conflicts in assessments for a variable and perform local backtracks for making consistency corrections. For example, consider a focused expression for a variable p to be $(q - t) \div (r - t)$ Suppose we wanted to propagate a constraint on p, *"increase"* to its domain variables. Using its knowledge about "÷, -", ER would see initially conflicting assessments for t, *"decrease" and "increase"*. One way to handle such problems is to include a form like the above in EB/ER's notion about special arithmetic expressions. Then such a pattern would be detected before propagation takes place, ensuring correct actions to be taken. However, there are innumerable such patterns with special properties, all of which cannot be possibly cataloged in advance. In the case of the above form, assuming it to be not pre-compiled in advance, ER first posts a tag for the numerator *"increase"* and a tag for the denominator *"decrease"*. It then posts a tag for t, *"decrease"* when propagating constraints to the sub-expressions in the numerator. It then pursues the denominator, attempting to post a tag for t, *"increase"*. A conflict is detected, and ER will backtrack to the immediate parent concept that subsumes the roots of the conflict, which in this case is $(q - t) \div (r - t)$. ER first attempts to resolve this conflict by considering the relative magnitudes of q and r (based upon data in SK). However, in the absence of any special knowledge about this concept, *ER resorts to simulation techniques* to propagate constraints on q, r, t. Within the bounds of situational constraints provided by SK, ER will actually plot the behavior of this term for some numerical value assignments to the variables involved, to come up with some reasonable qualitative assessments for those variables.

How does this mechanism apply to a real example? The intent is to use ER to solve a class of problems that arise in financial planning and analysis. The following section will describe the application of ER to qualitative reasoning about the arithmetics of finance.

2.1 Understanding the financial impact of a capital acquisition

When a corporation acquires a major item such as a large computing system, it has available to it various financing options which can be used for the acquisition. Which option to use is governed by certain well defined concerns of how the transaction could potentially affect the corporations' financial ratios and books, and by some related but not so well defined qualitative issues (e.g. a corporation may always choose a particular bank to finance its acquisitions because it "likes" the way the bank treats its corporate customers). ER's usefulness is limited to the first category of concerns. The financial ratios and book entries are governed by equations, some of which are shown in Figure 3.

The two major types of financing available to a corporation are leasing and purchasing. Of course, there are several sub-types in each category. Many times, a corporation may choose one method in one time interval and another in a different time interval. It may also choose different financing methods in the same time interval for different capital acquisitions. To determine a corporation's preferred financing method for a specific

$$\text{CurrentRatio}_t = \frac{\text{CurrentAssets}_t}{\text{CurrentLiabilities}_t}$$

$$\text{PercentageDebt}_t = \frac{\text{Debt}_t}{\text{TotalAssets}_t}$$

$$\text{EarningsPerShare}_t = \frac{\text{AfterTaxProfits}_t}{\text{TotalShares}_t}$$

$$\text{CashFlow}_t = \sum_{i=1}^{n} \text{CashFlowItems}_{t,\,i}$$

$$\text{TaxBenefits}_t = \text{TaxRate} \times \sum_{i=1}^{p} \text{TaxExpenseItems}_{t,\,i} + \text{TaxCredits}_t$$

$$\text{CashFlowAfterTax}_t = \text{CashFlow}_t - \text{TaxBenefits}_t$$

$$\text{CumulativeDiscountedCashFlow}_t = \sum_{i=1}^{t} \frac{\text{CashFlowAfterTax}_i}{(1 + \text{DiscountRate})^i}$$

$$\text{ProfitLossExpenses}_t = \sum_{i=1}^{q} \text{ProfitLossExpenseItems}_{t,\,i}$$

$$\text{ProfitLossBenefits}_t = \text{TaxRate} \times \sum_{i=1}^{q} \text{ProfitLossExpenseItems}_{t,\,i}$$

$$\text{ProfitLossImpact}_t = (1 - \text{TaxRate}) \times (\text{Interest}_t + \text{Depreciation}_t)$$

$$\text{AfterTaxProfits}_t = \text{OperatingIncome}_t - \text{Taxes}_t - \text{ProfitLossImpact}_t$$

Figure 3. Sample of financial equations: *A complete set is stored in its unfocused form in ER's equation base.*

capital acquisition, its financial profile needs to be assessed. The reasons as to why a company chooses a certain financing method is based on the company's concerns about how the acquisition may potentially affect its balance sheet and income statement. The company's concerns may be expressed as requirements on financial ratios, which along with the balance sheet and income statement may be expressed as a set of equations. The impact of the acquisition on the ratios and books may then be determined by propagating qualitative constraints across these equations.

When a financial expert deals with problems in planning and analyzing financing methods, he is typically using heuristics like *If the acquisition decision maker wishes to maintain a high profit margin, then use the ProfitLossImpact as a comparison basis*, or *If the corporation is cash-poor, has a low effective tax rate, and a high borrowing rate, then it will be strongly inclined to lease.* The human expert has a countless number of such rules that he uses over and over again. These rules may be thought of as qualitative solutions to situation specific problems, and for a large part, may be derived from equations of the type illustrated in Figure 3. For an expert system to be able to reason about financial problems, it will either require all such rules to be encoded in advance, or have the ability to reason directly from the underlying equations, when there are no readily applicable surface rules. The advantages of the latter are obvious, and we are fortunate to have an underlying theory consisting, in large part, of simple arithmetic equations.

Qualitative reasoning comes in very useful for answering questions that come up while performing the financial planning and analysis required for choosing an "attractive" financing method for acquiring a capital intensive item. For example, during the planning phase, typical questions may be of the type:

1. *What financing method will best suit a corporation?*
2. *How does the use of a certain financing method for acquiring some equipment affect the corporation's EarningsPerShare ratio?*
3. *What financial criteria to use for ranking the outcomes of analyses of a series of financing alternatives?*

To answer such questions, we have to essentially inspect the financial equations, and determine the implications and constraints imposed upon certain variables under domain specific situations. The answers lie in the derived implications and constraints. Situational knowledge that is available is not always precisely (numerical values) specified. We do have to make use of whatever specifications are available for deriving a reasonable and rational qualitative solution.

Let us consider some examples of qualitative derivations of situation specific interpretations. Consider question 3. Suppose the domain specific financial planner requires an answer to the problem, *If a corporation's concern is its EarningsPerShare ratio, what criteria should be used for ranking analyses of several financing alternatives*? The query passed on to ER by the domain problem solver will include the controllable set *<EarningsPerShare, CashFlow, CashFlowAfterTax, CumulativeDiscountedCashFlow, ProfitLossImpact>*. The variable identified as the focused variable in this set is *EarningsPerShare*, with a constraint on it *"increase"*. ER will attempt to solve for EarningsPerShare, coming up with the solution:

$$\text{EarningsPerShare} = \frac{\text{OperatingIncome} - \text{Taxes} - \text{ProfitLossImpact}}{\text{TotalShares}} \qquad \text{E1}$$

Using the strategy described previously, ER will deduce a constraint on *ProfitLossImpact*, to be "decrease". When passed back, the domain problem solver can infer that *If EarningsPerShare is of concern, rank alternatives by ProfitLossImpact.*

Taking another example, consider question 2. Suppose the domain problem solver requires an answer to the query *What is the impact in the first year on the EarningsPerShare ratio if outright purchase (using bank financing) is used for acquiring a $5 million worth computing system* The query passed on to ER will consist of the controllable set *<EarningsPerShare, Interest, Depreciation>*. The variable identified as the focused variable in this set is *EarningsPerShare*, with constraints on *"Interest"* to be *"about a $1 million increase"* and *"Depreciation"* to be *"about a $1 million increase"*. ER will attempt to solve for EarningsPerShare", coming up with the solution:

$$\text{EarningsPerShare} = \frac{\text{OperatingIncome} - \text{Taxes} - (1 - \text{TaxRate}) \times (\text{Interest} + \text{Depreciation})}{\text{TotalShares}} \qquad \text{E2}$$

ER will assess (using a fact from SK that OperatingIncome is not extremely large compared to Interest + Depreciation) a constraint on "Earnings per share" to be "significant decrease". Upon passing back, the domain problem solver may make an inference of the type *An outright purchase of a capital asset may adversely affect a firm's EarningsPerShare ratio in the first year.*

3 Discussion

The production rule formalism is not well suited for many quantitative problem solving mechanisms. Attempts at addressing this shortcoming has resulted in systems like SOPHIE

[Brown et al. 82] and ELAS [Apté and Weiss 85] that integrate numerical models of a problem with appropriate heuristic models. What eluded most of these systems was the ability to inspect and reason about the quantitative/numerical models at a qualitative level. This was mainly due to the absence of a model of general knowledge about quantities and their relations, and an appropriate representation for the quantitative problem solving models.

The recent surge of activity in the areas of causal modeling and qualitative reasoning may be partially viewed as attempts to remedy this lack of representational detail. Although the use of causal models is not new [Weiss et al. 78], the more recent approaches have begun to address the qualitative and causal modeling of specific quantitative problem solving methods. Much of the hallmark work in this area appeared in a special issue of Artificial Intelligence [Kuipers 84], [DeKleer and Brown 84], [Forbus 84]. Our approach to a specific domain problem is very much inspired by this recent work in qualitative reasoning and simulation.

What is it we are doing that is different? It is primarily in the representational detail we wish to use. In much of the work on qualitative simulation and understanding of physical systems, researchers have attempted to use some form of abstract representation to model mathematical entities like differential equations for the purpose of reasoning about variables that are constrained by such equations. This shift in the representation is a forced requirement when dealing with complex forms like differential equations. A major disadvantage in abstracting the representation of mathematical forms is that certain information is lost, and that may exclude reasoning steps that require the comparison of magnitudes or computation of numerical values. The domain of mathematics that we deal with is simple arithmetic, and thus it is possible to achieve our more exacting requirement, with lesser computational overhead than that associated with typical causal models of complex mathematical systems.

We have formulated an architecture for equation reasoning, and studied its application to the financial domain. The examples illustrate the potential role of ER as a special purpose reasoner in a financial/business expert system. The main advantage of using ER is for its ability to understand and reason about constrained variables in a generalized way. Because ER uses a system's underlying equational model, it ensures completeness and consistency in its answers. Another important advantage is in the use of EB for storing equations in their true form, so that they may be used for performing both numerical and qualitative computations. ER's inference strategy works on a prefix symbolic solution for a focused variable. The inference is a two-pass propagate-aggregate mechanism that detects and resolves inconsistent assessments by local backtrack actions. The power of ER lies in its own general knowledge about arithmetic, and it is the "knowledge engineering" of this base that is most critical to the working of our architecture. It is interesting to note that from the viewpoint of a domain, ER solves problems using a "first principles" approach, although the knowledge base of ER itself is quite akin to "expert rules" about arithmetic equations.

ER is currently not able to provide all that is needed for use in an expert system. Our final goal is to strengthen ER's knowledge base by building a comprehensive catalog of general knowledge about arithmetic expressions. We are also enriching the constraint language used by ER for propagating and storing derived qualitative assessments. In particular, we would like to see an external problem solver be able to compare and contrast ER's solutions for problems of comparable nature. For example, a question of practical and theoretical interest to financial and business analysts is the corporate assessment problem of determining in advance whether a firm should lease or purchase capital assets. It requires posing multiple queries to ER (e.g. one for evaluating the impact of a lease on the firm's "xyz" ratio, one for evaluating the impact of a purchase on the firm's "xyz" ratio, etc.). The external problem solver then should be able to compare the returned qualitative assessments of the impacts on this ratio to determine which financing method is preferred. This requires ER to have the capability of computing qualitative assessments using a sufficiently rich constraint language.

Another potential area where ER could be further strengthened is in its role in the creation of queries. Currently, the domain problem solver uses the situational knowledge base (SK) to form these queries. However, there do exist causal connections between domain concepts and the variables in ER's equations. One way to find out about the relevant in-terms-of and controllable sets is to actually interrogate ER about the variable in question while forming a query for it.

3.1 Current Research

Many interesting extensions can be made to ER. Some of these are essential before ER can become a practical real world tool. Others are interesting research extensions. We present some open problems that we are currently investigating.

Strategies for integrating domain dependent heuristics: As we pointed out earlier, the possibility of mapping all domain specific problems into constraints on arithmetic relations is too ideal a situation. While we intend to use a generic mechanism to reason with numerical relations themselves, there do exist domain specific heuristics that just can't be mapped into our structure yet are useful to the reasoning process. We therefore require the mechanisms to be able to separately represent such heuristics and make use of them where feasible. Many times, very specific heuristics produce solutions that are in complete disagreement to what ER may produce. In other cases, ER may fail to produce a solution while specific heuristics might. It is cases such as these that will require ER to be integrated with domain dependent heuristics. Integrating ER to a domain problem solver brings up an interesting issue, how does a domain problem solver know that it does not have a domain heuristic and therefore needs ER? Or vice versa? We would like to resolve these issues at least partially in the course of our on-going investigation.

Knowledge Acquisition: Many concepts that are derived from a formula based representation in the course of solving a problem may be useful over and over again, during the same problem solving process, as well as in new ones. There is an interesting possibility of *caching* the query-answer pairs as rules for future use within the domain problem solver. We illustrated how some typical questions (2 and 3) are solved by ER (E1 and E2). The answers returned from ER, once transformed back into domain terms, can be viewed as a domain inference. For example,

question 2 coupled with the answer of E2 may constitute a rule for a specific situation. The advantages of caching are twofold. One, we do not need to pre-compile all such concepts in advance. Also, often used lines of reasoning should be available as surface rules, rather than having to do something akin to theorem proving, or reasoning from first principles each time the same problem arises. Automatic derivation and caching is one kind of knowledge acquisition, for which useful derivations are dynamically compiled, along with supporting knowledge structures. Work in the area of learning apprentice systems like LEAP [Mitchell et al. 85] and LAS [Smith et al. 85] demonstrate this capability. The problem of generalizing a line of reasoning for the purpose of caching a useful rule is extremely hard, specially if the reasoning process employs qualitative concepts. We are currently investigating this problem of generalization as it applies to financial planning and analysis.

Explanation One of the desired aspects of intelligent problem solvers is that they be able to explain or present their solution in a way that allow a human user to understand not only the solution reached, but also *how* and *why* they were reached. This usually requires the problem solver to maintain some kind of a trace on the bodies of knowledge used during the problem solving process. When the underlying behavior is governed by numerical relations, composing an explainable solution from a large body of quantitative data can be quite complex, unless explicit knowledge is encoded in advance for composing such explanations from numeric solutions. We would like ER to preserve traces of its reasoning process so as to compose intelligible explanations from them.

3.2 Concluding Remarks

ER is a stand-alone experiment in testing mechanisms for applying qualitative reasoning to systems of arithmetic equations. Our mechanisms are currently confined to the class of simple arithmetic relations and concepts. The efforts of many others to develop powerful qualitative reasoners for a more complex class of mathematical models has given us good insight to developing this approach for a simpler subset. Many practical problems may be based on this subset, and developing more efficient mechanisms for this special class of problems would be very useful. ER is being developed within the scope of a wider project that is investigating the applications of AI to building a system that will serve as a powerful interactive consultant for financial marketing decisions [Kastner et al. 86]. We continue to work towards the implementation and integration of ER with this system.

Acknowledgments

We would like to thank members of the Knowledge Systems Group at IBM Research and the anonymous referees for their useful comments, criticisms, and suggestions.

References

[Apté and Weiss 85] C.V. Apté and S.M. Weiss, *An approach to expert control of interactive software systems*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-7(5):586-591, Sept. 1985.

[Brown et al. 82] J.S. Brown, R. Burton, and J. de Kleer, *Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III*, In D. Sleeman and J.S. Brown (editors), Intelligent Tutoring Systems, pages 227-282, Academic Press Inc., 1982.

[DeKleer and Brown 84] J. de Kleer and J.S. Brown, *Qualitative physics based on confluences*, Artificial Intelligence, 24(1-3):7-83, Dec. 1984.

[Derman and Van Wyk 84] E. Derman and C.J. Van Wyk, *A simple equation solver and its application to financial modelling*, Software Practice and Experience, 14(12):1169-1181, Dec. 1984.

[Forbus 84] K.D. Forbus, *Qualitative process theory*, Artificial Intelligence, 24(1-3):85-168, Dec. 1984.

[Hansen and Hansen 85] B.S. Hansen and M.R. Hansen, *Simple symbolic and numeric computations based on equations and inequalities*, IBM Research Report RJ 4754, June 1985.

[Hart et al. 86] P.E. Hart, A. Barzilay, and R.O. Duda, *Qualitative Reasoning for Financial Assessments: A Prospectus*, AI Magazine, 7(1):62-68, Spring 1986.

[Kastner et al. 86] J. Kastner, C. Apté, J. Griesmer, S.J. Hong, M. Karnaugh, and E. Mays, *A knowledge based consultant for financial marketing*, IBM Research Report RC 11904, May 1986.

[Kosy and Wise 84] D.W. Kosy and B.P. Wise, *Self-Explanatory Financial Planning Models*, Proceedings of AAAI-84, 176-181, August 1984.

[Kuipers 84] B. Kuipers, *Commonsense reasoning about causality: deriving behavior from structure*, Artificial Intelligence, 24(1-3):169-203, Dec. 1984.

[Mitchell et al. 85] T.M. Mitchell, S. Mahadevan, and L. Steinberg, *LEAP: A learning apprentice for VLSI design*, Proceedings of the ninth IJCAI, 1:573-580, August 1985.

[Smith et al. 85] R.G. Smith, H. Winston, T.M. Mitchell, and B.G. Buchanan, *Representation and use of explicit justifications for knowledge base refinement*, Proceedings of the ninth IJCAI, 1:673-680, August 1985.

[Weiss et al. 78] S.M. Weiss, C. Kulikowski, S. Amarel, and A. Safir, *A model-based method for computer-aided medical decision making*, Artificial Intelligence, 11:145-172, 1978.

[Wolfram 85] S. Wolfram, *Symbolic mathematical computation*, Communications of the ACM, 28(4):390-394, April 1985.