Finding Hypothetical Answers with a Resolution Theorem Prover

Debra T. Burhans and Stuart C. Shapiro
Department of Computer Science and Engineering
and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14206-2000
burhans@cse.buffalo.edu, shapiro@cse.buffalo.edu

[1]

Abstract

Resolution refutation is a powerful reasoning technique employed in many automated theorem provers. Various enhancements to resolution have enabled it to be used as a general question answering mechanism. Question answering systems employing resolution as the basic reasoning technique have been used to provide both "intensional" and "extensional" answers to questions by considering a theorem to be proven as a question. An intensional answer is a rule, such as "for all x and for all y if x is a cat and y is a dog then x detests y", and an extensional answer is a fact, such as "Rachel detests Fido". The notion of what constitutes an answer can be expanded so that, as resolution proceeds, the intermediate results generated *on the way* to finding an intensional or extensional answer may be regarded as answers. This view of resolution as answer generation, and resolvants as answers, requires an expanded notion of what constitutes an answer. A class of "hypothetical" answers is proposed, having the general form $X \Rightarrow Y$, where X can not be proven based on the information in the knowledge base. When there is not enough information in a knowledge base to provide an intensional or extensional answer, a hypothetical answer can be useful.

Introduction

Resolution refutation is a powerful reasoning technique employed in many automated theorem provers. Various enhancements to resolution have expanded its capabilities as a general question answering mechanism. In particular, question answering systems employing resolution as the basic reasoning technique go beyond simply answering the question of whether a particular theorem is consistent with a rule base, which would correspond to providing a yes or no answer to a question. Such systems provide two additional types of answers, often described as *intensional* and *extensional*. Intensional answers are rules, containing unbound variables. They frequently correspond to categories. Extensional answers are often termed facts, they contain no variables, and correspond to individuals. For example, if a student is consulting an electronic course advisement system, and asks which courses she needs to complete a computer science degree, an extensional answer would comprise a list of courses, such as *CSE 596, CSE 421*, and an intensional answer might have the form *one theory course and one hardware course*. Answers combining both extensional and intensional components are termed *mixed* (Motro 1989; 1994; 1996).

This paper considers intermediate results that are generated *on the way* to discovering intensional and extensional answers, as resolution proceeds. A category of *hypothetical* answers is proposed to describe some of these intermediate results. Hypothetical answers have the general form of an implication, where the antecedent can not be proved based on the current state of the knowledge base. The relationship between extensional, intensional, and hypothetical answers is discussed, as are search strategies for question answering, and future directions for this research.

Background

A traditional resolution refutation proof provides the answer to the question, "is this theorem consistent with a given rule base?". The negation of the theorem to be proven is added to the rule base, and if two clauses can be resolved to produce the empty clause then a contradiction is established, and the question is answered affirmatively. Otherwise the question may be answered negatively (closed world assumption) or not at all. If the theorem is a query, the production of the empty clause indicates the existence of an answer, but does not provide information about the answer.

The role of resolution as a question answering system was expanded by Cordell Green with the introduction of an answer literal (Green 1969b; 1969a). An answer literal is added to the clause created by negating a query, enabling a theorem prover to go beyond a simple "yes" answer by providing information about variable bindings used to complete a proof. The types of answers provided using this approach are often referred to as *extensional*. Consider the question *Who does Mary like?* An extensional answer to this question might be the fact *Mary likes John.* Prolog employs resolution in this manner, answering questions with facts, and failing to answer questions when no relevant facts are contained in the rule base being searched.

Cholvy and Demolombe (Cholvy & Demolombe 1986; Cholvy 1990) expanded upon Green's work, using a resolution theorem prover to find *intensional answers.* An example of an intensional answer to the above question is *Mary likes boys.* The reason this is considered a rule is that "boys" is a category rather than an individual. Thus, the answer given above might be represented as the rule, *for all X, if X is a boy then Mary likes X.* Cholvy and Demolombe recognized that intensional answers are often generated and discarded as the search of a rule base for extensional answers proceeds. In their work they use rules bases containing no facts, all answers produced are intensional.

Other Knowledge Representation and Reasoning (KR&R) systems, for example, Description Logics, employ a variation of resolution as a reasoning mechanism. Some, such as CLASSIC (Borgida & McGuinness 1996), are capable of providing both both extensional and intensional answers. ANALOG (Ali & Shapiro 1993) is a KR&R system designed to represent and reason about intensional information. It is capable of providing both intensional and extensional answers in response to questions.

Database researchers have also investigated the production of intensional answers. Motro has done considerable work on intensional answering in the context of relational databases (Motro 1989; 1994; 1996). He characterizes an intensional answer as "a description that implies part or all of the specific answer" (Motro 1989). Motro also recognizes "mixed" answers as those involving both extensional and intensional information. The intensional component of a database comprises the definitions of the data structures for the database, referred to as the schema, as well as integrity constraints (rules that describe relationships that must be satisfied by the facts in the database), and other definitions such as views and inference rules. An intensional answer is one that describes the data rather than presenting it directly: it is a component of the intensional portion of the database.

Imielinski has worked on intensional answering in the context of logic and databases (Imielinski 1987; Imielinski & Mannila 1996). In his system, intensional answers are viewed as intermediate steps along the way to extensional answers. Each of his intensional answers provides a complete specification of the extensional answers. One of the issues in intensional answering with respect to databases is that the intensional information is often based on a particular state of the data: when new data is added, rules that were formulated bottom up may no longer be complete or correct.

Considerable research has gone into intensional querying, and intensional query optimization, which is related to queries involving *intensional* predicates, or those predicates defined by more than one rule in Datalog (Godfrey & Gryz 1996). Intensional queries do not necessarily have intensional answers, and the work is not as directly relevant as that cited above.

The work described herein continues the tradition of using a resolution theorem prover as a reasoning mechanism and a rule base specified using first order predicate calculus as a knowledge representation to explore issues in question answering. This work expands on the current notion of answer by introducing *hypothetical* answers.

Types of Answers Produced using Resolution

Three types of answers have been described using resolution as a reasoning mechanism: yes/no answers (providing information that an answer exists, but no description of the answer), extensional answers (facts: *ground* answer descriptions), and intensional answers (rules: descriptions containing unbound variables). If the purpose of an answer is to provide the most useful information in response to a question, it is clear that in many cases simply knowing that an answer exists (yes/no answer) is not very useful. This led to the original enhancement of resolution by Green, and the introduction of the answer predicate and the extensional answer.

Considering an extensional answer as *the* answer to a question has remained the dominant paradigm for question answering, both in Artificial Intelligence and in the field of databases. A problem with this approach is that, in the absence of an extensional answer, no answer will be provided even though other useful information may be available. Neither Prolog nor Otter (McCune 1994) will provide an answer in the absence of extensional information.

In addition to the problem of ignoring useful information, it is clear that sometimes intensional answers are simply better than extensional answers. Given the question, *What barks?*, the intensional answer *dogs bark* is much more informative than a litany of individual dog names. Intensional answers are more general than extensional answers and are often more succinct. It is this feature of intensional answers that has propelled the interest in intensional answers from database researchers. From taxonomies of concepts to database views, intensional information is an important part of knowledge representation.

What happens when a piece of information necessary to infer an intensional or extensional answer is missing? The missing information could be the result of an oversight on the part of a rule base creator, or it could be due to a questioner providing a very underspecified question. For example, consider again the question, *Who does Mary like?* If information about Mary is lacking, but there is information about some types of entities liking other entities, hypothetical answers such as the following can be provided: *if Mary is a girl, then Mary likes boys*, and *if Mary is a girl,* *then Mary likes John.* The first of these is termed an intensional, hypothetical answer, and the second an extensional, hypothetical answer. They are hypothetical because it is not known whether Mary is a boy or girl, or possibly what Mary is at all. Such answers can help a questioner focus, by prompting her to provide greater constraints in a question, and they can alert a knowledge base designer that an important piece of information may be missing. Sometimes a question is deliberately underspecified when a questioner isn't sure herself what sort of answer she desires. For example, in an electronic course advising system, it may be desirable to leave your major underspecified when asking what courses you need to graduate. In some computer science departments there are both BA and BS tracks, in addition to computer engineering. It might be interesting to hear about different possible outcomes before constraining a question.

The form of answers can be described as follows. An extensional answer in its simplest form (that is, not including conjunctions and disjunctions) can be written $R(a, b, c, \ldots, n)$, where R is an n-place predicate and $a, b, c, \ldots, n$ are constants. An intensional answer in its simplest form can be written $P(X) \Rightarrow Q(Y)$, where X and Y contain unbound variables. An hypothetical answer in its simplest form can be written either $H(Z) \Rightarrow R(a, b, c, \ldots, n)$ or $H(Z) \Rightarrow (P(X) \Rightarrow Q(Y))$, where $H(Z)$ is an hypothesis relevant to the question that can not be proven given the current state of knowledge. In general, an answer can be viewed as having the following form, where, depending on the type of answer, at least one of these three components is present:
$(H(Z) \Rightarrow ((P(X) \Rightarrow Q(Y)) \Rightarrow R(a, \ldots, n))$

The three components of the answer being referred to as the hypothetical, the intensional, and the extensional component.

Generating Answers with a Modified Resolution Theorem Prover

This exploration of answer generation has been done using a modified resolution theorem prover as a reasoning mechanism and a rule base specified using first order predicate calculus as a knowledge representation. In the examples provided, it is shown that as resolution proceeds, if the rule base being searched contains rules that are relevant to a question, hypothetical and intensional answers will be produced. If an hypothesis is later proven, the hypothetical "answer" generated earlier will be subsumed by an intensional or extensional answer later in the resolution process. If there are facts relevant to a question, extensional answers will be found. Consider the following simple rule base used as an illustration. (The example was originally based on (Rich & Knight 1991, p.192).)

- cats would like to eat fish
- tuna are fish
- dogs would like to eat bones
- pete is a tuna
- charlie is a tuna
- rachel is a cat

Given this simple rule base, and the question *What would rachel like to eat?*, the following answers are produced (each answer is preceded by a gloss):

```
1. if rachel is a dog, then rachel would like
to eat bones

((DOG RACHEL)) =>
((BONE x0)) =>
((WOULDLIKETOEAT RACHEL x0))

2. if rachel is a cat, then rachel would like
to eat fish

((CAT RACHEL)) =>
((FISH x0)) =>
((WOULDLIKETOEAT RACHEL x0))

3. rachel would like to eat fish

((FISH x0)) =>
((WOULDLIKETOEAT RACHEL x0))

4. if rachel is a cat, then rachel would like
to eat tuna

((CAT RACHEL)) =>
((TUNA x0)) =>
((WOULDLIKETOEAT RACHEL x0))

5. rachel would like to eat tuna

((TUNA x0)) =>
((WOULDLIKETOEAT RACHEL x0))

6. rachel would like to eat charlie

((WOULDLIKETOEAT RACHEL CHARLIE))

7. rachel would like to eat pete

((WOULDLIKETOEAT RACHEL PETE))
```

Answers 1, 2, and 4 are hypothetical. Answers 2 and 4 should not ultimately be provided as answers because their (common) hypothesis can in fact be proven: the rule base indicates that rachel is indeed a cat, which is why 2 is subsumed by 3, and 4 is subsumed by 5. There is nothing in the rule base that says cats are not dogs, so 1 stands as a hypothetical answer that is not subsumed by any other answer. Answers 3 and 5 are intensional, their structure demonstrates that they are rules, and answers 6 and 7 are traditional, extensional answers.

The example is perhaps better motivated by showing what happens when the fact that *rachel is a cat* is removed from the list of rules. The resulting answers are:

```
1. if rachel is a dog, then rachel would like
to eat bones

((DOG RACHEL)) =>
((BONE x0)) =>
((WOULDLIKETOEAT RACHEL x0))

2. if rachel is a cat, then rachel would like
to eat fish

((CAT RACHEL)) =>
((FISH x0)) =>
((WOULDLIKETOEAT RACHEL x0))

3. if rachel is a cat, then rachel would like
to eat tuna

((CAT RACHEL)) =>
((TUNA x0)) =>
((WOULDLIKETOEAT RACHEL x0))

4. if rachel is a cat, then rachel would like
to eat charlie

((CAT RACHEL)) =>
((WOULDLIKETOEAT RACHEL CHARLIE))

5. if rachel is a cat, then rachel would like
to eat pete

((CAT RACHEL)) =>
((WOULDLIKETOEAT RACHEL PETE))
```

All answers are hypothetical because nothing about the identity of rachel is known. If a question answering system consideres answers that are intensional, extensional, or mixed, no answer would be returned as a result of this question, yet there is clearly some information that is relevant to the question that these hypothetical answers provide: in short, it may be the best information that is available. All possibilities that share the predicates of the question, namely, `WOULDLIKETOEAT`, are offered up as possible answers. Answer 1, 2, and 3 are hypothetical, intensional answers, and 4 and 5 are hypothetical, extensional answers.

Complete and Relevant Answers

A complete answer to a particular question is generally defined to be an answer that implies all answers to that question. Intertwined with the discussion of what constitutes a complete answer is the notion that an answer must be relevant to the question. Relevance for answers in most cases has been defined as sharing the predicates presented by the question.

In a rule base consisting solely of facts (no intensional information), the conjunction of all facts relevant to a question would be considered a complete answer. When rules are added, the picture becomes more complicated. When taxonomic relationships are represented in a rule base, a complete answer can be defined as the conjunction of all of the relevant answers that occur at the highest taxonomic level. Thus, in the first example in the above section, *rachel would like to eat fish* **and** *if rachel is a dog then rachel would like to eat bones* would be considered a complete answer.

Providing information about tuna, and specific tunas, is redundant and may be misleading when a category that includes all of those (fish) has already been given as an answer. This issue relates to the problem of providing an answer at the right level of detail, which is another important area of research in question answering. In this research this problem has been considered, but thus far it has taken the form of observations rather than formal theories. This represents an area of further research interest.

Earlier versions of the system used for the above examples included complicated definitions of what constituted the most relevant, or even the best, answers. Negative feedback on this approach led to the determination that relevance is best defined by a user, who is perhaps better acquainted with what is most important in an answer. The way a user can specify criteria for relevance is to provide the system with selection criteria to either direct the search for answers or to order the answers once they are all generated. This is discussed in the next section. The only built-in strategy employed at this point in the system is to use the negation of the query (containing the answer literal) as the initial set of support, which imposes the restriction that answers share predicates with the question.

Search and Ordering Strategies for Question Answering

Numerous search strategies have been developed for resolution theorem provers, all of which have the goal of generating the empty clause as quickly and efficiently as possible. When resolution is used primarily as a question answering mechanism, different search strategies and measures of success are needed.

A particular type of answer may be preferred over another, and other attributes, such as length of an answer, or the number of negative or positive literals, may be used to preferentially order answers. Some properties of answers, such as their categories (extensional, intensional, hypothetical), may be used as criteria to order sets of answers that have been generated. Other properties of clauses, such as choosing which clauses to resolve based on when they were generated, can be used to direct the search and thus the order in which answers are generated.

To aid in the exploration of search and ordering strategies for answers, our system associates the following attributes with clauses:

- the line number (lines are numbered as they are created)
- the clause
- the justification (assumption, from query, or the pair of line number for the lines resolved to produce this clause)
- number of negative literals
- number of positive literals
- list of clauses that subsume the clause
- list of clauses subsumed by this clause
- answer type (extensional, intensional, hypothetical).

Experiments that allow a user to choose to organize clauses in order of generation (or reverse order of generation) have been performed. Sorting by order of generation leads to either a breadth first or depth first search strategy. A user can also select an option for clauses to be sorted by number of positive or negative literals, in ascending or descending order. Negative literals correspond to positive properties, and might be preferred in terms of cognitive simplicity. Clauses may also be ordered by total length. Ordering clauses shortest to longest amounts to unit preference with a weighting factor of clause length.

Once answers are generated, a user may select an ordering criterion based on the type of the answer. Since different types of answers are encountered in an interleaved fashion, selecting for particular types of answers seems better suited to a post generation sort rather than a search strategy.

Here are several examples that illustrate preliminary work on search strategies. The first example employs set of support, weighting of clauses to favor the shortest clauses, and unit preference. The rule base is shown, then the query, followed by the original clauses.

```
rachel is a cat
charlie is a tuna
rambo is a rat
rachel likes to eat her catnip toy
tuna are fish
rats are rodents
rachel likes to eat birds
cats like to eat fish
cats like to eat rats
cats like to eat tasty expensive catfood

What does rachel like to eat?
```

```
 1  ((CAT RACHEL))
 2  ((TUNA CHARLIE))
 3  ((RAT RAMBO))
 4  ((LIKESTOEAT RACHEL HERCATNIPTOY))
 5  ((~ (TUNA ?27)) (FISH ?27))
 6  ((~ (RAT ?31)) (RODENT ?31))
 7  ((~ (BIRD ?35))
     (LIKESTOEAT RACHEL ?35))
 8  ((~ (CAT ?5)) (~ (FISH ?7))
     (LIKESTOEAT ?5 ?7))
 9  ((~ (CAT ?13)) (~ (RAT ?15))
     (LIKESTOEAT ?13 ?15))
10  ((~ (CAT ?21)) (~ (TASTY ?23))
     (~ (EXPENSIVE ?23)) (~ (CATFOOD ?23))
     (LIKESTOEAT ?21 ?23))
11  ((~ (LIKESTOEAT RACHEL ?107))
     (ANSWER (LIKESTOEAT RACHEL ?107)))
```

Answers were produced in the following order (only glosses are given):

1. rachel likes to eat her catnip toy
2. rachel likes to eat birds
3. rachel likes to eat rats
4. rachel likes to eat Rambo (the rat)
5. rachel likes to eat fish
6. rachel likes to eat tuna
7. rachel likes to eat Charlie (the tuna)
8. rachel likes to eat expensive, tasty, cat food

This strategy clearly favors succinct answers, which may be extensional (her catnip toy) or intensional (birds).

The second example employs set of support and places new resolvants at the end of the list (in the order in which they are produced). The initial clause ordering is the same as for the first example. Answers were produced in the following order:

1. rachel likes to eat birds
2. rachel likes to eat her catnip toy
3. rachel likes to eat fish
4. rachel likes to eat rats
5. rachel likes to eat expensive, tasty, cat food
6. rachel likes to eat tuna
7. rachel likes to eat Rambo (the rat)
8. rachel likes to eat Charlie (the tuna)

This clearly corresponds to a breadth-first ordering of the answers in terms of the taxonomic structure of the rule base. This technique begins by producing the components of the *complete answer*, that is, those facts and rules that occur at the top of the taxonomy. The complete answer would be represented (showing only the categories) as *birds* **and** *her catnip toy* **and** *fish* **and** *rats* **and** *expensive, tasty, cat food*. This answer implies all extensional and intensional answers.

The third example again has the same initial clauses, employs set of support, but pushes new resolvants onto the front of the list as soon as they are produced. Answers were produced in the following order:

1. rachel likes to eat her catnip toy
2. rachel likes to eat birds

3. rachel likes to eat expensive, tasty, cat food
4. rachel likes to eat rats
5. rachel likes to eat Rambo (the rat)
6. rachel likes to eat fish
7. rachel likes to eat tuna
8. rachel likes to eat Charlie (the tuna)

This corresponds to a depth-first ordering of the answers in terms of the taxonomic structure of the rule base. If the desire is to reach a specific answer as quickly as possible, or to describe a group of objects that are taxonomically related, this is an appropriate search strategy to employ.

There are two important questions regarding ordering of answers. First, how can the control strategy be altered to produce answers in a desirable order, and second, how can answers be ordered once they have all been produced. It may not be known until late in the resolution process whether or not an answer will be subsumed. If answers are returned to a user as they are produced, and later subsumed, it will be confusing. Most of the work in this area has focused on the latter question: how should answers be ordered once they have been produced. It is the former question that is more interesting, and that will be the focus of ongoing work.

Resolution is Useful for studying Question Answering

Using a modified version of resolution provides a good mechanism for studying question answering. The breadth of this approach can be seen by looking at a simple blocksworld problem. As equality is not yet part of the system, some simple assumptions, such as one that specifies that no more than one block can be on top of another, can not be given as rules. This leads to some spurious answers, but also some interesting results.

As simple rule base describing a set of blocks is given as follows:

- A is a block
- B is a block
- C is a block
- A is red
- C is green
- B is either red or green
- a block can not be on itself
- A is on B
- B is on C

The question posed is *Is there a red thing on a green thing?*

A typical resolution theorem prover using an answer literal will come up with two answers, that either B is green and A on B is an answer, or that B is red and B on C is an answer. The modified theorem prover employed here generated over 70 clauses in answering this question, many of which were ultimately subsumed by other clauses. Some of the 19 answers produced (given with their numbers in order of generation, the type of answer, and a gloss) are shown below:

```
1. (hypothetical, extensional - will be
subsumed because C is green is a fact)
if C is green and B is red,
then B on C is an answer

((GREEN C) & (RED B)) =>
(((RED B) & ((GREEN C) & (ON B C))))

3. (extensional)
if a red something is on C, then
that thing on C is an answer

((ON x0 C) & (RED x0)) =>
(((RED x0) & ((GREEN C) & (ON x0 C))))

4. (intensional)
if A is on a green something,
then A on that thing is an answer

((ON A x0) & (GREEN x0)) =>
(((RED A) & ((GREEN x0) & (ON A x0))))

5. (hypothetical/intensional)
if B is not green (this implies B is red),
then if B is on a green something,
then B on that thing is an answer

( (~ (GREEN B))) =>
((ON B x0) & (GREEN x0)) =>
(((RED B) & ((GREEN x0) & (ON B x0))))

6. (hypothetical/intensional)
if B is not red (note that this
implies B is green), then if a red
something is on B, then that thing on B
is an answer

( (~ (RED B))) =>
((ON x0 B) & (RED x0)) =>
(((RED x0) & ((GREEN B) & (ON x0 B))))

7. (hypothetical/extensional)
if B is red, then B on C is an answer
(this subsumes answer 1.)

((RED B)) =>
(((RED B) & ((GREEN C) & (ON B C))))

8. (hypothetical/extensional - will be
subsumed because C is green is a fact)
if B is not green (implies B is red)
and C is green (which is known),
then B on C is an answer
```

```
((GREEN C) &  (~ (GREEN B))) =>
(((RED B) & ((GREEN C) & (ON B C))))

13. (extensional)
Either B is red and B on C is an
answer or B is green and A on B is
an answer

(((RED B) & ((GREEN C) & (ON B C))) or
((RED A) & ((GREEN B) & (ON A B))))
```

Discussion and Future Work

This work has involved making modifications to a simple resolution theorem prover so that not only intensional and extensional answers are considered, but also hypothetical answers in the absence of constraining information.

While the dominant paradigm in theorem proving remains that of providing only extensional answers, it is relatively easy to enhance a theorem prover to provide additional, valuable information in the form of intensional answers. On examining all the resolvants produced during resolution, it is clear that these two categories are not sufficient to describe all of the information generated during the process of resolution. A new category of hypothetical answers was proposed to describe some other potential answers that are generated. As more complex rule bases are developed, the expectation is that further categories of answers may be discovered.

There are many future directions for this work, including the development of search strategies that guide resolution to produce answers with particular attributes *in order* (insofar as it is possible). Researchers have met with considerable success developing search strategies for resolution theorem provers, and the expectation is that this is a fruitful area to pursue related to the generation of answers. Other research interests include providing answers at appropriate levels of detail, and formally defining partial and complete answers.

References

Ali, S. S., and Shapiro, S. C. 1993. Natural Language Processing Using a Propositional Semantic Network with Structured Variables. *Minds and Machines* 3:421–451.

Borgida, A., and McGuinness, D. L. 1996. Asking Queries about Frames. In *Proceedings of KR-96*, 340–349. Cambridge MA: Morgan Kaufmann.

Cholvy, L., and Demolombe, R. 1986. Querying a Rule Base. In *Proceedings of the First International Workshop on Expert Database Systems*, 365–371.

Cholvy, L. 1990. Answering Queries Addressed to a Rule Base. *Revue d'Intelligence Artificielle* 4(1).

Godfrey, P., and Gryz, J. 1996. Intensional Query Optimization. Technical Report UMIACS TR 96-72, University of Maryland at College Park, College Park, MD.

Green, C. 1969a. Applications of theorem proving to problem solving. In Walker, D. E., and Norton, L. M., eds., *Proceedings of the International Joint Conference on Artificial Intelligence*, 219–239. IJCAI.

Green, C. 1969b. Theorem-proving by resolution as a basis for question-answering systems. In Michie, D., and Melzer, B., eds., *Machine Intelligence 4*. Edinburgh University Press. 183–205.

Imielinski, T., and Mannila, H. 1996. A Database Perspective on Knowledge Discovery. *Communications of the ACM* 39(11):58–64.

Imielinski, T. 1987. Intelligent Query Answering in Rule Based Systems. *Journal of Logic Programming* 4(3):229–257.

McCune, W. W. 1994. *Otter 3.0 Reference Manual and Guide*. Argonne National Laboratories.

Motro, A. 1989. Using integrity contstraints to provide intensional answers to relational queries. In *Proceedings of VLDB89, the 15th International Conference on Very Large Databases*, 237–246. Amsterdam The Netherlands: VLDB89.

Motro, A. 1994. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering* 6(3):444–454.

Motro, A. 1996. Panorama: A Database System that Annotates Its Answers to Queries with their Properties . *Journal of Intelligent Information Systems* 7:1–25.

Rich, E., and Knight, K. 1991. *Artificial Intelligence*. McGraw Hill, 2 edition.