In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05), Pittsburgh, PA.

Analysis of Strategic Knowledge in Back of the Envelope Reasoning

Praveen K. Paritosh and Kenneth D. Forbus
Qualitative Reasoning Group, Department of Computer Science,
Northwestern University, Evanston IL 60201.
{paritosh, forbus}@cs.northwestern.edu

Abstract

Back of the envelope (BotE) reasoning involves generating quantitative answers in situations where exact data and models are unavailable and where available data is often incomplete and/or inconsistent. A rough estimate generated quickly is more valuable and useful than a detailed analysis, which might be unnecessary, impractical, or impossible because the situation does not provide enough time, information, or other resources to perform one. Such reasoning is a key component of commonsense reasoning about everyday physical situations. We present an implemented system, *BotE-Solver*, that can solve about a dozen estimation questions like "What is the annual cost of healthcare in USA?" from different domains using a library of strategies and the Cyc knowledge base. BotE-Solver is a general-purpose problem solving framework that uses strategies represented as *suggestions*, and keeps track of problem solving progress in an AND/OR tree. A key contribution of this paper is a *knowledge level* analysis [Newell, 1982] of the strategic knowledge used in BotE reasoning. We present a core collection of seven powerful estimation strategies that provides broad coverage for such problem solving. We hypothesize that this is the complete set of back of the envelope problem solving strategies. We present twofold support for this hypothesis: 1) an empirical analysis of all problems (n=44) on Force and Pressure, Rotation and Mechanics, Heat, and Astronomy from Clifford Swartz's "Back-of-the-Envelope Physics" [Swartz, 2003], and 2) an analysis of strategies used by BotE-Solver.

1 Introduction

Consider the following examples:

- How many K-8 school teachers are in USA?
- What is the annual cost of healthcare in USA?
- What is Jason Kidd's point per game for this season?
- What is the annual gasoline consumption by cars in USA?
- How much is spent on newspapers in USA per year?

What these questions have in common is: 1) they seek numeric answers, and 2) even though exact answers might be hard to find, it is possible to generate good enough rough estimates. In this paper, we present BotE-Solver, an implemented system that can solve a dozen problems including the ones above. The two critical parts of such reasoning are using heuristics and strategies to simplify complex problems, and using one's *feel for numbers* to make suitable numeric estimates. At the heart of BotE-Solver is a library of estimation strategies. One goal of this work was to find a core collection of powerful reusable strategies. Given the open-ended nature of this domain, we are quite excited to report that there are only seven strategies that capture most of such reasoning.

The paper is organized as follows: next section presents the motivation for understanding BotE reasoning. Section 3 presents our approach and an analysis of the strategic knowledge and the seven core strategies followed by an empirical analysis of problems from Clifford Swartz's book. Section 4 presents a short description of the implemented system. Section 5 concludes with future work.

2 Motivation

BotE reasoning is useful and practical. In domains like engineering, design, or experimental science, one often comes across situations where a rough answer generated quickly is more valuable than waiting for more information or resources. Some domains like environmental science [Harte, 1988] and biophysics [O'Connor and Spotila, 1992] are so complex that BotE analysis is the best that can be done with the available knowledge and data. BotE reasoning is ubiquitous in daily life as well. Common sense reasoning often hinges upon the ability to rapidly make approximate estimates that are fine-grained enough for the task at hand. We live in a world of quantitative dimensions, and reasonably accurate estimation of quantitative values is necessary for understanding and interacting with the world. How long will it take to get there? Do I have enough money with me? These everyday, common sense estimates utilize our ability to draw a quantitative sense of world from our experiences. We believe that the same processes underlie both these common sense estimates and expert's BotE reasoning to generate ballpark estimates.

One goal of qualitative reasoning (QR) is to understand and model common sense. Some of the central assumptions of QR in practice must be rethought when considering commonsense reasoning, as opposed to narrow domain expertise. There is a striking resemblance between the key characteristics guiding commonsense reasoning and BotE reasoning. Here are the five important characteristics that we believe underlie common sense and BotE reasoning, but are violated by most QR approaches:

1. *Incompleteness*. Domain theories are incomplete in terms of their coverage, and even what they do cover is incompletely covered.

2. *Concreteness*. Domain knowledge includes knowledge of many concrete, specific situations. These concrete descriptions are used directly in analogical reasoning, in addition to first-principles reasoning.
3. *Highly experiential*. Domain expertise improves through the accumulation of information, both concrete and abstract. Experience improves our abilities to reason through similar situations, and helps us develop intuitions for what is reasonable.
4. *Focused reasoning*. Instead of maintaining uncertainty and ambiguity for completeness, assumptions are made aggressively to tightly constrict the number of possibilities considered.
5. *Pervasively quantitative.* Our interaction with the real world requires concrete choices for quantities. For example, the amount of salt one adds while cooking a certain dish cannot be safely specified as "+".

3 Analysis of BotE Strategies

All BotE estimation problems ask for a quantitative value for some parameter. Our approach breaks down the problem solving into two distinct processes:

1. *Strategies*: Using strategies to transform the current problem into possibly easier problems.
2. *Estimation*: Coming up with a numeric estimate for a parameter. The number could already be known, or similar examples might be used to make an estimate.

In this section we start with a formalization of the problems and strategies. We then present a knowledge level analysis [Newell, 1982] of the strategic knowledge in BotE reasoning. We present seven different strategies that are compositional and cover a large range of problems from various domains.

An abstract way to represent a BotE problem is **(Q O ?V)** where **Q** is the quantity, **O** the object and **?V** is the unknown value that is being sought. For example, in the question "How many K-8 teachers are there in US?," **Q** is cardinality, and **O** is the collection of K-8 teachers in US. Note that such decomposition of a question into a quantity-object pair is not necessarily unique, for example, "How much money is spent on newspapers in USA per year?" can be decomposed into:

- **Q**=Cost, **O**=All newspapers sold every year in US
- **Q**=Annual Sales, **O**=All US newspapers

Given a problem as **(Q O ?V)**, a strategy transforms it into a set of other problems **{(Q$_i$ O$_i$?V$_i$)}** such that **?V$_i$** are already known or easier to estimate. Besides transforming the problem, each strategy contains the answers to the following questions:

- When does it apply?
- How to combine **?V$_i$**s to find **?V**?
- What is the confidence of the estimate of **?V**?

With strategies represented as above, there are three syntactic possibilities for a strategy based on what aspect of problem it transforms:

1. *Object-based:* **(Q O ?V)** ® **{(Q O$_i$?V$_i$)}**
2. *Quantity-based:* **(Q O ?V)** ® **{(Q$_i$ O ?V$_i$)}**
3. *System-based:* **(Q O ?V)** ® **{(Q$_i$ O$_i$?V$_i$)}**

Next we look at each of these in turn. We present the seven strategies which are numbered S1 through S7.

3.1 Object-based Strategies

An object-based strategy related an object, **O**, to a set of objects, **{O$_i$}**, such that the quantity values for those objects, **{?V$_i$}**,combine in a known way to estimate the original quantity, **?V**. Note that since we are estimating the same quantity, this combination function can only be addition or subtraction since **?V** and **{?V$_i$}** have to have the same dimensions. Below are three object based strategies:

S1. Mereology: The mereology strategy transforms an object into other objects that are its parts. If **Q** is an extensive parameter, then, **?V=**S**?V$_i$**. For example, the weight of a basket of fruits is the sum of weights of all the fruits and the basket. If **O** is homogeneous, i.e., composed of the same kind of objects, then the above sum reduces to a product of the number of parts and the value for each part. This strategy requires making a closed world assumption, namely, that we know all the parts of the original object. In order for this strategy to be applicable, the parts should be non-overlapping in the quantity. If **Q** is an intensive parameter, then, **?V=**S**w$_i$*?V$_i$**, where **w$_i$** is the weight of the ith part. For example, the density of a mixture is the weighted average of the densities of the constituents.

S2. Similarity: The similarity strategy transforms the object into other object(s) which are similar to it. For example, if asked for the population of Austria, a reasonable guess could be the population of Switzerland, based on the similarity of the two countries. The strategy can be implemented in two ways:

- **O** is similar to **O$_1$** ⇒ **?V=?V$_1$**
- **O** is similar to **{O$_i$}** Þ **?V=Average(?V$_i$)**

If two objects are similar, it doesn't warrant the inference that values of all the quantities for two objects are similar. For example, another grad student in my department probably gets paid similar to me, but doesn't necessarily weigh the same. Two similar basketball players might have similar height, but not necessarily two professors. This notion of what features can be inferred from a similar example was called *projectability* by Goodman (1955/1983). There is increasing psychological evidence that projectability is based on *centrality* of the feature [Ahn *et al*, 2000; Hadjichristidis *et al*, 2004]. A feature is central to the extent that features depend on it. In our above example, height is central to basketball players, but not to professors. We have operationalized this notion of centrality as the structural support of the inference in computation of similarity using the Structure Mapping Engine [Falkenhainer *et al*, 1989].

S3. Ontology: The ontology strategy tries to find other objects from the ontology hierarchy which might be used to guess the quantity in question. In the simplest form, if `O` is an instance of O_1, then we can use the knowledge about the class to guess the value for the instance. For example, if we know that Jason Kidd is a point guard[1], then we can use the knowledge that point guards are relatively shorter than other players on the team to guess his height. If we didn't have information about point guards, we could even use the fact that Jason Kidd is a basketball player to guess his height.

Clearly, the higher up we go, the less accurate will be the estimate. Conceptually, the category hierarchy can be considered as specific → subordinate→ basic-level→ superordinate [Rosch, 1975]. An example of the four levels of hierarchy will be a specific chair in my living room, armchair, chair, furniture. In this framework, we can see that not only the accuracy decreases as we go higher, but also that we can go no higher than the basic level. Besides that, as in the similarity strategy, the centrality of the quantity is proportional to the accuracy of the estimate.

3.2 Quantity-based Strategies

A quantity-based strategy relates a quantity, `Q`, to a set of quantities, $\{Q_i\}$, such that the values of these quantities (for the object `O`) can be combined in a known way to derive the original quantity. Note that the combination function has to satisfy dimensional constraints, i.e., `?V` and $f(\{?V_i\})$ have to have the same units, where `f` is the combination function. Below are the two quantity-based strategies:

S5. Density: This strategy converts a quantity into a density quantity and an extent quantity. Here, density is used in a general sense to mean average along any dimension: we talk of electric flux density, population density, etc. Rates, averages, and even quantities like teachers per student are considered densities. For example, number of K-8 teachers in US can be estimated by multiplying the number of teachers per student by number of students.

S4. Domain Laws: This strategy converts a quantity into other quantities that are related to it via laws of the domain. Domain laws include laws of physics as well as rules of thumb. For example, Newton's second law of motion, `F=m*a`, relates the force on an object to its mass and acceleration. The application of domain laws by the problem solver requires formalizing the assumptions and approximations implicit in the laws. This has been well explored in compositional modeling [Falkenhainer and Forbus, 1991; Nayak, 1994]. In BotE reasoning, since we are not interested in an exact answer, but an approximate estimate, aggressively applying approximations to simplify the problem solving becomes crucial. Some of the approximations are:

- *Geometry*: Assume simplest shape, e.g. Consider a spherical cow [Harte, 1988].
- *Distribution*: Assume either a uniform distribution, or a Dirac-delta (point mass).
- *Calculus*: Integrals can be simplified by sums or average multiplied by extent, and differentials by differences.
- *Algebra*: Use simplification heuristics to reduce the number of unknowns [Pisan, 1998].

3.3 System-based Strategies

System-based strategies transform both the quantity and the object into other quantities and objects. They represent relationships between quantities of a system as a whole. For example, for a system with no external force, the momentum remains conserved. This translates into a conservation equation that relates the masses and velocities of the parts of the system. It would seem that this effect can be obtained by sequentially applying a quantity-based and an object-based strategy (or vice versa) since all the above strategies are compositional. There are two reasons for keeping this a separate type of strategy: 1) it represents a reasoning pattern that is different, 2) sometimes it is much more efficient to apply a system-based strategy, e.g., applying conservation of momentum leads to safely ignoring all the internal forces which need to be made explicit if one is not applying a system-based strategy. The two system-based strategies are:

S6. System Laws: This class consists of physical laws that are applicable to a system as a whole. Many physical quantities remain conserved for a system, e.g., energy, mass, momentum, angular momentum, etc. As a result of this, often one can write a balance equation that relates the expressions that denote the value of the quantity in two different states of the system. To write a balance equation, appropriate assumptions about the system in consideration have to be made.

S7. Scale-up: This is often an empirical BotE strategy. A smaller model that works under the same physical laws can be used to estimate the quantity values for a full-scale prototype. To ensure that the scale-up is valid, all the dimensionless groups must be kept the same in the model and the prototype. For example, the Reynolds number is a dimensionless group that corresponds to the nature of flow (laminar, transient or turbulent), and for a flow model to be valid for scaling up, the Reynolds number must be the same in both situations.

3.4 Discussion

BotE reasoning is very powerful because of its applicability to many domains. Clearly a human expert, or a computer problem solver, can do better with more facts about quantities and more strategies. The above analysis is an

[1] The point guard is one of the standard positions in a regulation basketball game. Typically one of the smallest players on the team, the point guard's job is to pass the ball to other players who are responsible for making most of the points.

attempt to identify the core strategies that have a broad coverage.

We arrived at these strategies by an analysis of knowledge that our problem solver was using. To confirm that we have a set of strategies which have broad coverage, we manually went through Clifford Swartz's "Back-of-the-envelope Physics." Swartz's book is a collection of estimation problems (along with solutions) from various domains in physics. We looked at all the problems (n=44) from Force and Pressure, Rotation and Mechanics, Heat, and Astronomy. Even though the goal of our work is not tied to Physics, this provides an independent confirmation about our strategies. Table I shows the results of our empirical analysis. The table shows the number of instances of each strategy, and the corresponding percentage, for Swartz's book as well as for the set of problems that BotE-Solver can currently solve[2]. We went through every solved problem and every time a problem was transformed into another, it was classified as one of the seven strategies or "other" category. We counted an application of the ontology strategy if it was specifically mentioned, since trivially almost all problems apply that strategy, as most problems are not about specific instances but classes of things.

Table I: Distribution of strategies.

Source →		Swartz		BotE-Solver	
Strategy ↓		Number of times used.	% of times used	Number of times used.	% of times used
S1	Mereology	11	14	9	32
S2	Similarity	5	6	5	18
S3	Ontology	6	8	0	0
S4	Density	10	13	10	36
S5	Domain laws	29	37	3	11
S6	System laws	11	14	1	4
S7	Scale-up	2	3	0	0
	Other	5	6	0	0
	Total	79	100	28	100

Some observations from table 1:

Coverage: The seven strategies account for 94% of strategy use in the problems from Swartz's book. The 5 strategies in the "other" category contain four instances of designing experiments to estimate a quantity, and one instance of a complex problem from statistical mechanics. The experimental strategies exploit one of the seven strategies, but there is considerable complexity in designing a good experiment. We focus on conceptual back of the envelope problems. The three syntactic possibilities for strategies – object-based, quantity-based and system-based are complete. Furthermore, our analysis of problems from various domains and our problem solver leads us to conjecture that the seven strategies are the complete set for the task of back of the envelope problem solving.

Domain-specificity: Strategies S1 through S4 are domain independent, and account for 40% of strategy use, while 60% of strategies are domain specific, with the largest component being domain laws. Interestingly, only 15% of the strategies in BotE-Solver are domain specific. This is expected because the problems that BotE-Solver currently solve are from common-sense domains, as compared to Swartz's book where most problems involve applying laws of physics.

The above analysis was done after we implemented our problem solver. We are using the analysis as a guide in designing more elegant representation of strategic knowledge in the problem solver. In the next section, we present a short description of the current implementation of BotE-Solver.

4 Implementation of BotE-Solver

A computational model of problem solving has to formalize the problem representation, has to have access to domain knowledge and the ability to retrieve knowledge that might be relevant. It also needs to have strategies, which it can try when the problem is complex and the answer is not directly found. It needs to maintain the workspace, where it keeps track of the work done and progress made on the problem. We have implemented BotE-Solver, a problem solver that uses –

- A large knowledge base for domain knowledge.
- Suggestions as representation for strategies.
- AND/OR tree as a model for maintaining the workspace.

In this section we explain the above ideas, and then present the core algorithm.

4.1 Domain Knowledge

The Knowledge Base (KB) and the reasoning engine are part of background infrastructure that this work builds on. The contents of our knowledge base are a 1.2 million fact subset of Cycorp's Cyc knowledge base, which provides formal representations about a wide variety of everyday objects, people, events and relationships. Problems, solutions, strategies are all represented uniformly and stored in this KB. We use the FIRE reasoning engine, jointly developed in collaboration with Northwestern University and Xerox PARC. FIRE uses a special purpose database for storing the knowledge base. It can do analogical reasoning using structure mapping [Forbus *et al,* 2002], and has facility for adding various kinds of reasoning source that allow it to do specialized reasoning, such as spatial reasoning [Forbus *et al,* 2003].

4.2 Strategies

We represent strategies using suggestions. A suggestion provides a decomposition for the problem. The suggestion

[2] The set of problems solved by BotE-Solver is different from those in Swartz's book. BotE-Solver's problems are from commonsense domains as opposed to Physics.

HouseholdStrategy-ForCountingUnits in Figure 1 captures the idea that the number of cars can be

```
(defSuggestion HouseholdStrategyForCountingUnits
 :trigger
  (unitsTotal ?obj ?place ?time ?total-units)
 :test (ownedBy ?obj FamilyCohabitationUnit)
 :subgoals
  ((numberOfHouseholds ?place ?time
     ?num-households)
   (unitsPerHousehold ?obj
     ?units-per-household))
 :result-step (evaluate ?total-units
          (TimesFn ?num-households
            ?units-per-household)))
```

Figure 1. An example suggestion.

estimated by finding the number of households. It says that if we know that something is owned by households (refered to as FamilyCohabitationUnit in the KB), then we can find how many units of it are owned by multiplying the number of households with the number of units per household. There are four parts of a suggestion –

***Trigger*:** The form which is query for which the suggestion might be applicable.

***Test*:** Additional test conditions which must be true in order for the suggestion to work.

***Subgoals*:** A list of forms that this suggestion decomposes the current problem into. These are AND-subgoals, meaning if any one of them fails, this suggestion fails to solve the original problem. These subgoals are fully ordered.

***Result-step*:** The final step of the suggestion, which combines the answers to the subgoals.

Inside the suggestion, the variables are order scoped such that any variable introduced can be used in the subsequent parts of the suggestion. defSuggestion is a facility for the suggestion author – it expands the suggestion into assertions in predicate calculus that are stored in the KB.

4.3 Tracking problem solving progress

BotE-Solver uses an AND/OR tree[3] to track the progress as it is working on a problem. The mapping between the AND/OR tree and our representations is very direct. For a problem, there could be many applicable strategies, any one of which succeeding lead to a solution to the problem. This results in an OR node in the tree. A suggestion, on the other hand, introduces one or more subgoals all of which have to be solved in order to solve the original goal. This results in an AND node in the tree. An AND/OR decomposition lets us keep track of dependencies between the original problem and new subgoals introduced. These inferences are made by maintaining flags at each node, which are updated/propagated after every unit of problem solving.

As BotE-Solver works on a problem, it maintains its progress in an AND/OR tree as mentioned above. It also maintains an agenda, which is a list of things that it can do next. The agenda consists of suggestions that have been found that it can try, and subgoals that have been suggested. The agenda is ordered by difficulty estimates so that the first thing on the agenda is the easiest one. The solver generates solutions incrementally. The combinatorial possibilities of solutions from subgoals can be large, and in this kind of problem solving we are often interested in one, or at most few answers[4].

4.4 Feel for Numbers

A key part of doing back of the envelope estimates is the feel for numbers. Once we have the model that relates the parameter to other quantities that might be known, the reasoning bottoms out with making good guesses for numeric parameters. Sometimes the exact numeric parameter might be known. In many estimation problems, we are looking for typical, high, or low values for a parameter. For example, consider the question – What does a good gaming PC cost? Now, we know that a good gaming PC has a high RAM, expensive video card, and a fast processor, and everything else might be the usual fare. Being able to represent and reason with notions like high, low and typical values for quantity is a key aspect of BotE reasoning. We have built CARVE, a system that automatically builds symbolic representations of quantity by exposure to examples [Paritosh, 2004]. We have shown that representations augmented with symbolic representation of quantities lead to more accurate estimates [Paritosh, in preparation].

4.5 Results

BotE-Solver can currently solve 13 problems from commonsense domains. The system uses a library consisting of 24 suggestions. The suggestions are less abstract than the strategies, so we have multiple suggestions that capture the same strategy. We are currently in the process of rewriting our system to let us elegantly describe the strategies at the level of abstraction discussed in Section 2. A full list of problems, solutions and details of implementation are available elsewhere [Paritosh and Forbus, 2004].

Most of the times BotE-Solver finds an answer that is in the ballpark. The goal of BotE-Solver is to find an answer that is no more than an order of magnitude off on either

[3] Because the solutions are obtained and cached in a truth maintenance system, we get the functionality of an AND/OR graph, i.e., we don't re-solve an already solved node, although the underlying representation is a tree. The advantage of having a tree is that the propagation algorithms are much simpler.

[4] One way to do a "sanity-check" on a BotE estimate is to generate another estimate. We prefer this to the approach of keeping track of ranges and bounds for every estimation step.

side. Sometimes, the estimates being off can be an interesting thing. So in a question about total cost of healthcare in the US, it comes up with an estimate of 0.8 trillion US dollars, which is roughly half of the real value for this. This estimate is based on the cost of buying everyone personal insurance. In cases like these, carrying out an estimate and comparing it to the expected value might trigger a model refinement and the fact that one needs to know more to understand the process.

4.6 Related Work

The most similar project in this spirit was FERMI [Larkin *et al*, 1988]. FERMI used two general principles, *decomposition* and *invariance*, with domain specific knowledge to solve textbook problems in fluid statics, DC-circuits and centroid location. Our approach is simpler, more general, builds upon existing large knowledge bases, and is more concerned with the kind of breadth of common sense reasoning as opposed to natural science. Such reasoning has relevance to education, especially engineering. More than 90% of mechanical engineering seniors (100 at MIT, and 250 from five other universities) came up with wrong order of magnitude estimates of value of energy stored in a 9-volt "transistor" battery [Linder, 1999]. The responses varied by nine orders of magnitude excluding outliers! Having a clearer understanding of BotE reasoning at the knowledge level and computationally, might be helpful in fixing that kind of innumeracy.

5 Conclusions and Future Work

We presented BotE-Solver, a system that can solve fairly interesting estimation problems. An analysis of the strategic knowledge used by the system and a corpus of problems revealed a collection of seven core strategies. We are rewriting our solver to express these seven strategies in an abstract and elegant manner. Another interesting future direction will be to see if these same strategies can be used in explanation, as opposed to problem solving; e.g., in a system that comes up with explanations for numbers appearing in a news article.

Acknowledgments

This research is supported by the Artificial Intelligence Program of the Computer Science Division of the Office of Naval Research. The authors would like to thank Johan de Kleer and the anonymous reviewers for helpful comments.

References

Ahn, W., Kim, N. S., Lassaline, M. E., & Dennis, M., 2000. Causal status as a determinant of feature centrality. *Cognitive Psychology*, **41**, 361-416.

Falkenhainer, B. and Forbus, K. "Compositional Modeling: Finding the Right Model for the Job", *Artificial Intelligence*, **51** (1-3), October, 1991.

Forbus, K., Gentner, D. 1997. Qualitative mental models: Simulations or memories? *Proceedings of Eleventh International Workshop on Qualitative Reasoning*, Cortona, Italy.

Forbus, K., Mostek, T., and Ferguson, R. 2002. An analogy ontology for integrating analogical processing and first principles reasoning. In *Proceedings of IAAI-02,* July 2002.

Forbus, K., Tomai, E., and Usher, J. 2003. Qualitative spatial reasoning for visual grouping in sketches. In *Proceedings of the 16th International Workshop on Qualitative Reasoning,* Brasilia.

Goodman, N., 1955. Fact, fiction, and forecast. Cambridge, MA: Harvard University Press.

Hadjichristidis, C., Sloman, S.A., Stevenson, R.J., Over, D.E. 2004. Feature centrality and property induction. *Cognitive Science*, **28**, 45-74.

Harte, J. 1988. Consider a spherical cow: A course in environmental problem solving, University Science Books, Sausalito, CA.

Larkin, J. H., Frederick R., Carbonell, J. and Gugliotta, A. 1988. FERMI: A Flexible Expert Reasoner with Multi-Domain Inferencing, *Cognitive Science*, **12**(1), 101-138.

Linder, B.M. 1999. Understanding estimation and its relation to engineering education, Ph.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Nayak, P.P. 1994. Causal Approximations, Artificial Intelligence, **70**, 1-58.

Newell, A. 1982. The Knowledge Level, Artificial Intelligence, **18**, 87-127.

Nilsson, N. 1994. Principles of Artificial Intelligence, Morgan Kaufman.

O'Connor, M.P., and Spotila, J.M. 1992. Consider a spherical lizard: Animals, models and approximations, American Zoologist, **32**, pp 179-193.

Paritosh, P.K., 2004. Symbolizing Quantity. In Proceedings of the 26th Cognitive Science Conference, Chicago.

Paritosh, P.K., *in preparation*, Analogical Estimation.

Paritosh, P.K. and Forbus, K.D., 2004. Using Strategies and AND/OR Decomposition for Back of the Envelope Reasoning. In *Proceedings of the 18th International Workshop on Qualitative Reasoning*, Evanston.

Pisan, Y., 1998. An integrated architecture for engineering problem solving. Doctoral dissertation, Northwestern University, Evanston, IL, USA.

O'Connor, M.P., and Spotila, J.M. (1992). Consider a spherical lizard: Animals, models and approximations, American Zoologist, **32**, pp 179-193.

Rosch, E., 1978. "Principles of categorization" in Rosch, E. and B.B. Lloyd, (eds.), Cognition and Categorization, Hillsdale, N.J.: Erlbaum.

Swartz, C. E., 2003. Back-of-the-Envelope Physics. Johns Hopkins University Press, Maryland.