A QUESTION ANSWERING INTERPRETATION OF RESOLUTION REFUTATION

by

Debra Thomas Burhans

January 31, 2002

A dissertation submitted to
the Faculty of the Graduate School of
State University of New York at Buffalo
in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy

Department of Computer Science and Engineering

Acknowledgments

A dissertation is the product of not just a single doctoral student, but also of the research and social communities of the student. In my case, this includes, most importantly, Stuart Shapiro. As my advisor, Stu never let up in asking difficult questions and in encouraging me to think and speak carefully and correctly. His exacting standards and expectations inspired me to continually strive to be a better researcher. Stu is an ethical, honest, and thorough scientist; I could not have asked for a better role model. The other members of my committee, William J. Rapaport and Jean-Pierre Koenig, have been extremely important as this dissertation has evolved. Bill's incredible breadth of knowledge and unstinting skills as an editor, both of English and logic, have been invaluable. I have worked through many conceptual problems with his help. JP has provided much-needed expertise in computational linguistics, particularly in the area of semantics. His guidance over the years and suggestions have helped shape the dissertation considerably. I also wish to thank Vinay Chaudhri, my outside reader, for taking an interest in this work and offering his guidance and expertise. His comments and suggestions were extremely helpful.

My research "community" is SNeRG, the SNePS Research Group at SUNY Buffalo. Some of the most important breakthroughs in this work have come from suggestions from my fellow students as well as my committee members at SNeRG meetings. In particular, the hypothetical answers that figure so prominently in this work were determined to be interesting during one of these meetings. SNeRG buddies include Fran Johnson, Haythem Ismail, John Santore, and Alistair Campbell. Other graduate student contemporaries who have helped me along are Chris Egert, Elisabeth Cuddihy, Phil Ventura, and, from my first years of graduate school, Hans Chalupsky, Susan Haller, Robin Hill, and Elissa Feit.

Computer Science faculty members not on my committee who have been helpful and influential are Ken Regan, Deborah Walters, Bharat Jayaraman, Carl Alphonce, Bina Ramamurthy, and Helene

Kershner. Faculty outside my department who have influenced and helped me are Erwin Segal (Psychology), John Corcoran (Philosophy), Mariam Thalos (Philosophy), and Len Talmy (Linguistics). For my first several years of graduate school I worked as a Research Assistant for Rohini Srihari at CEDAR (the Center for Excellence in Document Analysis and Recognition). Rohini sparked my interested in spatial language and helped me develop a lot of my natural language understanding interests and skills. Jeff Siskind, of Purdue University, has been a source of encouragement and inspiration.

The office staff have all been extremely helpful and encouraging, including Ellie Benzel, Jaynee Straw, Sally Elder, Joann Glinski, Jodie Zolnowski, and Lynne Terrana. Sandy Peters and Harry Delano have been great friends and motivators.

Finally, I have to thank my family and friends outside of school. Thanks to Sara and Molly for being patient when my school work took priority, and thanks to Bill for all he has done to make this possible. Virginia Conn has been a great friend, and Kim Cordingly and I, who helped each other through undergraduate school years ago, continue to do so now that we are both working on our doctoral degrees. Thanks to my mom for her unwavering support of my educational endeavors, and to my sisters Mary and Jeanne for all their sisterly encouragement and enthusiasm.

For my family

Contents

List of Figures

List of Tables

Abstract

Early use of theorem provers for question answering identified the presence of an answer with the presence of a proof. Later work recognized other types of answers whose existence was not associated with proof, including intensional and conditional answers. This work is concerned with the problem of defining what is meant by *answer* in the context of resolution theorem proving. Clauses that are *relevant* are all identified as answers, where relevance is determined with respect to a question and knowledge base: any clause descended from the negated question is deemed relevant. This definition of relevance is not in and of itself novel; rather, it is the way in which the set of relevant clauses is *partitioned* that provides the key to interpreting clauses as answers. A partition that divides the set of relevant clauses into three classes, termed *specific*, *generic*, and *hypothetical*, is proposed. These classes are formally distinguished by the way in which literals in a clause share variables, with class membership based on a property termed the *closure of variable sharing of a literal*. The best answers are identified with relevant clauses that are, additionally, *informative*. The *informativeness* of a clause is determined with respect to a set of clauses termed the *answer set*, where an informative clause contains information not conveyed by any other clause in the set. The process of reasoning in search of a proof is recast as the process of constructing a sequence of answer sets, starting with the empty set, where each set in the sequence is *at least as informative* as the previous set. It is this property that identifies question answering as an anytime reasoning process. The *complete answer set* for a given question and knowledge base is shown to be the fixed point of the answer set generation function. While this potentially infinite set may never be realized by a reasoner, the current answer set is guaranteed to contain the most informative, relevant clauses available at any point during reasoning. Different control structures give rise to different answer set sequences, and may be selectively employed to generate preferred answers as quickly as possible. The definition of answer sets elucidates the importance of separating answer search from answer

selection, and highlights the subjectivity of the notion of best answer as defined in many question answering systems. The complete characterization of resolvents as answers presented herein provides a framework that accounts for and expands upon previous work on question answering in a resolution theorem prover. In addition, it provides a foundation for further work in question answering by establishing a context-independent logical pragmatics of question answering.

Chapter 1

Introduction

1.1 Question Answering

Question answering is central to human activity. When we need information, we ask questions that will help us obtain it. Our societal measures of intelligence in large part rely upon question answering. We judge the quality of a Web search engine by its ability to provide useful answers to our questions. Tasks ranging from developing cooperative, multi-agent systems to extracting relevant information from the vast resources on the Internet have as a central requirement the ability to productively engage in question answering.

Within the purview of artificial intelligence (AI), question answering has been approached from a number of different perspectives. Cognitive-science-based approaches to question answering are concerned with trying to simulate human question answering. Problems of natural language understanding and generation, areas at the heart of AI, come to the fore in question answering. Work on open-domain question answering in large databases of documents requires sophisticated linguistic analysis, including discourse understanding and text summarization. The representation of questions and answers and reasoning mechanisms for question answering are concerns of researchers in knowledge representation and reasoning (KR&R). Formal, mathematical approaches to question answering based on logic and theorem-proving form a subset of KR&R approaches.

This dissertation presents a complete, formal characterization of answers in the context of resolution theorem proving. As such, it builds upon previous question-answering work in logic and theorem proving. While the results we present are formal, our interpretation of answer forms is based on work in linguistics and cognitive science. It is not that the form of answers has revealed an intrinsic meaning; rather, an interest in and knowledge of human question answering has led us to recognize novel forms as answers. Further, the approach we present can be viewed as a logical pragmatics of question answering. Broadening the notion of answer leads to a more pragmatic approach to question answering, yet this enhanced pragmatics is based purely on logical form rather than on considerations of context.

1.2 Theorem Proving as Question Answering

More specifically, our exploration of answers takes place in the context of automated reasoning. Resolution refutation, along with many variations and enhancement, is widely employed for automated reasoning tasks [Robinson, 1965, Chang and Lee, 1973, Loveland, 1978, Kowalski, 1979, McCune, 1994]. Resolution plus factoring is refutation complete for first order predicate calculus (FOPC) [Chang and Lee, 1973, Loveland, 1978]: given a consistent knowledge base $\mathcal{K}$ and a proposition p, it can be determined if $\mathcal{K} \cup \{\neg p\}$ is inconsistent. This in turn means that p follows from $\mathcal{K}$.

Automated theorem proving served as an early model for question answering in the field of AI [Green and Raphael, 1968, Green, 1969b, Luckham and Nilsson, 1971]. In this model, a question $\mathcal{Q}$ is represented as a proposition, and a traditional proof is initiated by adding the negation of the clause form of $\mathcal{Q}$ to a consistent knowledge base $\mathcal{K}$. If an inconsistency is found, then $\mathcal{Q}$ has a "yes" answer.

The use of theorem provers to answer questions established an analogy between the notions of "proof" and "answer" that has proved remarkably persistent.[1] We can think of a proof as providing verification of a proposition. In terms of question answering, a verification is appropriate when a question contains only ground terms.[2] For example, if you ask *Did David go to Mary's party?* , and it is in fact the case, a "yes" may suffice. However, "yes" may fall short of being a satisfactory answer when a question contains a variable. For example, an answer to the question *Who went to the party?* might be a list of individuals who meet the criterion *went to the party* or a description

[1] [Green and Raphael, 1968, Green, 1969b, Green, 1969a, Luckham and Nilsson, 1971, Reiter, 1978a, Nilsson, 1980, McCune, 1994, Rich and Knight, 1991, Russell and Norvig, 1995, Luger and Stubblefield, 1997]

[2] A *ground term*, or *fact*, is a variable-free term.

of these individuals. The fact that David went to the party may be used to complete a proof that *someone* went to the party, but the answer provided by a traditional theorem prover under these circumstances would still be *yes* rather than *David*. This lack of informativeness was addressed by modifying theorem provers so that they could return *satisfying individuals*, that is, those ground terms whose substitution for the variables in a question leads to a proof. This type of answer, termed an *extensional* or *specific* answer, is found in most theorem provers today, and is the type of answer returned by Prolog.

Specific answers represent a small fraction of answers we associate with human question answering. One of the problems with specific answers is their failure to capture generality. This may result in lengthy or misleading answers where a simple description might do. Consider again the question about the party-goers from the previous paragraph. It may be the case that *all of the students in Linguistics 581 went to the party*. If there are thirty such students, providing a list of their names is less succinct than the description. The preferred answer depends on the context in which a question is asked, but it is useful to be capable of providing as many types of answers as possible. An answer that embodies general information about a category is termed an *intensional* or *generic* answer. The notion of answer in the context of theorem proving was broadened with the introduction of generic answers,[3] although many theorem provers still do not provide such answers.

In addition to specific and generic answers, some researchers have introduced the notion of *conditional*, or *hypothetical*, answers.[4] These answers provide a contingency that must hold in order for an accompanying answer to be correct. For example, *if John went to Mary's party then David went to Mary's party* is a hypothetical answer. While these answers lie outside the mainstream of question answering, we will show that there is overlap between a subset of the contingencies associated with hypothetical answers and "abducibles" generated by an abductive reasoner.

Resolution refutation involves the repeated resolution of clauses to generate new clauses, termed *resolvents*, until either the empty clause is produced, indicating a proof, or resolution is halted for some other reason. External resource limitations may be imposed to limit a possible unsuccessful search for a proof. By employing an appropriate search strategy, all resolvents descend from the clauses derived from the original question to be answered (or theorem to be proved). All such

[3] [Reiter, 1978a, Nilsson, 1980, Cholvy and Demolombe, 1986, Imielinski, 1987, Motro, 1989, Cholvy, 1990, Motro, 1994, Gaasterland *et al.*, 1994, Borgida and McGuinness, 1996, Godfrey and Gryz, 1996]

[4] [Sergot, 1983, Vasey, 1986, Wolstenholme, 1988, Gilbert, 1988, Demolombe, 1992, Demolombe, 1997]

resolvents can be considered *relevant* to the question or theorem. In terms of question answering, we propose that all relevant clauses are *answers*. This criterion for relevance is not in and of itself novel; rather, it is the interpretation of the entire set of resolvents as answers coupled with the way in which the set of relevant clauses is *partitioned* that provides the key to understanding the theorem-proving paradigm for question answering. The partition we propose on resolvents divides them into three classes corresponding to the three answer classes mentioned above: specific, generic, and hypothetical. This partition allows us to formally characterize each type of answer in addition to facilitating a semantic analysis of the different answer types.

In addition to the property of relevance, we consider the *informativeness* of a clause when evaluating its quality as an answer. While relevance is based on a question and the knowledge base with respect to which it is answered, informativeness has to do with the *questioner*. A complete question answering system clearly must include a user model. Because our focus is on formalizing the notion of answer, we do not consider issues of user modeling. Instead, we judge informativeness at any point during the reasoning process based on the set of answers thus far discovered. This set is separate from the answering agent, and in a sense represents a crude user model: it embodies information that will be shared with the user, and lacks the answering agent's knowledge that enabled the generation of the answers therein. The construction of *answer sets* happens over time as reasoning proceeds. What counts as an informative answer at one time may not at a later time. The fact that the informativeness of answer sets never decreases makes answer set generation an anytime reasoning process. In an anytime reasoner results are available at any point during the reasoning process, not just at the time of completion or cessation. The control structure employed by a reasoning agent determines the structure of the intermediate answer sets. Different control structures may be viewed as selectively searching for particular types of preferred answers, though their exclusive production is not guaranteed. Each question and knowledge base is associated with a complete answer set, which is the fixed point of the answer set generation function. This set may not be realized by a reasoner, and is not necessarily the goal of question answering.

Our consideration of reasoning as answering in time naturally leads to a computational model of answering. In the model, answering is divided into four subtasks: answer search, answer set generation, answer selection, and answer communication. There is no *a priori* notion of sequential execution of these tasks, though there are obviously some dependencies among them: without answers, a set of answers may not be constructed. Similarly, unless there is an answer set, there is nothing from which to select an answers. Communicating a selected answer obviously depends on an answer choice. This model embodies our approach to question answering, where preferences for particular types of answers do not prevent the discovery of undesirable answers, yet it also captures the way in which most question-answering systems operate. It is typical for a question answering system to find all answers, then order them, possibly discarding some, and finally return all the answers or a selection from among them. Implicit in the control structure of most question answering systems is a selection strategy that results in bypassing many results we deem answers. This occurs in these systems because the search process is imbued with selection criteria. Note that Prolog allows for sequential answer discovery, one answer at a time, but this is related to the association of answers with proof, and to the fact that all answers found by Prolog are at the same level of generality (e.g. all are specific answers). The purely sequential approach to question answering is popular due to the fact that most question answering systems in the theorem proving domain restrict their clauses to function-free FOPC. When the search space is finite, finding all answers may be a reasonable first step. However, given an infinite search space, an approach that returns answers as they are generated is the only approach that will yield success. Luckham and Nilsson, unlike most others working on question question answering in a logic paradigm, allowed function symbols in the knowledge base and in answers. However, since all of their answers are associated with proof, including those containing variables and function symbols, the presence of an answer by definition is associated with a proof; thus, with a cessation of search. Allowing for a potentially infinite number of answers necessitates the development of a model of question answering that can return a partial set of answers.

Interpreting resolution refutation as question answering does away with the analogy between proof and answer. Every reasoning step produces an answer, and sometimes a reasoning step happens to be that which represents the culmination of a proof. In such a case, we know the precise form of the answer generated, but do not view it as the only possible answer. A failed proof attempt is no longer a failure; rather, it is associated with some answer set. A proof does not necessarily signal the production of the complete answer set. The notion of failure in terms of question answering is an empty set of relevant clauses: this is only the case when no clause in a knowledge base may be resolved with those emanating from an original question.

1.3 Motivation

It is clear that the centrality of question answering to the issue of intelligence makes it a worthwhile topic for any AI researcher. However, theorem-proving approaches to question answering have long since been abandoned by researchers in favor of other formalisms. Question answering in the context of theorem proving is many layers removed from the complexity of human question answering, and FOPC fails to provide the representational richness of natural language. What, then, is the motivation for a formally limited investigation of question answering? Despite efforts at many levels of investigation, there is still no consensus as to what constitutes an answer. There are clearly answers that we provide daily in conversation that are not represented by question answering systems. Even at the level of theorem proving, the picture remains incomplete. The purpose of this investigation is, therefore, to expand upon proposed logical formalisms for question answering from the foundation of theorem proving. When the complexity of natural language is set aside, and the well-understood formalism of FOPC as a representation language is employed to its full extent, the question of what constitutes an answer is more easily investigated. The discovery, even in this simplified world, of a cohesive, complex definition of answer that expands upon that which is currently accepted is extremely valuable.

The following observations served to motivate our work:

1. Question answering systems generate and discard a considerable amount of information that is relevant to questions en route to a preferred answer, should such an answer exist. In the absence of an answer, most systems return nothing, even though this "ignored" information may be relevant to the question and could lead the questioner to an answer.

2. Judgments about the informativeness of answers is generally based on the ontology of an answering agent. A questioner may have a very different ontology, meaning that which seems uninformative to an answering agent may in fact be very informative to a questioner. If one answer taxonomically subsumes another, the subsumed answer is typically viewed as uninformative: an exception to this is question answering in the context of description logics, where answers at different levels of generality may be offered. Information about how answers are related to one another exists in the mind of the answering agent and not necessarily in the mind of the questioner. This implies that the level of generality/specificity of an answer from

the point of view of the answering agent may not match that of the questioner. A failure on the part of an answering agent to provide a questioner with information at different levels of generality may result in a failure to satisfactorily answer a question.

3. What counts as an answer in one question answering system may be very different from what counts as an answer in another.

We address the problem of "lost" relevant information by broadening the definition of answer to include all information that is relevant to a question. This seemingly coarse filter is mitigated by considering answer *informativeness*: only informative answers will be used in generating question responses. The model we develop allows further filtering based on a user model.

The second issue is concerned with regarding answers at different levels of generality equally in the absence of a complete model of the questioner's knowledge base. Our definition of informativeness with respect to an answer set ensures that this problem does not arise. The answer set contains only answers, and lacks rules from the answering agent's knowledge base that taxonomically link answers and deem that one subsumes another. Clearly, an answering agent is more informative when it conveys taxonomic information that relates answers to one another, but this is a separate issue from collapsing an entire taxonomic hierarchy based on subsumption.

The problem of different systems providing different types of answers is related to the central question addressed by this dissertation: "what counts as an answer?" Question answering systems implicitly encode their answer definitions in their control structures. We provide a formal, explicit definition of what it means to be an answer in the context of resolution refutation. Our proposed partition of answers into the classes specific, generic, and hypothetical is broad enough to encompass all previous work on question answering in the context of resolution refutation. By cleanly delineating these three answer classes, we can compare resolution-based question answering systems according to the answers a system recognizes, as well as those it fails to recognize. The limitations of a system in terms of the types of answers it provides may now be viewed as a function of its control structure rather than an inherent inability to generate different types of answers. We make no claim that we have provided a definition for *all* possible types of answers; rather, we have provided a complete picture for answers within a representational formalism of full FOPC. Further partitioning of the classes we define may be important, and we anticipate the discovery of additional answer classes in other formalisms.

Finally, any research concerning a topic related to language understanding ought to make clear its relationship to syntax, semantics, and pragmatics. In restricting this investigation to FOPC, a formal language whose syntax and semantics are well understood, the less cleanly defined boundary of semantics and pragmatics, if indeed it is a boundary, is avoided. The significance of this work in relation to natural language is the following: working from a purely syntactic form, a partition of clauses is discovered whose parts map cleanly onto three distinct semantic answer classes. In addition, the entire program can be viewed as a pragmatic approach to question answering. This pragmatics is achieved by being as informative as possible in a timely fashion, and is achieved in a purely logical fashion.

1.4 Research Contributions

The major theoretical contributions of this research include:

- A complete characterization of resolvents as answers
- A formal definition of three answer classes: specific, generic, and hypothetical
- A clear demonstration of the importance of all three answer classes, including their relationship to natural language
- A detailed characterization of the class of hypothetical answers
- A clear distinction between generic and hypothetical answers
- A canonical form for answers
- A formal specification of an answer set and a procedure for constructing answer sets
- A characterization of the complete answer set for a given knowledge base and question
- A proof that the complete answer set is the fixed point of the answer set generation function
- A reasoning-based model of answering that separates answer search from answer selection
- A logical approach to question answering that embodies a context-independent pragmatics

1.5 Dissertation Outline

This introduction has served to outline and motivate the research. Chapter 2 presents a review of relevant literature. Chapter 3 describes a reasoning-based model of answering. In Chapter 4 a complete exploration of answer search in a resolution theorem prover is provided. The answer classes *specific*, *generic*, and *hypothetical* are introduced and characterized. Chapter 5 addresses the process of generating answer sets. The *informativeness* property of answers is defined, and its relation to answer set generation is clarified. A procedure for generating sequential answer sets is explicated, and the *complete answer set* for a given knowledge base and question is shown to be the fixed point of the answer set generation function. A discussion of desiderata for answers is included, focusing on taxonomic relationships among answers. Chapter 6 presents an overview of our implementation, and Chapter 7 discusses broader applications of our work, including the applicability of the results to natural language understanding as well as to other reasoning systems. Chapter 8 concludes with a summary and list of future research directions.

Chapter 2

Background

In the early days of AI research, "question answering systems" would have fallen squarely into mainstream AI. With the divergence of AI into a number of subareas, different aspects of question answering migrated into various niches within AI (e.g., user modeling and deductive databases) as well as into other areas of computer science entirely. Specialized conferences such as *Flexible Query Answering Systems (FQAS)* and *Knowledge Discovery and Data Mining (KDDM)* have attracted a number of researchers interested in question answering, while large, more general AI conferences have seen a dearth of sessions devoted exclusively to the topic. The importance of question answering to information retrieval is clear, from databases to unstructured text. The importance of large knowledge bases and databases has been steadily growing, with the Human Genome Initiative providing just one example of a project whose success depends on the ability of researchers to *ask the right questions* about data.

Recently there has been a renewed interest in question answering within the AI community, due in part to the TREC (Text REtrieval Conference) initiative. For the past several years, TREC has sponsored a "QA Track" (Question Answering) that has engaged the efforts of a number of AI researchers. The American Association for Artificial Intelligence (AAAI) sponsored a 1999 Fall Symposium devoted to question answering, and the 2001 International Joint Conference on AI (IJCAI) included a question-answering tutorial. The resurgence of interest in question answering is related to the plethora of data that is available electronically, primarily on the World Wide Web (WWW). More importantly, this data is both multimodal and highly unstructured: making use of it depends upon an extremely sophisticated question answering agent with capabilities in natural language understanding and computer vision. You can't expect search engines to do a good job of retrieving *relevant* documents if they can't, at least at a rudimentary level, *understand* those documents.

The examination of background work in this chapter focuses on a number of different aspects of question answering. The roots of question answering in AI are found in automated reasoning systems. Automated theorem provers provide the basis for our work, in particular, resolution theorem provers. A general overview of resolution refutation and its use for question answering is provided in order to place relevant background material in context. In addition to a general discussion of automated reasoning, we demonstrate an important connection between abductive reasoning and hypothetical answers.

Theoretical work in question answering is found in AI, psychology, linguistics, and philosophy. Work in these various areas forms the basis for implemented question answering systems. Among theoreticians, there is no general agreement as to what constitutes an answer, and there is scant mention or recognition of the answers we term *hypothetical*.

Important question answering *systems* are discussed and their capabilities compared. There are systems that, with no additional requirements in terms of reasoning, could provide all of the types of answers described in this dissertation, yet their definition of answer is narrowly construed in keeping with the prevailing definition of *yes* or a list of *satisfying individuals*. Some systems provide generic answers or rules: description logics in particular provide the capability of returning object descriptions, or concepts, as answers. There are a few systems that recognize and return hypothetical answers. No single system embodies all of the question answering features presented in this dissertation.

2.1 Resolution Refutation

Resolution refutation is an efficient procedure for establishing the truth of a proposition in FOPC. The failure of a proposition to be true can not, in general, be established: FOPC is semi-decidable. Resolution plus factoring is a refutation complete proof strategy for FOPC.[1] If an inconsistent clause

[1]Factoring allows for unification of literals within a clause, and without it resolution is not refutation complete. The term *resolution*, however, is generally used to refer to *resolution plus factoring*.

is added to a set of consistent clauses, the inconsistency will be detected using the resolution procedure. The following brief overview of resolution is based on Russell and Norvig [1995, pages 277–286].

We assume that there is a set of well-formed formulas (wffs) in FOPC that represent the rules and facts with which reasoning will proceed. Resolution is defined for wffs in clause form, so converting from standard FOPC to clause form is the first step. Clause form is presented in Davis and Putnam [1960]. The following steps from Russell and Norvig describe a procedure for converting a wff into clause form.

1. Implication elimination: eliminate all $\rightarrow$ symbols using the fact that $a \rightarrow b$ is equivalent to $\neg a \vee b$.

2. Mini-scope negation: move negation symbols in so that they apply to single terms by using deMorgan's laws, the fact that $\neg(\neg p) = p$, and the standard quantifier equivalences ($\neg\forall x = \exists\neg x$, and vice versa.)

3. Standardize variable names so that each quantifier binds a unique variable.

4. Move all quantifiers to the left of the formula without changing their relative order. Once this has been done the formula is in what is known as *prenex normal form* (*PNF*), comprising the quantifiers followed by a quantifier free *matrix*.

5. Skolemize: eliminate the existential quantifiers by replacing the variables over which they quantify with Skolem functions of zero or more variables. The variables in a Skolem function are those that are universally quantified where the existential quantifier in question falls within the scope of the universal quantifiers. A Skolem function of zero variables is generally called a Skolem constant.

6. Drop the universal quantifiers: at this point all variables in the matrix are universally quantified.

7. Distribute $\vee$ over $\wedge$, so that $(a \wedge b) \vee c$ becomes $(a \vee c) \wedge (b \vee c)$.

8. Put the formula into *conjunctive normal form* (*CNF*) by flattening nested conjunctions and disjunctions. A formula is in CNF if it is expressed as a conjunction of disjuncts.

9. Create a separate clause corresponding to each disjunct. Each clause comprises the set of literals from one of the disjuncts. Literals within a clause are implicitly assumed to be disjoined. The set of clauses has now replaced the original FOPC formula, and clauses are implicitly assumed to be conjoined.

Russell and Norvig take an additional step to express each disjunct as an implication, placing the negative literals as the antecedent and the positive literals as the consequent. This form is logically equivalent to CNF, and is termed *implicative normal form* by Russell and Norvig, who note that the term is not standard. In this dissertation we employ CNF rather than implicative normal form.

The rule of resolution may be applied to two clauses that contain complementary literals that can be unified. Unification involves finding a substitution that makes the two literals look the same. While an infinite number of possible substitutions may exist, a *most general unifier* is selected and applied to both clauses in question. The complementary literals are deleted, and the rest of the literals in the two clauses are collected into the new clause.

Resolution refutation is initiated when the negation of a proposition that expresses a question or potential theorem is added to a consistent set of clauses. Clauses are resolved until the empty clause is produced. The empty clause indicates a contradiction. Because the original set of clauses is presumed consistent, this contradiction must have arisen from the negation of the starting proposition. Therefore the original proposition follows from the set. In case of a question, a contradiction indicates a "yes" answer, and in case of a potential theorem, it indicates a proof.

The following example shows a simple resolution refutation. Consider the following knowledge base, with facts and rules shown in standard FOPC and clause form along with an English gloss.

1. $\forall x(cat(x) \rightarrow animal(x))$ $\{\neg cat(x),\ animal(x)\}$ cats are animals

2. $cat(Fred)$ $\{cat(Fred)\}$ Fred is a cat

3. $cat(Dave)$ $\{cat(Dave)\}$ Dave is a cat

4. $calico(Rover)$ $\{calico(Rover)\}$ Rover is a calico

The question to answer is $\exists y\ animal(y)$, the negation of which is shown below:

5. $\forall y\ \neg animal(y)$ $\{\neg animal(y)\}$ Nothing is an animal

A refutation follows. The new clause, or resolvent, is listed first, followed by the numbers of the clauses that have been resolved and the substitution employed.

6. $\{\neg cat(z)\}$ R 1,5 $\{z/x, z/y\}$

7. $\{\}$ R 2,6 $\{Fred/z\}$

The empty clause has been generated, thus the question has a "yes" answer: there is something that is an animal. In this case the "something" is the individual *Fred*, and the corresponding specific answer is *Fred is an animal* .

2.2 General AI Approaches

Question answering from a traditional AI point of view has been relatively narrowly focused on the task of searching for and returning as answers *individuals* that satisfy a query. This type of answer is what we term a *specific answer*. Consider this surprisingly current description of question answering, written by J. L. Kuhns [1967, page *v*] more than thirty years ago:[2]

> The problem of computerized question-answering is seen to involve a two-step procedure: first, transforming the question into a certain sentential formula of the predicate calculus (called the symbolic question); second, generating the answer by calculating (what we shall call) the value set of the transformed question.
>
> The main purpose of the present work is to investigate the second, or answering, process. The process is to output the answer by acting on the symbolic question and the data base for the question-answering system. The data base consists of a dictionary of descriptive names together with a file of elementary sentences. The value set of a symbolic question containing free variables (e.g., those which stem from natural-language questions such as 'What books has Scott written?') is the list of names which when substituted for the free variables yield a "true" sentence on the data base. For a symbolic question without free variables (e.g., one which stems from natural-language questions such as 'Did Scott write Waverly?') the value set, or simply the value, is an expression indicating the truth value.

Simmons [1965, 1970] reviewed a number of early question answering systems developed by AI researchers. In his 1965 article, five of the fifteen systems reviewed were classified as *Logical Inference Systems*: *The Darlington Logic Programs* (Darlington); *The Cooper System* (Cooper); *The Specific Question Answerer (SQA)* (Black); *The Semantic Information Retriever (SIR)* (Raphael),

[2]Underlining in the text is from the original.

and *Student* (Bobrow). These systems were developed before the publication of Robinson's seminal paper, "A machine-oriented logic based on the resolution principle" [1965], and bear less resemblance to the work in this dissertation than later, resolution-based systems. The *Baseball* system is noted (under List-Structured Data-Base Systems) for its logical processing achievements [Simmons, 1965], where logical processing was considered a third phase in question answering. The first two phases of question answering were identified as syntactic and semantic analysis, respectively, which reflects the strong interest at that time in language understanding and machine translation. As far as questions and answers were concerned, translation in and out of English was of greater interest than the logical inferences required for answer discovery.

In a follow-up article five years later, Simmons [1970] devotes a section to "Inference in Question Answering", opening with the statement [1970, page 27], "After a question and an answering text have both been translated into an explicitly structured formal language, the process of question answering can be seen to be essentially one of theorem proving." Simmons cites the work of Green and Raphael [1968] as a use of the Robinson resolution algorithm that shows promise in the area of question answering. It is the work of Green and Raphael that is truly the starting point for the use of resolution theorem provers as question answering systems. Regarding the form of answers, Simmons notes that answers are identified with lists of names. These names are what we have referred to previously as satisfying individuals, and are identified with specific answers. They are "satisfying" because they represent variable substitutions that lead directly to a proof. The names in a list identified with an answer may be substituted in turn for variables in the original question, yielding a set of ground atomic formulas that are referred to as "facts". Thus, specific answers may be referred to as lists of names or lists of facts. With the emergence of theorem-proving as a dominant paradigm for question answering, specific answers became the focus of question answering systems.

The *answer is a list of facts* dogma has proven remarkably persistent in mainstream AI. Presentations in standard textbooks on AI [Rich and Knight, 1991, Ginsberg, 1993, Russell and Norvig, 1995, Luger and Stubblefield, 1997] perpetuate this view. Allen's book on natural language understanding refers exclusively to the retrieval of facts in its discussion of question understanding [Allen, 1995]. In Russell and Norvig's (1995) text on artificial intelligence, there is no mention of replying to queries with anything other than facts in their discussion of query processing.

Early deductive question answering systems that provided *specific answers* to questions include

DEDUCOM [Slagle, 1965], which was capable of retrieving relevant facts in response to questions. DEDUCOM could increase its knowledge through interaction with a human: Slagle notes that a question answering program may be a good candidate for a program that can learn. The identification of self-questioning with learning was later made by Ram [1991], among others.

BLAH [Weiner, 1980] was a question answering system that could explain its reasoning. In particular, BLAH was capable of computing and describing what we call hypothetical answers. In the face of a failure to prove X, not only would BLAH construct possible explanations for X, but also for $NOT\ X$. This capability is discussed in a section of the paper on counterfactuals, which are deemed an interesting phenomenon that a question answering system should be capable of handling. The following is an apropos quote [Weiner, 1980, page 36]: "When the user requests the system to (SHOW X), and X is not provable given the current state of the data base, BLAH assumes that the user wants to know not only why (NOT X), but also under what circumstances X could be proved." Hypothetical answers are those that describe the circumstances under which X could be proved.

Prolog, a logic programming language, has been employed in a number of AI systems. It provides a query answering feature that accords with the view expressed by Kuhns [1967]: an answer is a list of ground terms ("names", in Kuhns's and Simmons's terminology), or a "yes". Negation as failure [Clark, 1978] allows the failure to find an answer to be interpreted as a "no". Prolog as a knowledge representation and query language is discussed in Dahl [1983] and Sterling and Shapiro [1999], among others, while the foundations of logic programming are described in Lloyd [1987]. Prolog in particular is noted for its restricted and efficient logic-based reasoning. It is interesting to note that, given the right sort of information, Prolog *will* return a "non-standard" answer that represents a subtype of generic answers. Consider the following simple rule base, shown in Prolog with English glosses:

- likes(X,Y) :- cat(X).

  Cats like everything.

- cat(rachel).

  Rachel is a cat.

And the following query:

- likes(rachel,X).

What does Rachel like?

If these Prolog statements are entered in Prolog, the result of the query is, surprisingly,

`X = _8183` (`_8183` *is a variable*)

We might write the full answer, in Prolog form, as:

```
likes(rachel,_8183)
```

Note that the variable `_8183` can match any term. Thus, `likes(rachel,_8183)` means that Rachel likes everything. Our term for this type of answer is *unconstrained generic*. In contrast to answers traditionally considered as generic, the universally quantified variables in this answer have no restrictions on them: they match all known individuals. The reason this answer appears at all is because there are *no* known individuals in the rule base that match the query. Unconstrained generic answers are associated with proof, thus will be available in typical AI question answering systems.

If we add to this rule base a statement about an individual that Rachel likes, the result is interesting. Note that nothing has been removed from the rule base. The rule that cats like everything and the fact that Rachel is a cat are still there.

```
| ?- consult(user).  | likes(rachel,bugs).  | ^D
% user compiled in module user, 0.000 sec 56 bytes

yes | ?- likes(rachel,X).

X = bugs ;

no
```

The unconstrained generic answer has vanished despite the fact that the rule that cats like everything is still in the rule base. Prolog, as a typical theorem prover, does not recognize the need to treat generic and specific answers separately; thus, `bugs` is simply bound to `X` and the proof is complete. This demonstrates why, in general, traditional theorem-provers return specific answers. What is interesting in this case is that there is a proof that produces a generic answer, yet it is not recognized by Prolog. Generic answers that are not unconstrained are associated with intermediate results of reasoning rather than proof. It is the *interpretation* of these intermediate results as *answers* that renders them interesting, not the role they play in finding a proof. These intermediate results are produced and discarded when the focus is on proof. In principle, this information is not out of reach, and might be retrieved with simple alterations to a theorem prover. In the case of Prolog, intermediate reasoning results can be returned by a meta-interpreter.

Otter [McCune, 1994] is a powerful theorem prover developed at Argonne National Labs. There is nothing to distinguish Otter from other theorem provers in its lack of ability to provide anything but specific answers. Its purpose is, after all, to derive proofs in the most efficient manner possible. If Otter were capable of returning generic and hypothetical answers, they might prove useful in error detection in input files. When a proof attempt fails, generic answers provide descriptions of objects that will allow a proof to succeed. What is lacking are the names of these objects, and having a description may alert a researcher to a missing piece of information or a generality not previously foreseen.

SNePS [Shapiro, 1979, Shapiro and Rapaport, 1987, Shapiro and Rapaport, 1992] is a semantic network processing system that has been used in a number of knowledge representation and natural language understanding tasks. In SNePS, rules and propositions are terms in the representation language. Thus, SNePS variables may match any node in a SNePS network, including rules, individuals, and propositions. In SNePS, a specific answer is associated with a base node, which represents a simple individual. Reduction inference in SNePS allows for the retrieval of partial concept descriptions, which is analogous to some of the work described in the context of question answering in description logics [Borgida and McGuinness, 1996]. Thus, SNePS, which is not a standard theorem prover and lacks many of the restrictions that accompany such systems, has a relatively broad answering capability due to its flexible representation and reasoning abilities. In "SNePSLOG: A Higher Order Logic Programming Language" [Shapiro *et al.*, 1981], SNePS's capability for representing rules that quantify over predicates, rules that have rules as their antecedents, and meta rules (rules about rules) is demonstrated. SNePSLOG is a formal language that serves as a logic programming interface to SNePS. It can be considered a higher order logic because of the structures it can build and the entities to which it can refer. Shapiro *et al.* [Shapiro *et al.*, 1981] demonstrates the use of generic rules in question answering, where rules may be used "as data". In response to a question about whether the ship the "Flying Dutchman" has a cruise speed, the system initially asks itself whether *all ships have a cruise speed* and uses this information in answering the question: *since all ships have a cruise speed, the Flying Dutchman has a cruise speed* [Shapiro *et al.*, 1981, page 12]

ANALOG [Ali and Shapiro, 1993, Ali, 1994] is based on SNePS but embodies in its representation Fine's theory of arbitrary objects [Fine, 1983]. In ANALOG, specific and generic objects are represented using the same underlying structure, and are related to one another by subsumption. Objects in the same class lie along a common branch of a subsumption taxonomy, facilitating retrieval algorithms that need to "move up and down" the taxonomy. While this is similar to a description logic such as LOOM [MacGregor and Bates, 1987], description logics tend to separate concepts (generic objects) and instances (specific objects) into different components of the knowledge base. Question answering in ANALOG allows for specific and generic answers, all of which are related by subsumption. While general answers subsume specific answers in the same branch of the taxonomic hierarchy, answers at different levels of generality are all considered interesting, as demonstrated in the following example [Ali, 1994, page 99], taken from a section in Ali's dissertation titled "Rules as Answers to Questions". The system is given the following information:

All men are mortal

John is a man

In response to the question *Who is mortal?* , the response is *John is mortal and all men are mortal*. This response includes both a specific and a generic answer.

Ashwin Ram's work focuses on a theory of *questions* rather than answers [1991]. His particular interest is in how an agent poses questions to *itself* in order to learn new things. We have not focused on the learning aspect of question answering, and we make the assumption that the questioner and answerer are not one and the same. There is an important relationship between our work and that of Ram, however. The class of hypothetical answers that we describe involves answers that have two parts. The first part is, effectively, a question that the answering agent does not (yet) have the ability to answer. The second part describes an answer that is *contingent* on the first part being true, or in terms of question answering, having a "yes" answer. The process whereby an answering agent attempts to resolve away the portion of a clause associated with a contingency may be viewed as self questioning.

The question answering system that comes closest to our work is CONVERSE [Klahr *et al.*, 1980]. The desiderata expressed for this system include the ability to return *concrete expressions*, *rules*, and *conditional answers*. These answer categories correspond approximately to what we term specific, generic, and hypothetical answers, though the overlap is not exact. The following quote summarizes the expectations regarding conditional answers [Klahr *et al.*, 1980, page 134]: "Ability to generate and display incomplete derivations is an aid to identifying missing information and giving conditional answers." Klahr *et al.* recognized the value of conditional answers as a means of identifying missing information. The focus in the CONVERSE system was on efficient processing and making use of non-standard answers in order to provide answers more quickly. To this end, the research focused on minimizing the length of deduction chains and developing efficient representations of paths of deduction. Differences between this work and our endeavor include the focus and the depth of investigation of answers as a whole. Our focus is on the formal characterization of answers and their relationships to one another rather than on the deductive process. Klahr makes the important observation that, while completeness is important for mathematical problems, it is not a primary consideration for question answering. This is clearly a point that sets question answering apart from theorem proving. While some researchers in question answering accept incompleteness, evidence for the unwillingness to let go of the requirement for complete reasoning systems is found in current papers on question answering systems [Motro and Rakov, 1997] as well as in comments we have received on our work.

2.3 TREC

TREC is the Text REtrieval Conference organized by the National Institute of Standards. For the past two years, TREC has included a question answering track (QA Track) whose challenge to researchers is to find answers to open-domain questions using several gigabytes of unformatted text data [Voorhees and Harman, 2000, Voorhees and Harman, 2001]. Research in the TREC competition involves the categorization of questions into types such as "who", "how many", etc. These types are largely based on syntax and do not reflect deep semantic understanding. Each type of question is associated with a desired type of answer, where answers may range from templates with slots to fill in to descriptions of answers. Various techniques are used to perform retrieval of texts likely to contain an answer. Some groups employ a Web search engine, while at the other extreme some groups have indexed the entire text database [Woods *et al.*, 2000]. Success rates for answering the questions in the challenge, which includes several hundred questions, are close to 80% for the long answer competition, where the answer must be contained in a 250-byte window of text. Scores for the short answer competition, where the answer must be found within 50 bytes of text, are about twenty percentage points lower. Some of the more successful systems include those of Harabagiu and Moldovan [Moldovan *et al.*, 2000, Harabagiu *et al.*, 2001], which employ sophisticated answer analysis involving syntax, semantics, and pragmatics. Their FALCON system [2001] verifies that answers are correct using an abductive procedure introduced by Hobbs *et al.* [1993]. Their verification step allows them to search for more answers if those retrieved are not satisfactory, boosting their accuracy and making their system generally more robust to text search engine shortcomings. Also of note is the system presented by Cymfony, Inc. [Srihari and Li, 2000], which achieves 66% accuracy employing an information extraction engine: deep semantics is not involved. The interest and focus in TREC is on specific answers. The 2000 competition (TREC-9) included, for the first time, a number of definitional questions, such as *what is a horse?* . It is particularly notable that these questions proved to be the greatest challenge to participants: such questions require generic answers. Thus, the work in this dissertation may inform the text retrieval effort in terms of presenting new answer types and forms, in particular, generic and hypothetical answers, where hypothetical answers have yet to be considered by the text retrieval community.

2.4 Knowledge-Based Querying

Recent work in AI has focused on what is loosely termed "knowledge based information retrieval". The High Performance Knowledge Bases effort (HPKB) might be considered as part of this, and involves a number of research groups around the country. Knowledge-based approaches to question answering are often contrasted with efforts in information retrieval, but, as the TREC competition has demonstrated, both are pieces of the same puzzle, and both are important in building intelligent question answering agents.

A recent paper from a group at SRI International begins by proposing desiderata for question answering systems that accord with the approach of expanding the notion of answer [Starr *et al.*, 1998, page 18-1]:

> Innovative query interfaces to knowledge and database systems must go beyond simply returning the requested information. They must be capable of producing intensional answers when a description improves the understanding of an answer ..., producing conditional answers when no one answer matches the conditions of a query, and using ontological information in processing a query."

This paper [Starr *et al.*, 1998] points out that there are no query processing systems capable of returning conditional answers. An example they cite for conditional answering involves a disjunctive answer, a classic case of a lack of information. The general form of their disjunction is *if A is the case then C is the answer, otherwise if B is the case then D is the answer*. This type of answer is clearly important in the defense domain, where most of their work is applied. It may not always be possible to know what a particular military or political scenario is, and the ability to hypothesize about it and draw relevant conclusions is at the heart of hypothetical answering.

Another group working in the area of knowledge-based question answering has developed a robust question answering system that has been used in several different domains, from tutoring to augmenting on-line manuals [Clark *et al.*, 1999]. Their goals include the ability to perform well in answering previously unanticipated questions, which is a focus of TREC and clearly an important, yet difficult, aspect of any question answering system today. They note the need for hypothetical reasoning in order to answer "what if" questions. If a hypothetical scenario holds, what follows from it? This is somewhat the reverse from the role of a hypothetical answer, which is a potential cause of a known effect rather than the other way around. Their efforts have focused on developing a robust, portable architecture for question answering rather than on categorizing in a more theoretical manner the types of answers that might be provided, but they do discuss definitional and intensional answers as system desiderata that early question answering systems hard coded rather than generated in real time.

Finally, Chalupsky and Russ's work on the "WhyNot" querying tool for PowerLoom [Chalupsky and Russ, 2002] is intimately related to our work on hypothetical answering. The "WhyNot" tool is capable of returning answers found "along the way" to a (possible) proof. Such answers have the form of rules: the definition of these rules is broad and encompasses both generic and hypothetical classes of answers as we present them. These answers may be viewed as the results of "partial proofs", though at the time they are generated it is not known whether they in fact lie on a reasoning path that leads to a proof. Of particular note in this work is that they have implemented "WhyNot" on a version of Cyc [Lenat, 1995] containing over 1,000,000 facts and 35,000 rules. The name of the system reflects the fact that it is capable of providing explanations when traditional answers are not found. Explanations take the form of hypothetical answers. Preliminary results show a plethora of hypothetical answers generated: part of their work is focused on limiting the size of the set of such answers, including restricting the number of "unprovable" hypotheses, allocating small time slices to the exploration of hypotheses, and reordering clauses to help limit the search space. The problem of limiting the number of hypothetical answers remains an open area of research.

2.5 Cooperative Answering Systems

Kuhns's description of question answering clearly does not reflect the way people answer questions. The field of *Cooperative Answering* is concerned with making computational agents better at answering questions: most cooperative answering systems incorporate ideas about human cognition. They typically are able to provide general, descriptive answers along with explanations about their answers (or lack thereof). Gricean maxims of conversation [Grice, 1975] are frequently used as the basis for designing cooperative answering systems. These maxims describe various ways in which speakers engaged in a cooperative conversation expect one another to behave. For example, the maxim of quantity states that you should say enough, but not too much. In describing the maxims, Grice effectively codified commonly employed rules of conversation. An excellent overview of cooperative answering can be found in an overview of the field by Gaasterland, Godfrey, and Minker [1994].

Early work in cooperative answering was done by Lehnert [1978, 1981]. Her question-answering system, QUALM, was one of the earliest that provided cooperative answers. Lehnert developed a theory of questions and answers as well as a model of question answering. She defined a number of different classes of questions, and showed for different question types what sort of answer was most appropriate. According to Lehnert's question ontology, a question like, "What objects X have property Y?" would be classified under *Concept Completion* or *Feature Specification*. "Does object X have property Y?" represents a *Verification* question. Both questions are informational; the questioner uses them to gain information from an agent about its knowledge (as opposed to its feelings, intentions, etc.). The other ten question categories described by Lehnert are not relevant for this investigation, which is focused on *information gathering* as prompted by questioning.

In "The Process of Question Answering" [1978, pp.107–108], Lehnert says that the processes of finding and formulating an answer can be divided into two parts: content specification and memory search. Content specification "determines what kind of answer should be produced and how to look for it", while memory search is simply the process of actually locating the information. Once an answer is found, a final process is to generate a natural language response. Lehnert's definition of the process of question answering exemplifies the conflation of preferences with search. This type of approach is destined to fail to return relevant information in case a narrowly specified answer cannot be found. There is no need, in principle, to intertwine search and preferences, and the result of doing so is to narrow the possible answers that may be found. The principle in our approach is that it is better to provide some information even if it falls short of a desired response, thus, deferring the influence of preferences to the time when answers are selected, and even then, ignoring preferences if they can not be satisfied.

Joshi and Kaplan describe intensional answers as important in human language and desirable in machine responses to questions [Joshi and Kaplan, 1977]. In addition to intensional answers, they describe cooperative answers that indicate possibility, "maybe", a general lack of knowledge, or "I don't know". Kaplan [Kaplan, 1981, Kaplan, 1984] continued this work in CO-OP, a system that provided *corrective indirect responses* to questions based on mistaken presuppositions on the part of a questioner. Kaplan represents the information from a question in a graph, viewing each category name or modifier as corresponding to a set of entities. It is useful to examine an example from [Kaplan, 1981]. The question "Which students got Fs in linguistics courses?" is posed. The data indicate that no students received Fs, and that there are in fact no linguistics courses. Kaplan's system generates this response: "I don't know of any Fs and I don't know of any linguistics courses." Kaplan's system was capable of returning rules about the data rather than lists of specific entities: intensional answers. This was done using heuristics that checked the length of the list of entities that constituted a question response. If it were longer than a certain upper bound, an intensional statement about the data would be returned instead of the list. For example, if every student who had taken a linguistics course had received an F, and there were hundreds of these students, the response, "all students in linguistics courses received Fs" would be generated rather than a complete listing of the student names. These generic statements were not generated on the fly but were part of the static data structure. He discusses the importance of being able to answer questions in this manner, and cites it as a desideratum of a question answering system as does Berwick [1983]. This has a flavor of data mining, which involves learning rules *about* data: the data are there, and upon examination an intensional description may be generated.

Hirschberg developed a formal model of scalar implicature and implemented a question-answering system, QUASI, that incorporated a portion of her theoretical work [Hirschberg, 1991]. QUASI answered yes/no questions in a cooperative manner. That is, rather than providing straight yes/no responses to questions, QUASI was able to elaborate the answers. The goal was for QUASI to *not* generate answers that would license false implicatures in the mind of a questioner. Answers judged by the system to be cooperative were sent to a discourse planner, which determined the most appropriate response to give based on its discourse model.

While Hirschberg's emphasis differs from ours, her work is relevant. She incorporated notions of basic-level categories [Rosch and Mervis, 1975] into her theory, used taxonomic hierarchies of concepts, and generated partially ordered sets of answers. Basic-level categories are important in determining appropriateness of answers, and specificity and generality of answers is conveyed in a taxonomic hierarchy by subsumption relations. The relationship of this work to ours is that Hirschberg recognizes a partially ordered set of answers and that taxonomic relationships between answers do not render less specific answers uninteresting; rather, there is a recognition of the fact that answers at different levels of specificity are all potentially of interest.

QUEST [Golding *et al.*, 1990, Graesser and Franklin, 1990, Graesser *et al.*, 1992] is a more recent computational model of cooperative answering. Taking into consideration cognitive science research, in particular psychological models of question answering, QUEST embodies a complex network of generic concepts that can be retrieved in response to questions. The focus is on structuring knowledge to enable human-like question answering. There are two aspects of the work done by Graesser *et al.* that are directly relevant. First, answers lie on "legal paths" from the "entry node" in the network that is activated by a question. Our work relies on a notion of relevance is directly related to being able to construct a *path* (chain of reasoning) from a question to an answer. Second, the notion of good answer is related to the structural distance between a question and an answer (shorter paths mean better answers), which is analogous to the answering approach in description logics: the first answer returned is either the direct sub- or super-concept of a query concept. The goal of the system is to select good answers, and the concept of "goodness" is part of the search strategy: Graesser is a psychologist who has extensively studied question answering in people in order to develop a theory of "answer goodness"; thus, the theory is naturally an integral part of the model and corresponding computational system. There appears to be no separation of information in this system into extensional and intensional; what is represented is a hierarchy of concepts (intensional entities). In answer to a question that asks for an example of an electronic device, "VCR" (which is a category, not a particular instance of a VCR) is given. Different question categories are identified with different answer categories; it is the type of question that determines which arcs should be followed in the answer search. This is a more focused idea of answer that what is proposed here, which has the advantage of producing only desired types of answers, but the disadvantage of failing to produce an answer when, in fact, relevant information may be available.

CARMIN [Gaasterland *et al.*, 1994, Godfrey *et al.*, 1994] is another example of a cooperative answering system. Work on CARMIN has included correcting false presuppositions, query relaxation, query redundancy detection, and a user interface. A particular focus in this systems is semantic query optimization. The emphasis here has been work on the queries rather than the answering process. Query relaxation is a response to a failure to find information. It is one way in which information that is *less* desirable, or less specific than that originally desired (and expressed in a query), may be returned to a questioner. Our approach is to expand the notion of answer, rather than query. When a query is viewed as a probe for relevant information, regardless of its information content, there is no need to go back and relax a query after a failure. A failure of a query, given our broad definition of answer, means there is nothing relevant to the query known by an answering agent.

CoBase [Chu *et al.*, 1994, Chu *et al.*, 1996, Minock and Chu, 1996] is a cooperative database system that makes query relaxation, generalization, and specialization available to a user. As part of this effort, the query language *CSQL* has been developed. *CSQL* interacts with a type-abstraction hierarchy that contains general domain knowledge at the top level and specific domain knowledge at the lower level of the hierarchy. This domain knowledge can be automatically generated from a domain database, and the system has been used successfully in several different domains. Its major features include the ability of a questioner to make adjustments to a query, the ability of the system to return approximate answers that do not precisely match the question that was asked, explanations of answers, summarization of large answers, and user feedback in order to improve the system over time. The type-abstraction hierarchy allows the system to move very easily between general and specific concepts that are taxonomically related, embodying some of the attractive features of description logics in terms of querying.

2.6 Resolution-Based Systems

A traditional resolution refutation proof provides the answer to the question, "is this proposition consistent with a given rule base?". If the proposition 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 the *answer literal* [Green, 1969a, Green, 1969b]. The answer literal enables a resolution refutation theorem prover to keep track of variable bindings as a proof proceeds. For example, if the question asked has the logical form $\exists x\, P(x)$, then a refutation proof is initiated by adding the clause $\{\neg\, P(x)\}$ to the knowledge base. When an answer literal is employed, the clause $\{\neg\, P(x),\ ANSWER(x)\}$ is added instead. The x in the answer literal ($ANSWER(x)$) will reflect any substitutions made to the x in $\neg\, P(x)$, but the $ANSWER$ predicate will not participate in (thus, will not effect) resolution. This is because it is a newly introduced predicate that only appears in positive form.

When a refutation is found, the variable bindings stored in the answer literal at that point correspond to individuals in the domain that satisfy the original question. An answer that mentions individuals in this manner is called a *specific* or *extensional* answer. Green's work defined answers as individuals who satisfy a set of properties. This approach has proven to be remarkably persistent. Green showed that when the set-of-support strategy [Wos *et al.*, 1965, Loveland, 1978] is employed, and the initial set of support comprises the negation of the query disjoined with an answer literal, all resolvents will contain at least one answer literal. Note that answer literals are never resolved, because they are always positive. In addition to tracking substitutions, we use the answer literal in our work as a "tag": any clause containing at least one answer literal is a descendent of the original query, making it *relevant* to the query according to our definition. Our definition of answer thus includes clauses containing other literals in addition to answer literals, while the traditional definition of answer includes only clauses with no non-answer literals.

Luckham and Nilsson proposed a method for extracting answers from resolution proof trees [1971]. Nilsson further refines the technique in his textbooks [Nilsson, 1980, Nilsson, 1998]. As in traditional approaches, they focus on existence questions of the form $\exists x\ P(x)$, where in the general case x is a set of variables and P represents properties asserted to hold over the x. Existence questions may be viewed as yes/no questions that ask whether or not there exist individuals x that have property P. If the answer is "yes", then returning individuals that satisfy P as an answer can be viewed as interpreting the original question as *Which individuals, if any, satisfy property P* .

Employing the rule of resolution as the single rule of inference, Luckham and Nilsson construct a refutation proof (if possible). If there is a proof, the proof tree is used as input to a procedure that constructs an individual or individuals that comprise a satisfying answer, which is termed the *answer statement*. The individuals in Luckham and Nilsson's technique are constructed by disjoining (in separate iterations) literals from the original question to the clause form of its negation. Thus, if the question is $\exists x\, P(x)$, resolution proceeds from the clause $\{\neg\, P(x)\}$. If a proof is found, the disjunction $P(x) \vee \neg\, P(x)$ is formed, and the proof steps are retraced with this new clause, substituting values for x and renaming as was done in the original proof. When the bottom of the proof tree is reached, a substitution instance of the disjunct $P(x)$ will be either a specific answer or what we have termed an unconstrained generic answer. Had reasoning proceeded with an answer literal, the clause added to the knowledge base would have been $\{\neg\, P(x),\, ANSWER(x)\}$. As mentioned above, the answer literal has no effect on resolution. Thus, the result of resolution is the same whether or not an answer literal is employed. A substitution instance of $P(x)$ is formed by instantiating x with the argument to the answer literal once the proof is complete. Thus, the techniques are logically equivalent. The annotation performed by Luckham and Nilsson that instantiates the variables from the original question could be accomplished directly by using a slightly altered form of the answer literal: $ANSWER(P(x))$. The only difference between this approach and Green's is the presence of the predicate from the original question in addition to the variable. The clause that would be added to the knowledge base would be $\{\neg\, P(x),\, ANSWER(P(x))\}$, and a substitution instance of $P(x)$ would be created in tandem with a proof.

Luckham and Nilsson's technique is capable of providing answers that contain variables, expanding Green's original answer definition. This is because non-ground clauses may be resolved to the empty clause, and this in fact is what happens in some of their examples. Their only requirement for an answer is the presence of proof, not the presence of ground terms that are used in discovering a proof. Because Luckham and Nilsson's answers may be ground or may contain variables, they are able to prove that their technique always finds an answer at least as general as Green's. In other words, there are cases where their answer contains a variable and Green's contains a ground term. Their annotated resolution tree may contain generic and hypothetical answers along a branch leading to the empty clause. These are not, however, considered by Luckham and Nilsson as answers: it is the leaves of the proof tree where their answers are found. Most significantly, there must be a resolution proof in order for there to be answers, which is not the case in our work.

An important result presented by Luckham and Nilsson [1971, 1980] is the treatment of Skolem functions and constants in answers. The following example from Nilsson's 1980 text [1980] serves as an illustration:

> Given the following rules:
>
> - For all x, x is the child of $p(x)$
> $C(x, p(x))$
> - For all x and y, if x is the child of y, then y is the parent of x
> $\neg C(x,y) \vee P(y,x)$
>
> What is the answer to the question:
>
> - For any x, who is the parent of x?
> $\forall y \exists x P(x,y)$

The answer found is *P(p(A), A))*, where A is a Skolem constant generated on converting the negated question into clause form. Nilsson and Luckham offer a proof [Luckham and Nilsson, 1971] that such Skolem entities actually represent universally quantified variables. They can be viewed as corresponding to the "arbitrary individual" described by Fine [Fine, 1983, Fine, 1985]. The point emphasized by Nilsson is that such an answer shows that *any* individual satisfies the property. The validity of replacing Skolem functions with universally quantified variables is an important component of the generalized characterization of answer we develop in the context of resolution theorem provers.

Cholvy and Demolombe have used a logic framework for investigating generic answers [Cholvy and Demolombe, 1986, Cholvy, 1990]. They define an intensional answer to a query $Q(X)$ as a set of formulas $A(X)$ such that $\forall X(A(X) \rightarrow Q(X))$ follows from the knowledge base. They use first-order predicate calculus for knowledge representation and resolution with the set-of-support strategy for reasoning. The resolvents generated in this manner are deemed the *pertinent* clauses. They prune the set of pertinent clauses by deleting tautologies and clauses that are subsumed by other clauses. The conjunction of the remaining pertinent clauses constitutes an intensional answer. The issue of hypothetical answers does not arise in the work of Cholvy and Demolombe due to the fact that hypothetical answers are always subsumed by intensional answers. There are two reasons for this. First, with respect to the questions Cholvy and Demolombe investigated, the hypothetical answers found were all subsumed by generic answers. Second, they determine answers with respect to the *complete* set of pertinent clauses. One of our contentions is that question answering happens in time. Given this fact it is theoretically interesting, but not always practical, to work with the complete set of answers. The set of *pertinent clauses* is not exactly analogous to our notion of answer set, which we partition according to answer type so that different types of answers are not automatically subsumed by other types of answers. What is close in spirit to our work is the lack of selectivity during the search process: answers are not precluded from the pertinent set simply because they are not in the correct form.

2.7 Question Answering in Description Logics

Using the description logic CLASSIC, Borgida and McGuinness have developed a representation for queries that allows considerable flexibility in retrieving answers [Borgida and McGuinness, 1996]. CLASSIC can respond to questions with *concepts*, which are intensional entities. Queries themselves are represented as under-specified concepts in the concept hierarchy from which answers are derived. A query concept is viewed as a mask for filtering data: it can partially match concepts without having to match or retrieve values from *all* of the slots in a concept. While there has been work on developing rich concept languages for querying [Lenzerini and Schaerf, 1991, Donini *et al.*, 1992], in all such cases the variables used in query concepts could only match individuals.

Borgida and McGuinness's approach is focused on issues of generality and specificity in question answering. By including a query concept in a taxonomy and allowing partial matches of other concepts, general answers that are more general than a query in some, but not all respects, can be returned as answers. Both specific and generic answers may be found. Hypothetical answers to not arise in the description logic framework.

2.8 Abduction

The use of logic and the *interpretation* of resolvents links our work to abductive reasoning. Abductive hypotheses are associated with hypothetical answers: both are related to reasoning in the face of a lack of information. Abduction also describes an *interpretation* of resolvents that, under other circumstances such as theorem proving, would have no interpretation. This relates to our work in understanding "ignored" resolvents.

Abductive inference is described by Hobbs *et al.* as, "inference to the best explanation" [1993]. A slightly different definition of abduction is found in Kakas, Kowalski, and Toni [1998, page 237]:

> Abduction consists in computing ... explanations for observations. It is a form of non-monotonic reasoning, because explanations which are consistent with one state of a knowledge base may become inconsistent with new information. ... The existence of **multiple explanations** is a general characteristic of abductive reasoning, and the selection of "preferred" explanations is an important problem.

Abduction involves the search for explanations. Reasoning may proceed from a situation to a set of possible explanations, or the other way around. The way in which reasoning proceeds is not important for our discussion.

The philosopher Lorenzo Magnani [1998] provides the following definitions of abduction that relate to the difference between what is commonly termed "abductive reasoning" and hypothetical question answering.

> There are two main epistemological meanings of the word abduction: 1) abduction that only generates plausible hypotheses (selective or creative), and 2) abduction considered as inference to the best explanation, that also evaluates hypotheses. All we can expect of our "selective" abduction, is that it tends to produce hypotheses that have some chance of turning out to be the best explanation. Selective abduction will always produce hypotheses that give at least a partial explanation and therefore have a small amount of initial plausibility. In this respect abduction is more efficacious than the blind generation of hypotheses.

Considerable work has been done on abductive reasoning as a way of generating explanations for observed phenomena [Poole *et al.*, 1987, Poole, 1989, Poole, 1990, Kakas *et al.*, 1998, Kowalski *et al.*, 1998, Satoh, 2000]. These reasoning systems are capable of assuming a (defeasible) hypothesis that is consistent with, but not provable from, an existing (consistent) knowledge base. Diagnosis is a common application area for abductive reasoning, including medical diagnosis and circuit-fault diagnosis. The trigger for abductive reasoning is an observation or fact. For example, if an observation such as *the grass is brown* is made, possible abductive explanations might be *there is a drought* or *it is winter*. These explanations are related to the observation by rules: *there is a drought* $\rightarrow$ *the grass is brown*; *it is winter* $\rightarrow$ *the grass is brown*. Different possible explanations are thought of as *competing*, and it is desirable to select the *best* explanation from among such a set.

A set of *assumable* predicates is generally predefined and associated with an abductive reasoner. This set restricts the predicates that may be abductively assumed.

While abductive hypotheses are generally associated with failed branches of a proof tree [Cox and Pietrzykowski, 1986, Cox and Pietrzykowski, 1987], other criteria are also used to narrow the selection of abductive hypotheses:

- The hypothesis should convey a *cause*; it should not explain an effect in terms of another effect [Kakas *et al.*, 1998]. Cox and Pietrzykowski refer to this property as *basic* [1987].
- The hypothesis should not be subsumed by any other hypotheses.

In order to enforce these requirements and to limit the number of possible abductive hypotheses a system may infer, abductive reasoning systems incorporate a preferred, predefined set of "abducibles": predicates that generally do not appear as the consequence of any rules. In order for a clause to be considered as a possible abductive hypothesis, it must match one of these predefined predicates. Abducibles are "needed to distinguish abductive explanations from inductive generalizations" [Kakas *et al.*, 1998, page 237].

Cox and Pietrzykowski [1986] broaden the typical notion of abductive hypothesis in their paper, "Causes for Events: Their Computation and Applications". Reasoning is triggered by new knowledge of an event, where an event is any closed formula. A cause is "a prenex formula, the matrix of which is a conjunction of literals" [Cox and Pietrzykowski, 1986, page 608]. Their causes are all associated with the "dead ends" of branches of the proof tree, as described above. Their procedure for converting a cause from clause form to a quantified formula employs their procedure for reverse Skolemization [Cox and Pietrzykowski, 1984] and is analogous to the way in which we translate the hypothetical component of an answer into the antecedent of a rule. They do not, however, discuss the utility of the hypothetical answer part as a follow up question: their focus is on identifying causes. They also note the fact that causes may be used to discover "deficiencies of the knowledge base" [Cox and Pietrzykowski, 1986, page 608].

Another body of work on abduction is associated with research in natural language understanding [Stickel, 1990, Hobbs *et al.*, 1993]. This work diverges from abduction in service of diagnosis: the goal of abduction for natural language is to infer, from a sentence, a set of assumptions that must be made by a listener in order for the listener to know that a sentence is true. In addition, the goal is for these inferences to be the *least* specific inferences that can be made. This may be contrasted with abduction in service of explanation, where *most* specific abductive inferences are typically sought. There is also no notion of an *assumable* set of predicates where natural language understanding is concerned: the goal is to understand language as completely as possible, and restricting the form of abductive hypotheses would impede this goal.

Charniak [1988] presents a detailed analysis of what it means to understand actions in terms of motivations, where actions are observable "facts" and motivations for the observed actions are discovered by reasoning. One of the greatest difficulties he encounters is in selecting the *best* motivation, and this is presented as one of the important challenges in abductive reasoning. An additional difficulty is in coming up with possible motivations. In order to carry out his analysis, Charniak devises a method of unifying Skolem functions and constants, provided their equality in terms of role is apparent. The following is an example presented in Charniak's paper. If it is known that Fred is holding a piece of reading material that is dependent upon Fred (the reading material object is represented as a Skolem function), and it is also known that Fred is holding a book and that a book is a kind of reading material, it seems perfectly reasonable to allow for the unification of the book being held by Fred (a constant) and the reading-material object held by Fred that is represented by a Skolem function [Charniak, 1988, page 280]. Analyzing Skolem functions and constants in a manner that makes sense, even if it does not accord with commonly accepted logical practice, is clearly important in explanation. We discuss problems of interpreting Skolem constants and functions in the next chapter in relation to hypothetical answers. We do not encounter the specific problem addressed by Charniak, who is working in a different formalism with different goals, but the need to identify Skolem constants and functions with known, "matching" individuals is an example of a problem where linguistics meets logic and linguistics points towards an interpretation that makes sense. Our interpretation of Skolem functions and constants also draws upon linguistics and from what is clearly common sense.

Abductive inference to least or most specific hypotheses sets the goals of this type of reasoning apart from the task of question answering. If hypothetical answers exist at different levels of generality in a question answering situation, they are all deemed equally valuable.

Sergot [Sergot, 1983] noted the utility of abductive hypotheses in terms of question answering: rather than having a reasoner assume one or more defeasible, abductive hypotheses, he uses the abductive hypothesis as a question and immediately obtains information to "settle" the hypothesis. This information could come in the form of sensing the world or asking another agent. The goals of abductive reasoning and question answering differ: abductive reasoning seeks probable causes for situations, where question answering seeks to *inform* a questioner in the best manner possible given a particular question. An answering agent does not add the hypothetical portion of an answer to his or her knowledge base, whereas an abductive reasoner will add an abductive hypothesis: the goal of question answering is to satisfy a questioner, which is generally (though not necessarily) presumed to be an agent other than the answering agent, whereas abductive reasoning is a way in which an agent performs reasoning to serve its own goals. Some of the work described later in this dissertation on abduction and question answering is presented in Burhans and Shapiro [2001].

2.9 Anytime Reasoning

Anytime reasoning [Dalal, 1996] refers to a reasoning process whereby the quality of a solution is related to the amount of time taken to compute the solution. At any time during reasoning, an anytime reasoner can give you the *best* information that has been discovered up to that point. Thus, in an anytime reasoner, each reasoning step produces a meaningful result. In resolution refutation, each reasoning step produces one or more resolvents. When resolvents are identified with answers, each step of the reasoning process is identified with the production of an answer. Thus, our proposed question answering interpretation of resolvents has the flavor of anytime reasoning.

Anytime reasoning is extremely useful when the resources available for reasoning may not permit a complete exploration of the reasoning space. Anytime reasoning is a form of approximate reasoning [Verberne *et al.*, 2000], where the quality of a solution may not be ideal, but is nonetheless good enough.

The process of question answering shares some, but not all, of the features of approximate or anytime reasoning. As long as there is information relevant to a question in the knowledge base, it *is* the case that, at any point in the reasoning process, we guarantee that there is *some* answer. This allows bounds to be placed on the time alloted for question answering with guaranteed results. This is not the norm in most question answering systems, where reasoning proceeds until *all* answers are discovered (when possible). Answer discovery is then followed by the selection of a *best* answer from the answer set.

The difference between question answering and approximate reasoning is that there is no guarantee that the question answering agent is "honing in on" a best answer. In fact, there is no notion of what constitutes a best answer, *unless* the answering agent is *completely* familiar with the questioner's knowledge base. While user modeling is an important and pervasive part of user interactions such as question answering, it is unrealistic to assume a *complete* user model. Absent that, it is impossible to assume, for example, which level of generality of an answer best suits a particular questioner.

Question answering is examined by Shapiro and Woodmansee [1969] using the SAMENLAQ II system. They describe an iterative style question answerer that, like Prolog, may be asked to search for additional answers each time an answer is produced, and will attempt to continue to find answers as long as it does not come up empty handed. It is not necessarily the case, either in Prolog or in SAMENLAQ II, that each, single reasoning step produces an answer according to their definitions. Like anytime reasoning, further reasoning may prove fruitful in expanding an answer set, but every reasoning step is not necessarily identified with an answer. This contrasts with our definition of answer, where we can guarantee that each reasoning step is associated with the production of some answer.

There is no doubt that specific or generic answers are always more informative than hypothetical answers, so the search process up to the point of discovery of specific or generic answers *is* analogous to anytime reasoning. However, once generic or specific answers are being discovered, there is no sense of convergence to a *best* answer unless a user is only interested in specific answers, which would always be at the ends of branches of a proof tree.

2.10 Question Answering in Databases and Data Mining

In "Intelligent Query Answering in Rule Based Systems" [1987], Imielinski proposes the idea of intensional answering:

> We propose that in large knowledge bases which are collections of atomic facts and general rules (Horn clauses), the rule should be allowed to occur in the answer for a query.

Imielinski identifies advantages of intensional answers, including informativeness and succinctness. In addition, he describes how the use of rules as answers can be viewed as a "resource limited computation" in that computing a full, extensional answer may require significant time. The rules comprise the *intension* of a database, that is generally quite small relative to the facts, or *extension* of a database using the formalism of deductive databases. His definition of relevance is based on predicate sharing between an answer and a query. Each of his intensional answers is a complete answer; that is, it implies the entire extensional answer. Imielinski has continued his work on intensional answering in the area of knowledge discovery and data mining (KDDM) [Imielinski and Mannila, 1996], where the large size of the databases can make it impractical to provide complete or even partial lists of facts in response to queries. Still, the view of the intensional answer is that it "stands" for an extensional answer, which is a narrower definition than that we propose. This view is, however, a consequence of the completeness requirement for all answers.

Shum and Muntz develop a theory of how to construct and reason with "aggregate responses" to queries in a database [1988a, 1988b]. They note, as have other researchers, that "An enumeration of individual objects is not always the best means of information exchange" [Shum and Muntz, 1988b, page 479]. While recognizing the need for intensional answers, they are also concerned with efficiency of reasoning. They develop measures of *conciseness* and *preciseness* by which numeric expressions describing a fraction of the members of a group may be evaluated for information content. Conciseness is simply the number of terms in an expression, and preciseness is the amount of uncertainty associated with an expression, measured using Shannon entropy. The types of aggregate expressions with respect to which precision and conciseness are measured are, for example, "40/100 engineers" [have property *P*]. This aspect of their work requires full information content in the database: if you don't know how many engineers there are, you can't refer to 40/100 of them. They also work with taxonomies of concepts where numeric information is not required. In this context, they still look for a minimal answer set, where once again what is minimized is the number of terms in an expression. They describe answers such as "all managers except for John Smith", which, in the next paragraph, will be termed a mixed answer.

Further work on intensional answers that involves integrity constraints to filter out inadequate answers was done by Pirotte, Roelants, and Zimanyi [1991]. Their formalism includes complete separation of the extensional and intensional portions of the database, which is typical in the database context. Their intensional answers are generated without any recourse to the extensional database;

this work predates the development of large-scale knowledge discovery from the extensional database that could be used to update the intensional database component.

Motro also worked on intensional answering in the context of relational databases [Motro, 1989, Motro, 1994, Motro, 1996]. He characterizes an intensional answer as “a description that implies part or all of the specific answer” [Motro, 1989]. Motro (1994) identifies two types of intensional answers: pure and mixed. A pure intensional answer contains no extensional information, while a mixed answer may contain both intensional and extensional. Motro (1996) further classifies intensional answers into three categories with respect to their extensional counterparts: constraints (necessary conditions), containments (sufficient conditions), and descriptions. A constraint describes a condition that is satisfied by the whole answer, and a containment describes a subset of the database that is entirely contained in the answer. For example, in response to a query about employees with a salary greater than $20,000, “age greater than 30” might be a constraint, while “salary greater than $30,000” is a containment. Other intensional information is termed descriptive. The focus of Motro’s work has been on characterizing types of intensional answers. Note that the classifications are based on set membership of *individuals*, or extensional information; the expectation is that the database will have an extensional component. One of the most important contributions of Motro’s work is the departure from the completeness requirement for answers: containment descriptions are considered as valid answers even though they do not by themselves entail the original query.

Recent work by Bonnet and Tomasic [1998] examines the problem of queries that can not be completely answered in the context of multiple, heterogeneous data sources. A partial answer is given in response to a query that can not be completely answered due to problems accessing some of the data sources. They propose splitting a partially answered query into two parts: the retrieved information, which can be used to help attain the information not yet found, and the unsuccessful part of the query, which is a subset of the original query. Their representation is a slot and filler template. While hypothetical answers *are* incomplete, there are differences between their approach and our hypothetical answers. The “unanswered” portions of their queries are formatted according to the template for the original query, whereas a hypothetical answer can bring together two seemingly disparate pieces of information, where there is no sharing of predicate structure. For example, “If you like cricket, you should have a beer” seems very odd, but would qualify as a hypothetical answer as long as the answering agent possessed the appropriate information linking beer drinking and cricket. The relationship of Bonnet and Tomasic's work to hypothetical answers is that further information is needed in order for a hypothetical answer to be considered informative. The hypothetical part of the hypothetical answer may be considered as an unanswered query that might be posed to a new data source in hopes of finding an answer.

2.11 Linguistic Theories of Question Answering

There are many interesting aspects of questions and answers from the point of view of linguistics. The interaction between AI and linguistics is seen in some of the work described in this and the previous section. Most AI researchers in question answering are also interested in linguistics. There is, however, a lot of work in linguistics that is not computationally oriented. The work discussed in this section shows an interest in *generic* answers on the part of some linguists. It also emphasizes the important role of Gricean maxims [1975] in choosing among responses. The role of these maxims as filters in determining the relative value of different answers is clearly important. If filters are used to select from among answers, *when should this take place?* . Our answer to this question is that it should occur in *selecting answers* from an answer set rather than precluding answers from joining the answer set. While some answers may be better than others, there may be no guarantee at the start of answering a question that good answers will be available. There is no uniform agreement among linguists as to the way in which Gricean maxims should be used in computational approaches to language understanding in general, which includes the problem of question answering. Hirschberg's approach, described previously, stands in sharp contrast to the view taken by Dale and Reiter of Gricean maxims [Dale and Reiter, 1995, Dale and Reiter, 1996]. Their opinion is that Gricean maxims are useful as a *post hoc* description of the result of interacting. They feel that any goal-driven natural language generation system is implicitly conforming to Gricean maxims, and there is no need to model them explicitly. This is not the case for the model we propose, where the underlying knowledge representation and reasoning is purely logic-based. The first possible stage in the answering process where Gricean maxims could be incorporated is when answers must be selected from the answer set. While we do not address the natural language generation component of question answering in this dissertation, it is an important enough piece of the question answering picture to warrant including some discussion of it.

In *The Handbook of Contemporary Semantic Theory*, Ginzburg describes a semantics of questions [1996] that builds on the context-dependent framework of situation theory [Barwise and Perry, 1983]. The meaning of a question depends on a situation-invariant semantic analysis in conjunction with a number of situation-dependent parameters: the discourse context, the speaker, the hearer, the time, etc. While Ginzburg does not discuss user models, he emphasizes the fact that finding a resolution to a question (that is, satisfactorily resolving any issues the question has raised) is agent relative. Ginzburg raises issues that accord with our approach to the problem of question answering. Unlike other work on question answering, we are *not* making an *a priori* assumption about which answers are best. In teasing apart the problem of identifying relevant information from determining which piece thereof to return to a user, we are allowing the contextual factors to which Ginzburg refers to enter into answer *selection* at a level above knowledge representation and reasoning. In addition, if there is a paucity of contextual information, our answer set is structured so as to enable a reasonable return and explanation of answers to a questioner.

Kempson's interest in Gricean maxims is focused on presupposition [Kempson, 1975]. It is not our intent to explore issues in presupposition, which is a large area of research in computational linguistics. However, Kempson makes several points that are relevant to this work. First, she notes that the variation in possible answers to a given question (given the same context) reflects different weightings of the Gricean maxims. In addition, it is clear that some answers represent conflicts in choosing which maxims to obey. Consider the following example from *Presupposition and the Delimitation of Semantics* [Kempson, 1975, 163]:

1. A: How long does it take by taxi to Piccadilly Circus?
   B: One minute.

2. A: How long does it take by taxi to Piccadilly Circus?
   B: You don't need a taxi – it's only two minutes' walk.

The first answer is well-formed and follows the quantity maxim, that is, it states exactly as much information as is requested: it answers the question. The second answer fails to follow the quantity maxim. Nothing was asked about walking, yet information about walking is provided in the answer. On the other hand, the second answer may be more in keeping with the quality maxim, which exhorts one to exercise perspicuity. Kempson suggests that apparent conflicts among maxims regarding question answering are indicative of the fact that the questioner has left too much information out of the question; hence, the hearer makes a number of assumptions and proceeds without worrying whether or not they are correct. Note that the answer to the second question is not of the sort that we could produce. If, however, the question had been, *How long does it take to get to Piccadilly Circus*, we could provide a set of *hypothetical* answers that would prove very useful. For example, *If you walk to Piccadilly Circus, then it will take you two minutes*, or *If you take a taxi to Piccadilly Circus, then it will take you one minute*, etc. The second answer provided by Kempson relies upon a presupposition that getting to Piccadilly Circus in the most effective manner is the subject of discussion. If an antecedent to a hypothetical answers is a presupposition in the sense of Kempson, then hypothetical answers may provide a means of incorporating the maxim of quality into computational question answering.

The notion of conflicting maxims is important in providing elaborations of answers to yes/no questions. A one-word *yes* or *no* answer is the best response with respect to quantity and brevity, while an elaboration violates the quantity maxim. An elaboration tells the questioner something she did not explicitly ask to know. On the other hand, the quality maxim is best served by a more informative answer. In determining which answer to select from an answer set, Gricean maxims can be used even in the absence of a user model.

Groenendijk and Stokhof are formal semanticists whose primary interest is in the logic of interrogatives [Groenendijk and Stokhof, 1993, Groenendijk and Stokhof, 1997, Groenendijk, 1999]. Working within the traditions of logic programming and possible-world semantics, they develop a theory whereby a question is seen to divide the set of possible worlds into a number of mutually exclusive alternatives. They refer to their approach to the semantics of interrogatives as *partition semantics*. A partition corresponds to an alternative, defined as a set of possible worlds wherein the answers to a given question are the same. For example, given the question *What barks?* there is a set of possible worlds in which the denotation of $barks(s)$ is the same. These worlds are one of possibly many partitions created by the question *what barks?*The contrast between indicatives and interrogatives is that an indicative *selects* a subset of possible worlds, whereas an interrogative is associated with a *partition* of possible worlds and does not impose a particular selection. We have not explored a possible worlds semantics for questions: our work is firmly rooted in clausal logic for FOPC. An interesting consideration of the approach presented by Groenendijk and Stokhof is the relationship of their semantics to hypothetical answers. The contingency associated with a hypothetical answer is itself a question, thus, it defines a partition on the set of possible worlds. Consider the question *Did John come to the party?*Ȧs a yes/no question, this partitions the set of possible worlds into two alternatives: one includes the worlds where it is the case that John came to the party, and the other includes worlds where it is not the case that John came to the party. What does it mean to have a hypothetical answer? It means that, while the set of possible worlds where *Did John come to the party?* is identified with some partition of possible worlds, we cannot identify a world in that partition. Suppose the following hypothetical answer is discovered: *If Bill came to the party then John came to the party.* This answer is associated with the following question: *Did Bill come to the party?*Ṫhis question defines a partition on the set of possible worlds, and the form of the hypothetical answer *guarantees* that in all the worlds where *Did Bill come to the party?* has a yes answer, *Did John come to the party?* also has a yes answer. Note, however, that Bill's presence at the party is not a requirement of John's having come to the party. If the contingency is false, we are confronted with the original question all over again. A hypothetical answer provides an alternative interrogative that is associated with the same partition as the original: it partitions *this* partition. Thus, partition semantics suggests an interpretation of hypothetical answers and their relationship to original questions. In addition to partition semantics, Groenendijk and Stokhof discuss the issue of providing some versus all answers to a question. Providing all answers is associated with completeness, and may not always be possible, thus, their recognition of the possibility of providing some answers accords with our approach to answering.

The framework within which Groenendijk and Stokhof work is sometimes called the *proposition set approach* to linguistic semantics. Krifka [1999], another formal linguist and philosopher who has focused on questions and answers, espouses another framework within which questions and answers may be interpreted. This framework is the *structured meaning approach* to understanding language. The primary point made by Krifka is that the proposition set approach is unable to avoid the production of under- and over-focused answers, where such answers are super- and subsets of the actual answers. The interesting parallel here is that our theory espouses the production of poorly focused answers. In the interest of finding *some* information, we are willing to sacrifice precision in the short run. Note that in the long run we may hone in on the precise answer(s) desired. One of the difficulties in the linguistic approaches to the semantics of questions and answers is the absence of the notion of reasoning over time. The linguistic theories generally account for an exhaustive set of answers, or a precise set of answers, but not for whatever it is we discover *en route* to answers.

Specific answers are obviously mainstream when it comes to describing answers to questions. While linguists may recognize the importance of generic answers, there is no mention in any of this work of hypothetical answers, which is not surprising. Considering hypothetical answers as "answers" is objectionable to some, given their tenuous status. The discovery of hypothetical answers and their importance is fundamentally intertwined with answering in time. This is something that is not taken into account by the linguists in the way that it must be accounted for when observing a reasoning process. Regardless of whether linguists are interested in hypothetical answers, such answers do accord with the way that we talk. In the absence of what we might term *solid* information, we simply do the best we can in answering questions, and that includes responding to questions with statements like, "If it doesn't rain then I'm going to the parade", which is exactly what hypothetical answers are all about. If they are not answers, the task is set to determine exactly *what* they are. It appears that in the context of Groenendijk and Stokhof's semantics of interrogatives there is a nice characterization of hypothetical answers as alternate ways of describing a partition on the set of possible worlds. If hypothetical answers are not answers, they would be infelicitous responses to questions. Their existence in everyday conversation, and our lack of outrage thereon, indicates that this is not the case. While our methodology and formalism may not directly inform linguistics, we believe that our proposed broadening of the notion of answer represents a step forward not only for the computer science community, but also for linguists interested in question answering.

2.12 Erotetic Logic: The Logic of Questions

In *The Logic of Questions and Answers* (1976), Belnap and Steel develop a comprehensive erotetic logic.[3] This logic relies on the existence of a standard assertoric logic, but the two logics are not analogous. Standard FOPC and propositional logic are assertoric logics; that is, logics of assertions. The primary difference between assertoric and erotetic logic is that erotetic logic is not a deductive system: questions function neither as premises nor conclusions.

The use of a theorem prover as a question-answering system requires us to cast our entire ques-

[3]The term *erotetic logic* was introduced in Prior and Prior [1955].

tion answering endeavor in assertoric logic, where a question *is* an ordinary proposition. This completely contradicts the approach proposed by Belnap and Steel, though other important aspects of their logical endeavor are evident in some of our answer definitions.

Belnap and Steel characterize a question as having two parts: the subject and the request. A question presents a range of alternatives as its subject. These alternatives may be explicitly listed, or indicated using a reference to a condition or matrix. A condition or matrix is a statement form that contains variables, and it may correspond to a finite or an infinite set. This characterization of questions that contain variables is clearly relevant to generic answers. In fact, even in a hypothetical answer we can view the hypothesis as a description of a condition *on* the condition that characterizes the matrix. The request allows a question to contain information about the answer sought, for example, specifying the number of alternatives desired in the answer.

Questions are classified as either elementary or non-elementary. *What* questions are of interest in this research. A *what* question is non-elementary and has matrices as its posits *and* desiderata, where the posits are the alternatives presented by the question and the desiderata are the answers. In other words, these are questions whose subjects and answers are generic. Belnap and Steel identify four subtypes of *what* questions as follows (where Ax is the posit and Bx is the desideratum):

1. Equivalence: $\forall x(Ax \equiv Bx)$
2. Necessary: $\forall x(Ax \rightarrow Bx)$
3. Sufficient: $\forall x(Bx \rightarrow Ax)$
4. Nonempty Intersection of Ax and Bx: $\exists x(Ax \wedge Bx)$

Consider the question, "What sorts of things are mammals?". Belnap and Steel point out that this question is ambiguous. The ambiguity, however, is purely an artifact of English syntax. Regardless, the two interpretations of this question as stated serve to illustrate the difference between necessary and sufficient questions. If "What sorts of things are mammals?" is interpreted as a *necessary* question, an appropriate answer might be, "mammals are vertebrates". On the other hand, it can be interpreted as a *sufficient* question, in which case an appropriate answer might be, "horses are mammals". Thus, a *necessary* question elicits an answer that subsumes the referent in the question, and a *sufficient* question seeks an answer that is subsumed by the referent in the question. An *equivalence* question implies that *A* and *B* are intensionally distinct, but extensionally equivalent. This type of question is exemplified by an example from the previous section: the question that asks what sorts of entities are bachelors can be answered with a statement that describes what a bachelor is. A *nonempty intersection* question can be viewed in two different ways. An appropriate answer could either be a concept subsumed by both *A* and *B*, or, in case *A* and *B* are sets of properties, a concept whose properties are the intersection of those sets. An example of such a question is, "What sorts of things are green and mobile?", with the answer, "green robots are green and mobile."

Note the universal quantification in these questions: the typical question posed to a computational question-answering system is existentially quantified. We impose no such restrictions on the form of questions: generic answers are critical for answering universally quantified questions.

A *which* question is an elementary question whose posit is a specific entity and whose desideratum may either be a generic or specific entity. Examples include "Which art works here are for sale?" (specific reference in both cases) and "What would you like for your birthday?" (specific posit, but could be either generic or specific desideratum). While the first question is considered to have a specific reference for the desideratum, we would *not* agree in terms of our answers: where a generic answer exists, we will consider it acceptable. Regarding that question, a possible *generic* answer would be *All of the paintings in Gallery X are for sale*.

While Belnap and Steel discuss the fact that a *what* question may be ambiguous among the four different interpretations (specific/generic in all possible combinations), they do not examine the potential ambiguity between *which* and *what* questions. One reason this did not arise in their book is that they carefully circumscribed the types of questions they were interested in, eschewing pragmatics and natural language.

The purpose of Belnap and Steel's investigation was to develop a formal theory, which might provide a basis for enlarging the scope of the question understanding task. This work is relevant because it recognizes questions with generic subjects and answers as a distinct ontological class. This is in contrast to many of the other analyses of questions, which for the most part fail to recognize the existence of this class. Higginbotham provides a general discussion of approaches to the semantics of questions [1996]. He discusses the difference in meaning between the following two questions:

- Which men are bachelors?

- Which bachelors are men?

He notes that Karttunen's proposal for the semantics of questions [1977], while mapping these two questions onto different underlying semantic representations, ultimately yields the same answer for both: an enumeration of all bachelors. This is precisely the *typical* answer defined in the context of AI to which we have already referred. Higginbotham provides his own analysis of this question pair: the first should be answered with an enumeration of all bachelors, and the second is, "a silly question, since all bachelors are men" [Higginbotham, 1996, 368].

Our interpretation differs from those of Higginbotham and Karttunen. If Gricean maxims are assumed, the second question may be taken as an indication that the questioner is not aware of the relationship between the concepts *bachelor* and *man*. Our approach to question answering would allow the rule *all bachelors are men* to stand as a generic answer to the question. This is a clear case where providing generic answers is superior to specific answers. Regarding the first question, I agree that an enumeration of all bachelors is an *appropriate* answer: it accords with the idea of specific answers. I can, however, imagine a context in which an answer that amounts to a definition of bachelor would be appropriate. That is, the first question under certain circumstances might be equivalent to, "What is it that makes a man a bachelor?". A generic answer such as "An unmarried man eligible for marriage is a bachelor" would be a suitable response. It is interesting to note that linguists have generally overlooked the need to provide generic answers to questions, at least in some circumstances where a generic seems to be the most appropriate response.

Hintikka [1985] describes the logic of scientific investigation as "the logic of information-seeking by questioning" [Hintikka, 1985, page 9]. Our approach to question answering might be termed "information-providing". While the informational content of a hypothetical answer is certainly less that its specific and generic counterparts, it is *not* vacuous.

Recent work on erotetic logic has been done by Wisniewski [1999]. He defines an erotetic inference as one that has questions and/or declaratives as premises, and questions as conclusions. His interest is particularly in *abnormic hypotheses* which are needed to explain exceptions to general rules. For example, if you know that birds fly and Tweety is a bird, and you also know that Tweety doesn't fly, the question this abnormality poses (using Wisniewski as a guide) is not *Why doesn't Tweety fly?* , but rather *Why is Tweety a bird?* . We have not particularly focused on situations involving inexplicable hypotheses. Working within classical FOPC, we have eschewed problems of conflicting belief, belief revision, acceptance of abnormal circumstances, and so on. Wisniewski's focus is on how erotetic logic can be construed so as to allow for exceptions to rules and to extend general rules by adding to them explicit abnormic hypotheses. In a sense, the hypothetical portion of an answer can be regarded as something "inexplicable", but not in the sense of realizing that something has property $\neg P$ when it ought to have property P. A hypothetical part of an answer does not contradict what is generally accepted. Failure to verify the hypothesis associated with a hypothetical answer is not associated with contradiction; rather, the unverifiable hypothesis is consistent with respect to the knowledge base used in answering a question.

Bromberger [1992] is concerned with *explanation*: abductive reasoning is the search for explanations. This is where Bromberger's philosophical theories inform our search for answers. The need for explanations arises when we are in a *p-predicament*. A *p-predicament* is a situation in which nothing we know accounts for the truth or lack thereof of a proposition *p*. The search for an explanation that alleviates a *p-predicament* is often initiated by asking a "why-question". For example, you may hear that a good friend of yours smokes. You know nothing that supports or detracts from this piece of information, but you are puzzled and seek an explanation as to *why* your friend smokes. Fundamentally you are unsure as to the truth of the assertion. If I ask you if your friend smokes, and you have no definitive belief either way, you may try to come up with explanations that support or subvert the assumption that your friend smokes: *His father had lung cancer so he would never smoke.*, or *He always was rebellious, and rebellious people smoke.* Such conjectures fall into the category of hypothetical answers, though the purpose of such answers is not necessarily explanation. If I ask you if your friend smokes, I have not asked a why question but rather a verification question. While failing to answer the question directly, you might be able to offer a plausible explanation as to why the propositional content of the question is the case (or conversely, is not the case). This is essentially our argument for the validity of hypothetical answers. In effect, a hypothetical answer is an answer to a different question from that which was originally posed, but it is better than no answer at all. In addition, support for considering them at all can be found in the way in which people answer questions: we often take a while to "find" an answer, and in the absence of an answer we try to be helpful.

2.13 Summary

In summary, work to date has explored different types of answers within and outside the theorem-proving context. The focus for theorem-proving systems has been on specific answers and on efficient answering techniques. Even where the notion of answer has been decoupled from that of proof, the theorem-proving paradigm lingers in the demand for completeness of systems. Where different types of answers in the theorem proving context have been proposed, they have not been formally characterized, and they do not represent a complete partition of resolvents.

Linguistic and philosophical approaches to question answering do not recognize hypothetical answers as a legitimate answer class. The status of hypothetical answers is clearly different from that of specific and generic answers, and they do not fall under traditional definitions of "answer". Our contention is that hypothetical answers are found in conversation as responses to questions. They clearly, then, do bear a relationship to questions. Linguists and philosophers have explored issues related to explanation, which is clearly an important property of answers. The lack of a strong connection between computational systems and linguistic and philosophical theory continues to be an issue. In particular, without the fundamental idea of reasoning over time to discover answers, it is difficult to recognize the importance of hypothetical answers. Theories tend to start with a specification for a complete set of answers rather than considering how such a set might be constructed. We are not against theories, but they can be built so as to reflect the embodied nature of an act as fundamental as question answering. It is our goal to relate our work on question answering not only to the theoretical frameworks developed by philosophers and linguists, but also to current implemented systems and work on question answering using large text corpora. All of these areas of research contribute to our overall understanding of the meaning and role of questions and answers.

Chapter 3

A Reasoning-Based Model of Answering

3.1 Introduction

The question-answering process is initiated when a question is posed to an answering agent. Figure 3.1 shows our proposed model of answering. Because our interest is in the process of *answering*, the details of the *questioner* are not addressed by our model. While we show the questioner and answering agent as disjoint, they need not be separate entities: self-questioning is a special case of question answering. For purposes of discussion, however, we will refer separately to a questioner and an answering agent with the understanding that they may be the one and the same.

Broadly, answering may be seen as a complex process involving four major steps: answer search, answer set generation, answer selection, and answer communication. Answer search involves finding answers. Answer set generation involves keeping track of answers that have been found. Answer selection refers to the process of selecting from among answers that have been found, and answer communication is the task of responding to the question with an answer. These processes are not strictly sequential, though there are dependencies among them. Until at least one answer is found, the answer set is empty; thus, the answer set generation process is idle. Once there is an answer, an answer set can be generated. Given an answer set, the selection process can be invoked to select an answer from this set, and the selected answer can then be communicated. Thus, the dependencies between these processes involve data availability. Once an answer set is generated, all processes may proceed in parallel as long as sufficient data is available.

These four tasks may not all be present in a question-answering system. Minimally, ques-

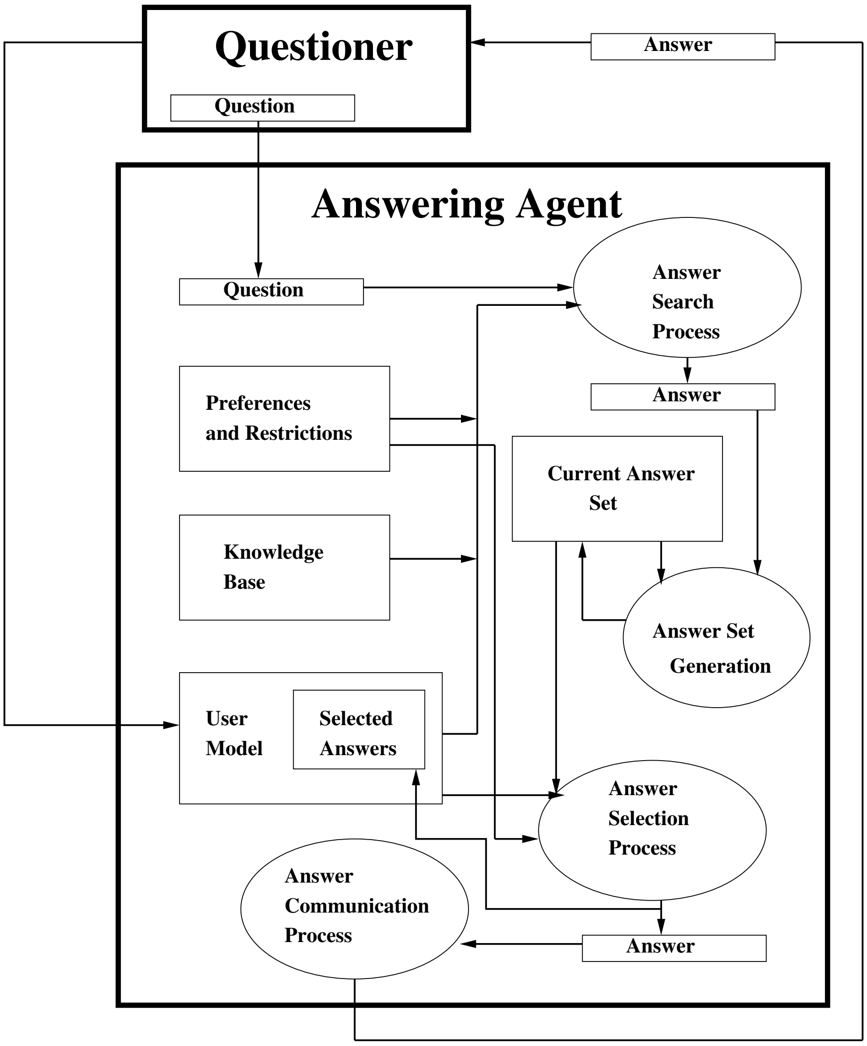

Figure 3.1: **A Reasoning-based Model of Answering** The two agents involved in question answering are the **Questioner** and the **Answering Agent**, represented by the rectangular boxes with heavy outlines. Rectangular boxes shown inside of the two agents represent data, while ovals represent processes. The arrows indicate data paths. A detailed description of each component of the model is provided in Section 3.2.

tion answering involves finding and communicating answers. Not all question-answering systems construct an answer set, and answer selection is moot when all answers that are discovered are communicated to the questioner. As discussed in the previous chapter, the selection process may be intertwined with the search process, in which case potential answers are discarded if they fail to meet certain criteria. We capture this in our model by including "preferences and restrictions" as a potential input to both the search and selection processes: a set of hardwired selection criteria that are used to guide answer search may be viewed as system preferences or restrictions.

Of these four tasks, this dissertation is focused entirely on answer search and answer set generation. Answer search presumes some criteria by which "answerhood" is judged. Chapter 4 is concerned with the problem of defining "answer", and how answer definition is related to issues of answer search. This investigation takes place in the context of a resolution refutation theorem prover. Answer set generation is the subject of Chapter 5, where the problem of generating an answer set is examined. The remaining two tasks, namely, selection and communication, fall squarely into discourse analysis, and are not addressed in this work.

The purpose of the model presented in this chapter is to clearly demonstrate where our contribution fits into the larger picture of question answering, and to cleanly separate the different components involved in answering. The model is based on the identification of the process of reasoning with the process of answering. In this dissertation, we will specifically interpret reasoning in a resolution theorem prover *as* a process of answering.

3.2 The Components of the Model of Answering

This section describes the different components of the model of answering shown in Figure 3.1. The model comprises two agents, termed the questioner and the answering agent. These agents are represented in Figure 3.1 by the rectangular boxes with heavy outlines. Data that is internal to the agents is represented as rectangular boxes inside the agents, while the ovals represent processes, and the arrows represent data paths.

1. **Questioner**

   The questioner is the agent that devises and poses the question. As mentioned above, there is no requirement that the questioner and answering agent are disjoint; rather, for purposes of discussion it is helpful to separate the answering process from that of questioning. The paucity of detail in the box representing the questioner mirrors our focus on the task of answering.

(a) **Question**

The only internal component shown for the questioner is the question, which is conceived by the questioner and posed to the answering agent. The question would obviously be represented in the language of the questioner, which may differ from the language of the answering agent.

2. **Answering Agent**

The answering agent is the agent that attempts to answer the question. Most of the components of the model are internal to the answering agent, and comprise the following:

(a) **Question**

The question is the probe for information that initiates the entire question answering process. The question here is assumed to be represented in the language of the answering agent. If this differs from the language of the questioner, there is a need for translation between the two languages, which could occur at any point along the arrow from the questioner's "Question" to the answering agent's "Question". This arrow indicates the flow of information (the question) from the questioner to the answering agent.

(b) **Answer Search Process**

The question is input to the answer search process, which is the reasoning mechanism employed by the answering agent in answering a question. Other inputs to answer search include the answering agent's knowledge base, preferences and restrictions, and the answering agent's model of the questioner (user model).

i. **Knowledge Base**

The answering agent's knowledge base is the primary source of information used to answer a question. Other information may come from preferences and restrictions and a user model (of the questioner).

ii. **Preferences and Restrictions**

In the model, "preferences and restrictions" refers to anything that affects the answer search process or the answer selection process. Preferences may be implicitly or explicitly incorporated in search strategies. For example, a question-answering system that provides only specific answers may do so in accordance with the preference of a user, or may do so because the system is incapable of providing other types of answers. While we advocate a broad definition of answer, with preferences being incorporated into answer selection rather than used to restrict answer search, the connections of this component both to the search and selection processes reflect our bias as well as those of other question-answering systems that apply restrictions to the answer search process.

iii. **User Model**

The issue of user models in question answering is not addressed in this dissertation. Nevertheless, a user model is clearly an important part of the question-answering process. The arrow from the questioner to the user model indicates a potential flow of information from the questioner that is used in constructing a user model. The better the user model an answering agent has of a questioner, the better it will be able to select *good* answers for a question posed by the questioner. A user model is not required for question answering, and will complicate the processes of answer search and selection. Information in the user model may be used in the answer search process to help target particular types of answers more quickly. A user model is obviously important in answer selection. Minimally, any answering agent should construct a user model consisting of the answers it has returned to a questioner, which is reflected by the presence of the box labeled "Selected Answers" inside the user model. An additional possibility is that the questioner provides information to the answering agent in order to help the answering agent answer a question. For example, if the answering agent produces a hypothetical answer, and the questioner knows whether or not the hypothesis is true, the questioner can communicate this information to the answering agent. Such information may then be used in the search for answers. Whether or not this information becomes incorporated into the answering agent's knowledge base is not specified: this issue lies outside the scope of the model, and depends upon considerations such as the reliability of the questioner as a source of information. Note that it makes sense to use information *believed* true by the questioner in the answer search process even if the answering agent remains agnostic as to its opinion regarding the truth of the information.

iv. **Selected Answers**

"Selected Answers" are those that have been selected from the answer set to generate responses for the questioner. Note that this set may include answers *not* found in the current answer set, because an answer that is found at one point in time may later be subsumed by another answer and discarded due to the fact that is is not as informative as the new, subsuming answer. We suggest that a minimal user model includes the answers that have been returned to the questioner by the answering agent.

(c) **Answer**

The output of the answer search process is an answer. This answer is input to the answer set generation process.

(d) **Current Answer Set**

The current answer set contains all of the informative answers found thus far in reasoning. This set is initially empty. An informative answer is one that provides information not provided by any other answer. The members of this set may change as new answers are discovered. The adjective "current" reflects the fact that answer discovery is an ongoing process that happens over time.

(e) **Answer Set Generation**

The answer set generation process takes the current answer set and a newly derived answer as input and produces a new current answer set. The new answer will be added to the answer set if it is informative with respect to the set, that is, if it provides information not provided by any answer currently in the set. It may be the case that the new answer provides information that supersedes the information provided by some answer(s) currently in the answer set, in which case they will be discarded when the new current answer set is created. This prevents redundancy among answers in the answer set.

(f) **Answer Selection Process**

The current answer set, along with the user model and preferences and restrictions, are inputs to the answer selection process. Any answer in the current answer set may be selected. The output of the answer selection process is an answer, which is sent to the answer communication process in addition to being added to the set of selected answers.

(g) **Answer Communication Process**

The answer communication process is responsible for translating an answer into a response that can be communicated to the questioner. The output of this process is an answer.

3. **Answer**

The box labeled "Answer" along the arrow from the answer communication process to the questioner represents the response. Note that translation from the answering agent's language to the language of the questioner may occur at any point along the arrow indicating the path of the answer, should this be necessary. There is no assumption regarding the way in which the answer is received by the questioner, thus, the answer is not represented inside of the questioner.

3.3 The Process of Question Answering

In "The Process of Question Answering" [1978, pp.107–108], Lehnert states that the processes of finding and formulating an answer can be divided into two parts: content specification and memory search. Content specification "determines what kind of answer should be produced and how to look for it", while memory search is simply the process of actually locating the information. Once an answer is found, the final step is to generate a natural language response that conveys the information. Lehnert's model of question answering exemplifies the conflation of preferences with search. There are several problems with the intertwining of answer selection and search.

First, this approach is destined to fail in case there is no match for a narrow content specification, despite the fact that an answering agent may possess other information relevant to the question. As a consequence, an answering agent may fail to act in a cooperative manner. One option for handling failed questions is for an answering agent to ask the questioner to "relax" the criteria given in a question, or for the answering agent to perform a relaxation operation itself [Gaasterland *et al.*, 1994, Godfrey *et al.*, 1994, Godfrey and Gryz, 1996, Harabagiu *et al.*, 2001]. Constructing an answer set that contains answers at many different levels of generality achieves some of the goals of query relaxation as an initial answer search proceeds, rather than requiring a new search with weaker search criteria.

The second problem with narrowing the search rather than the selection process is that a questioner may wait considerable time for an answer. If the search is fruitless, or possibly infinite, then this is wasted time. On the other hand, if relevant information is added to the answer set as it is discovered, each step in the search process may provide information that can be returned to a questioner. An early response that conveys a hypothetical answer might elicit a quick reply from a questioner that verifies or denies the hypothetical component of such an answer. This information can then be used by the answering agent to effectively prune the answer search space. Even if the questioner has no information relevant to the hypothetical component of an answer, providing such an answer gives the questioner the opportunity to put *this new question* (the question comprising the hypothetical component) to a different answering agent.

A third problem with imposing too many limitations on what may be considered as an answer is that it does not seem to accord with the way that we answer questions. When asked questions, particularly those that can be answered at different levels of generality, we often find that a number of possible answers "come to mind" from which we may choose. The manner in which this occurs is not of interest here; rather, what is of interest is the fact that we are often in the position of selecting from among answers, and our selection is related to what we know about the questioner.

For example, if someone points to a tree and asks me what kind of tree it is, what springs to mind may be that the tree is a white oak, thus (clearly) also an oak, and perhaps its Latin name: *quercus alba*. I will select from among these depending on the questioner. If it's someone who knows little about trees, I may respond that it is an oak tree. If it's someone from my woody plants class, I will probably respond with the Latin name. Consider another situation where answers come to mind and are not utilized. If a young child asks the question, "Where do babies come from", a number of answers may come to mind that are not used in responses in the interest of responding in an age-appropriate manner.

Another issue to consider is efficiency. If we pause for a while before responding to a question, the questioner may want to know what we are thinking (or if we are thinking at all). This can be considered a request for *intermediate information* that will assure the questioner that his or her question *is* being considered. Our response to such a request may involve sketchy or incomplete information. The reason such information would not have been offered thus far is that we *believe* we have not found a *suitable* answer. Nevertheless, if intermediate information is available, it may enable a questioner to answer his or her own question or to provide additional information that will help hasten the answer search.

3.4 Summary

The model of answering presented in this chapter is *reasoning-based* in its fundamental reliance on the process of reasoning through time. Clearly, there is room for parallel execution of some of the processes involved in answering: it need not be a purely sequential task. In particular, the analogy of the answering process to anytime reasoning discussed in the previous chapter means that answers that have been found are available at any time, and that, over time, the set of answers from which to choose will not become less informative.

The reasoning paradigm within which we explore this model is resolution refutation. The next chapter demonstrates that resolution refutation may be viewed as an answering process, where each resolvent generated has an interpretation as an answer.

Chapter 4

Resolvents as Answers

4.1 Overview

When a resolution theorem prover is used as a question-answering system, a question is expressed as a proposition in first order predicate calculus (FOPC). The proposition is negated and converted to clause form [Davis and Putnam, 1960], the resulting clauses are added to a *consistent* set of clauses, and a refutation proof is initiated [Robinson, 1965, Chang and Lee, 1973, Loveland, 1978, Kowalski, 1979]. A successful refutation is signaled by the derivation of the empty clause. When a refutation succeeds it is interpreted as a "yes" answer to the original question [Green and Raphael, 1968]. In the absence of a refutation, the search proceeds until all possibilities are exhausted: this potentially infinite search is in practice constrained by the finiteness of computational resources. The interpretation of the *failure* to find a proof is an answer of "no" when a closed world assumption is made [Reiter, 1978b], and "unknown" otherwise.

Using a theorem prover to provide "yes" answers to questions requires nothing above and beyond the machinery used for proving theorems. For example, given the question, "Is John home?", the first step is to express this question as the proposition "John is home." Answering the question now amounts to showing that this proposition follows from what is known. If a contradiction is derived by asserting that *it is not the case that John is home*, it means that the knowledge base contains the information that *John is home*; thus, the question has a "yes" answer.

Interest in expanding the question answering capabilities of resolution theorem provers led to a focus on two types of answers: *extensional*[1] and *intensional*.[2] Both of these answer types are associated with questions whose logical form contains *variables*. *Extensional* answers are associated with ground terms corresponding to variable substitutions that enable the generation of the empty clause. Like "yes" answers, extensional answers are associated with the existence of a refutation. We refer to the aforementioned ground terms as *satisfying individuals*, because they designate individuals in the knowledge base that satisfy the properties specified by a question. In addition to signaling a "yes" answer, these satisfying individuals provide the information that explains *why* the answer is "yes". When there is more than one substitution that leads to the empty clause, we refer to a *set of satisfying individuals*, each of which is associated with an extensional answer. The query feature in Prolog provides a user with the ability to generate the elements of this set sequentially. It is the set of satisfying individuals that is identified as a more informative answer than a simple "yes" in the early question answering work by Green and Raphael [Green and Raphael, 1968, Green, 1969a, Green, 1969b]. Some identify extensional answers with satisfying individuals, while others consider extensional answers to be the proposition associated with the original question with the satisfying individuals appropriately substituted for variables. Given the question, "Is someone at home?", extensional answers would *include* all entities that satisfy the properties "is a person" (someone) and "at home". If John and Mary are individuals that satisfy these properties, then $\{John,\ Mary\}$ is a set of satisfying individuals. "John", or "John is at home", and "Mary", or "Mary is at home", are extensional answers.

In contrast to extensional answers, *intensional answers* are *rules that characterize satisfying sets of individuals*. In the literature, the hallmark of an intensional answer is that it has the form of a rule. Intensional answers are not associated with the empty clause: their existence depends neither on the presence nor the absence of a refutation. Intensional answers depend on the presence of rules in a knowledge base that are relevant to a question. Given the question, "Is someone at home?", an intensional answer might be "All children are at home", provided this rule is contained in or inferable from the knowledge base.

Luckham and Nilsson [1971] extended Green's work on extensional answers to include answers

[1] [Kuhns, 1967, Green and Raphael, 1968, Green, 1969a, Green, 1969b, Luckham and Nilsson, 1971, Reiter, 1978a, Nilsson, 1980, McCune, 1994, Rich and Knight, 1991, Russell and Norvig, 1995, Luger and Stubblefield, 1997]

[2] [Reiter, 1978a, Nilsson, 1980, Cholvy and Demolombe, 1986, Imielinski, 1987, Motro, 1989, Cholvy, 1990, Motro, 1994, Gaasterland *et al.*, 1994, Borgida and McGuinness, 1996, Godfrey and Gryz, 1996]

containing universally quantified variables, but their answers do not conform to the traditional notion of intensional. Their answers are associated with proof: in the absence of a refutation, there are no answers. In addition, these answers do not have the form of a rule. The satisfying individuals with which these answers are associated are universally quantified variables rather than ground terms. Given the question, "Is someone at home?", an answer of this type could be, "Everyone is at home."

In addition to extensional and intensional answers, there has been some investigation of *conditional*, or *qualified*, answers. [3] These answers have been characterized as rules with all ground antecedents, where the antecedent is consistent with, but does not follow from, the knowledge base. Green mentions conditional answers as an interpretation of clauses containing literals in addition to answer literals, but does not fully explore the concept [Green, 1969a]. Sergot recognized the utility of conditional answers in terms of providing follow-up questions to a user. Given the question "Is John at home?", a conditional answer could be "If Mary is at home then John is at home", with the associated follow-up question, is "Is Mary at home?"

In this chapter, we propose the following three classes of answers that account for, and expand upon, previous work on question answering in a resolution theorem prover.

- **Specific Answers:** Specific answers correspond to Green's original answers and are equivalent to the category commonly referred to as *extensional answers*.
- **Generic Answers:** Generic answers include and expand upon the category of *intensional answers*. We include Luckham and Nilsson's non-ground answers in this category.
- **Hypothetical Answers:** Hypothetical answers include and expand upon what have previously been identified as *conditional* or *qualified* answers. According to our definition, the antecedent of a hypothetical answer is not purely ground: it may contain quantified variables.

Recognizing these three answer classes allows us to decouple the notion of *answer* from the notion of *proof*.

4.2 Preliminaries

This section introduces basic terminology used in the chapter and formalizes our underlying assumptions about the way in which a resolution theorem prover is employed for the question answering task.

[3] [Sergot, 1983, Vasey, 1986, Wolstenholme, 1988, Gilbert, 1988, Demolombe, 1992, Demolombe, 1997]

4.2.1 Questions

In the context of theorem proving, the *form* of a question is often assumed to be a ground or an existentially quantified formula [Green, 1969b, Reiter, 1978a, Rich and Knight, 1991, Ginsberg, 1993, Russell and Norvig, 1995, Luger and Stubblefield, 1997]. While Green [1969a] discusses different question forms in the context of applying the theorem-proving model to general problem solving, including universally quantified questions and questions with mixed quantification, his *answer literal* is designed for use with existentially quantified questions. Luckham and Nilsson [1971] do not limit the types of questions that can be asked: their proof-tree annotation technique provides answers for any questions expressible in FOPC *provided there is a corresponding refutation*. Like Luckham and Nilsson, we impose no limitations on the form of questions: our proposed characterization of answers is complete without regard to question form. However, we follow Green's strategy of employing an answer literal to keep track of answers as reasoning proceeds rather than Luckham and Nilsson's approach of annotating the proof tree once a refutation is found. This is not an arbitrary choice: one advantage of our expanded notion of answer is the fact that each reasoning step may be associated with the generation of an answer, and answers should be made available to return to a questioner as soon as they are generated. In addition, by decoupling the notions of answer and proof we are not tied to the paradigm of first obtaining a refutation, then finding answers.

4.2.2 Answer Literal

For the purposes of this discussion, we assume the use of an *answer literal* [Green, 1969b]. In Green's work, a question to be answered is negated and converted to clause form. This step may yield one or more clauses. An *answer literal* is then added to each of these clauses. The answer literal has the form $ANSWER(\overline{x})$, where $\overline{x}$ are the existentially quantified variables from the original question. Green's examples were concerned with existentially quantified variables only: later in this section, we introduce an expanded notion of answer literal that may be used with existentially and universally quantified variables. Answer literals are unresolvable because they are all positive; thus, they have no effect on the resolution process. Given a proposition of the form $\exists \overline{x}\, \overline{P}(\overline{x})$, where $\overline{x}$ are the existentially quantified variables from the original question and $\overline{P}$ are the predicates thereof, the corresponding clause form with an answer literal would be $\{\neg\, \overline{P}(\overline{x})\ , ANSWER(\overline{x})\}$. Note that this representation embodies the requirement that the original question is existentially quantified: upon negation, the existential quantifiers are converted to universal quantifiers. In addition, the nature of the connectives between multiple predicates is not explicitly specified in this notation: both *and* and *or* are permitted in questions. In converting to clause form, universal quantifiers are dropped entirely, leaving the variables unchanged: there are no Skolem functions. When an answer literal is employed, a refutation is signaled by the presence of a clause containing only answer literals: it is no longer possible to derive the empty clause. This is due to the fact that the empty clause can only be derived directly or indirectly from the clauses associated with a question. It was previously pointed out that, since the original knowledge base is required to be consistent, the only possible source of inconsistency is the clause(s) associated with the question. Green's procedure for using answer literals involves adding an answer literal to each clause associated with a question. Thus, all the clauses descended from these clauses contain answer literals, including the "empty" clause.

Demolombe [1992] describes a variant of Green's answer literal. If the original question is represented as the general formula $F(x)$, he adds the formula $\forall\, x\ (F(x)\ \rightarrow\ q(x))$ to his deductive database, where q is a new predicate symbol. Implicit in this description is that the original question is existentially quantified. The predicate q is analogous to Green's answer predicate, as are the arguments to q, namely, the variables from the original question. The advantage of Demolombe's approach is that his formula is directly converted to clause form and added to the database: there is no intermediate step of adding answer literals to each clause derived from the negated question.

Shapiro [1995] proposes a more direct way of incorporating an answer literal into clause representation that extends Demolombe's idea. Shapiro's technique replaces a question of the form $\exists \overline{x}\ \overline{P}(\overline{x})$ with $\forall \overline{x}\ (\overline{P}(\overline{x})\ \Rightarrow\ ANSWER(\overline{P}(\overline{x})))$. This formula is then converted to clause form, and resolution can proceed directly. A distinguishing feature of Shapiro's answer literal is the fact that the entire predicate structure of the question is included as the argument to the answer predicate. Note that this means that the answer predicate is second order. This is not problematic as long as predicates are treated as lists with arbitrary depth of nesting for purposes of substitutions. The answer literal is only for bookkeeping and is not involved in the reasoning process: reasoning is still first order with this answer literal form. Shapiro's answer literal obviates the need for substituting back into the original question in order to obtain a completely informative answer. In this sense, it resembles Luckham and Nilsson's [1971] annotation procedure. At each step of the proof tree, substitutions are made into the entire clause structure. As mentioned above, the advantage of employing an answer literal is that the answer is built up along with the proof; there is no need to revisit a proof tree to find an answer after the fact.

While both Green's and Shapiro's answer literals are defined for existentially quantified questions, Shapiro's answer literal is easily generalized so as to apply to all types of questions. Writing the original question and answer literal as an implication *before* converting to clause form ensures that any substitutions of Skolem functions for variables will be made both in the question as well as in the answer predicate. Skolem functions are inevitable once universal quantification in questions is admitted. Note that Green's approach to incorporating answer literals into clauses does not handle Skolem functions.

A generalization of Shapiro's answer literal:

Given a proposition in FOPC, proceed as follows:

1. Put the proposition in PNF so that it has the form $\overline{Q}\overline{x}\,\overline{P}(\overline{x})$, where $\overline{Q}$ are all of the quantifiers from the original question, $\overline{x}$ are quantified variables, and $\overline{P}$ are predicates.
2. Swap the universal and existential quantifiers. This is justified because it reflects the fact that the proposition is negated before it is added to the knowledge base. As the negation is pulled through the quantifiers to the matrix, it has the effect of swapping universal with existential by the standard quantifier definitions.
3. Replace the matrix $\overline{P}(\overline{x})$ with $[\overline{P}(\overline{x}) \Rightarrow ANSWER(\overline{P}(\overline{x}))]$
4. Convert the resulting proposition to clause form and proceed directly with resolution

A question that has been transformed according to steps 1–3 (above) we term an *augmented negation*. Clauses generated from the augmented negation (step 4) we term *question clauses*.

For the most basic question types, Table 4.1 shows a proposition corresponding to a question, the associated augmented negation, and the associated question clause. The notation employed in many of our examples is non-standard. The form $(P\ x_1,\ x_2, \ldots,\ x_n)$ should be interpreted as the standard $P(x_1,\ x_2, \ldots,\ x_n)$.

The use of answer literals is critical to our characterization of the answer classes *specific*, *generic*, and *hypothetical*. Given their importance with respect to our formalization, the following must be shown:

[4] $S10$ is a Skolem constant
[5] $S513$ is a Skolem constant
[6] $S521$ is a unary Skolem function

Table 4.1: Questions, Augmented Negations, and Question Clauses

Question Proposition	*Augmented Negation*	*Question Clause*
$(dog\ fido)$	$[(dog\ fido) \Rightarrow (ANSWER\ (dog\ fido))]$	$\{(\neg(dog\ fido)), (ANSWER(dog\ fido))\}$
$\exists\, x\ (dog\ x)$	$\forall\, x\ [(dog\ x) \Rightarrow (ANSWER\ (dog\ x))]$	$\{(\neg(dog\ x)), (ANSWER(dog\ x))\}$
$\forall\, x\ (dog\ x)$	$\exists\, x\ [(dog\ x) \Rightarrow (ANSWER\ (dog\ x))]$	$\{(\neg(dog\ S10)), (ANSWER(dog\ S10))\}$[4]
$\forall\, x\, \exists\, y\ (R\ x\ y)$	$\exists\, x\, \forall\, y\ [(R\ x\ y) \Rightarrow (ANSWER\ (R\ x\ y))]$	$\{(\neg\ (R\ S513\ x)), (ANSWER(R\ S513\ x))\}$[5]
$\exists\, x\, \forall\, y\ (R\ x\ y)$	$\forall\, x\, \exists\, y\ [(R\ x\ y) \Rightarrow (ANSWER\ (R\ x\ y))]$	$\{(\neg\ (R\ x\ (S521\ x))),$[6] $(ANSWER(R\ x\ (S521\ x)))\}$

1. If answer literals are ignored in question clauses, the question clauses correspond exactly to what they would be under normal circumstances: the clause form of the negated question. This is formalized in Theorem 4.1.

2. All and only the question clauses and their descendents contain answer literals. This is formalized in Theorem 4.2.

The first of these affirms that the presence of answer literals does not alter the new clauses with which reasoning will proceed. The second assures that all and only clauses descended from a question are appropriately marked with answer literals.

Theorem 4.1 (Logical Equivalence of Reasoning with Question Clauses) *In the context of resolution refutation, reasoning from the question clauses associated with a given question plus an existing, consistent knowledge base $\mathcal{K}$ is logically equivalent to reasoning from the clause form of the negation of the question plus $\mathcal{K}$.*

Proof: Any proposition can be translated into an equivalent prenex normal form. The standard approach in resolution refutation is to (1) negate the question, (2) translate the resulting proposition to clause form, and (3) add the resulting clauses to a consistent knowledge base. There is no logical difference in the following: (1) translate the question into prenex normal form, (2) negate the proposition (in PNF), (3) continue translation to clause form, (4) add the resulting clauses to a consistent knowledge base.

Case 1: Reasoning proceeds from the clause form of the negated question. Proceed as follows:

1. Translate the question into prenex normal form, yielding a formula of the form $\overline{Q}\overline{x}\,\overline{P}(\overline{x})$, where $\overline{Q}$ are existential or universal quantifiers, $\overline{x}$ are quantified variables, and $\overline{P}$ are predicates.

2. Negate the formula, yielding $\neg(\overline{Q}\overline{x}\,\overline{P}(\overline{x}))$

3. Push the negation through the quantifiers so that it immediately precedes the matrix, yielding the dual of the quantifier sequence.

4. Remove all quantifiers by (1) replacing existentially quantified variables with Skolem functions and dropping the existential quantifiers, and (2) dropping the universal quantifiers, yielding $\neg(\overline{P}(\overline{x}',\overline{s}))$, where $\overline{s}$ represents Skolem functions of zero or more arguments and $\overline{x}'$ are the remaining variables (i.e. those that were universally quantified in the negation).

5. Write this in clause form as $\{\ \neg(\overline{P}(\overline{x}',\overline{s}))\}$

Case 2: Reasoning proceeds from the question clauses associated with the question. Proceed as follows:

1. Translate the question into prenex normal form so that it has the form $\overline{Q}\overline{x}\,\overline{P}(\overline{x})$, where $\overline{Q}$ are existential or universal quantifiers, $\overline{x}$ are quantified variables, and $\overline{P}$ are predicates. This is identical to step 1 of **Case 1**.

2. Change each universal quantifier to an existential quantifier, and vice versa. This is precisely what happens in step 3 of **Case 1** when the negation is pushed through the quantifiers. The resulting formula is $(\overline{Q})'\overline{x}\,\overline{P}(\overline{x})$.

3. Replace the matrix $\overline{P}(\overline{x})$ with $[\overline{P}(\overline{x}) \Rightarrow ANSWER(\overline{P}(\overline{x}))]$. The resulting formula is:
$$(\overline{Q})'\,\overline{x}\,[\overline{P}(\overline{x}) \Rightarrow ANSWER(\overline{P}(\overline{x}))]$$
which is logically equivalent to:
$$(\overline{Q})'\,\overline{x}\,[\neg(\overline{P}(\overline{x})) \vee ANSWER(\overline{P}(\overline{x}))]$$

4. Remove all quantifiers by (1) replacing existentially quantified variables with Skolem functions and dropping the existential quantifiers, and (2) dropping the universal quantifiers, yielding $\neg(\overline{P}(\overline{x}',\overline{s})) \vee ANSWER(\overline{P}(\overline{x}',\overline{s}))$, where $\overline{s}$ represents Skolem functions of zero or more

arguments and $\overline{x}'$ are the remaining variables (i.e. those that were universally quantified in the negation).

5. Write this in clause form as $\{\ \neg(\overline{P}(\overline{x}',\overline{s})),\ ANSWER(\overline{P}(\overline{x}',\overline{s}))\}$

6. The resulting clause is identical to that obtained in the normal course of resolution refutation excepting the presence of an additional disjunct, namely, the answer literal. The answer literals have no effect on the proof procedure, as described by Green [1969b, page 190]:

> When a new clause is created, each *ANSWER* literal in the new clause is instantiated in the same manner as any other literal from the parent clause. However, the *ANSWER* literal is treated specially; it is considered to be invisible to resolution in the sense that no literal is resolved against it ... The addition of the *ANSWER* literal to the clauses representing the negation of the theorem does not affect the completeness of this modified proof procedure. The theorem prover generates the same clauses, except for the *ANSWER* literals, as the conventional theorem prover.

Because the question clauses are identical to those generated by negating a question excepting the presence of answer literals, and the answer literals are invisible to the resolution refutation proof procedure, the question clauses are logically equivalent to the clause form of the negation of the question. □

Thus, we are free to incorporate answer literals into the reasoning process according to the procedure we presented for generating question clauses from questions. The incorporation of these literals has no effect on the reasoning process. The next theorem addresses the problem of exactly which clauses contain answer literals.

Theorem 4.2 (Propagation of Answer Literals) *A clause contains an answer literal if and only if it is a question clause or is a descendent of a question clause.*

Proof:

1. **If a clause contains an answer literal, then it is either a question clause or is descended from a question clause**.

   The proof is by contradiction. Assume that there is a clause containing an answer literal that is neither a question clause nor the descendent of a question clause. Let $\mathcal{KB}$ represent the knowledge base with respect to which the question is asked; $\mathcal{QC}$ represent the question clauses; and $\mathcal{C}$ represent a clause containing an answer literal, where $\mathcal{C}$ is neither a question clause nor a descendent of a question clause. Thus, $\mathcal{C} \in \mathrm{Th}(\mathcal{KB})$, and $\mathcal{C} \notin \mathrm{Th}(\mathcal{KB} \cup \mathcal{QC})$. No clause in $\mathcal{KB}$ contains an answer literal, because answer literals are introduced into the reasoning process in the clauses that constitute $\mathcal{QC}$. But if no clause in $\mathcal{KB}$ contains an answer literal, then no clause in $\mathrm{Th}(\mathcal{KB})$ can contain an answer literal. Therefore, $\mathcal{C} \notin \mathrm{Th}(\mathcal{KB})$. $\square$

2. **If a clause is a question clause or is descended from a question clause, then it contains at least one answer literal**.

   The proof is by induction on the distance of a clause $\mathcal{C}$ from a question clause, where distance is defined by the number of intermediate resolvents in a chain of reasoning from a question clause to $\mathcal{C}$. This number is equivalent to the number of nodes in a resolution proof tree between a question clause and $\mathcal{C}$.

   (a) Question clauses are distance zero from themselves, and all question clauses contain at least one answer literal.

   Question clauses are generated using the following algorithm:

   i. Translate the question into prenex normal form so that it has the form $\overline{Q}\overline{x}\ \overline{P}(\overline{x})$, where $\overline{Q}$ are existential or universal quantifiers, $\overline{x}$ are quantified variables, and $\overline{P}$ are predicates.
   ii. Change each universal quantifier to an existential quantifier, and vice versa.
   iii. Replace the matrix $\overline{P}(\overline{x})$ with $[\overline{P}(\overline{x}) \Rightarrow ANSWER(\overline{P}(\overline{x}))]$
   iv. Convert the resulting proposition to clause form and proceed directly with resolution

   Step iii of this procedure constructs the *augmented negation*. It is from this form, in Step iv, that the question clauses are produced. The quantifier-free matrix of every augmented negation has the following general form:

$$
\begin{aligned}
& [P \Rightarrow ANSWER(P)] \\
\equiv\ & (\neg P \vee ANSWER(P)) \\
\equiv\ & (\neg P\ in\ CNF \vee ANSWER(P)) \\
\equiv\ & [(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \vee ANSWER(P)], \text{where } (P_1 \wedge P_2 \wedge \ldots \wedge P_n) \\
& \text{is the CNF of } P.
\end{aligned}
$$

$$\equiv \quad [(P_1 \vee ANSWER(P)) \wedge (P_2 \vee ANSWER(P)) \wedge \ldots \wedge (P_n \vee ANSWER(P))]$$

$$\equiv \quad \{P_1, ANSWER(P)\}, \{P_2, ANSWER(P)\}, \ldots ,\{P_n, ANSWER(P)\}$$

in clause form.

Clearly every conjunct of the CNF of the matrix contains an answer literal. On conversion to clause form, each conjunct is associated with a single clause; therefore, every clause generated by this algorithm will also contain an answer literal. These are precisely the question clauses.

(b) Assume that clauses at distance n from question clauses all contain at least one answer literal.

Clauses at distance $n + 1$ from a question clause all have as a parent one of the clauses at distance n from a question clause. We assumed that all of these clauses contain answer literals. Since answer literals (by definition) are not resolvable, they will not be resolved away when a parent clause containing them participates in resolution; therefore, they will appear in any resolvents generated: namely, the clauses at distance $n+1$ from the question clauses. □

Therefore, not only do answer literals provide a step by step view of substitutions made as reasoning proceeds, but they also provide a way of *tagging* clauses descended from the question clauses. The relationship between answers and proof is clear: a clause containing only answer literals corresponds to the empty clause in an ordinary proof. We make use of answer literals in defining relevance:

Definition 4.1 Relevance. *Given a knowledge base $\mathcal{KB}$, a question $\mathcal{Q}$, and question clauses $\mathcal{QC}$, a clause $\mathcal{C}$ is relevant to $\mathcal{Q}$ if and only if the following is true:*

$$\mathcal{C} \in Th(\mathcal{KB} \cup \mathcal{QC}) \setminus Th(\mathcal{KB})$$

In other words, question clauses and their descendents are relevant to a question. By Theorem 4.2, this property is indicated by the presence of at least one answer literal in a clause.

Finally, we provide a broad definition of answer.

Definition 4.2 Answer. *All relevant clauses except for the question clauses are answers.*

The import of this definition is that our notion of answer fails to accord with answer definitions proposed by linguists and philosophers. It does so by classifying information that does not necessarily answer a question as "answers". The efficacy of these "not really an answer" answers, that we term hypothetical answers, ranges from unhelpful to providing information necessary for the questioner to connect something he or she knows to the question, and in doing so discover an answer him or herself. This definition does, however, capture the way in which people respond to questions in that they will try to be as informative as possible. Whether or not hypothetical answers are "answers", they are valid responses to questions, and as such should be regarded by a question answering system.

4.2.3 Summary

This preliminary section has provided the basic definition of what we consider to be an *answer*. We have identified answers with non-question clauses containing one or more answer literals. The question clauses are not considered answers: if they were, we would have to say that the vacuous "If John is at home then John is at home" is an answer to the question, "Is John at home?"

The next three sections of this chapter describe the answer classes *specific*, *generic*, and *hypothetical*. These classes partition the set of *answers*, as defined above. In describing the form of answers, we assume that an answer literal has been employed in resolution as described above.

4.3 Specific Answers

We identify specific answers with what have previously been characterized as *extensional answers*. In this section, we describe the form of a specific answer in our framework and provide some examples of specific answers. A specific answer is associated with a substitution that replaces all the variables in the proposition associated with the original question with *non-Skolem* constants, thereby producing a ground proposition. The prohibition on Skolem constants is in keeping with Luckham and Nilsson's work [1971], in which they prove that Skolem constants and functions may, outside the context of active reasoning (i.e., during a proof-tree annotation or in an answer), be interpreted as universally quantified variables. Note that Skolem symbols replace variables that are universally quantified in the original question.

Specific answers are associated with proof: it is the derivation of the empty clause, or in our

Table 4.2: Simple Knowledge Base

Logic Form	*English Gloss*	*Clause Number*	*Clause*
$\forall x\ (\forall y\ (cat(x)\ \wedge\ fish(y)\ \rightarrow\ likestoeat(x,y)))$	cats like to eat fish	1.	{(¬ (CAT ?5)),(¬ (FISH ?7)), (LIKESTOEAT ?5 ?7)}
$\forall x\ (calico(x)\ \rightarrow\ cat(x))$	calicos are cats	2.	{(¬ (CALICO ?11)), (CAT ?11)}
$\forall x\ (tuna(x)\ \rightarrow\ fish(x))$	tunas are fish	3.	{(¬ (TUNA ?15)), (FISH ?15)}
$tuna(Charlie)$	Charlie is a tuna	4.	{(TUNA CHARLIE)}
$tuna(Herb)$	Herb is a tuna	5.	{(TUNA HERB)}
$calico(Puss)$	Puss is a calico	6.	{(CALICO PUSS)}

Table 4.3: Question

Logic Form	*English Gloss*	*Clause Number*	*Question Clause*
$\exists\ x\ (likestoeat\ Puss\ x)$	what does Puss like to eat	7.	{(¬ (LIKESTOEAT PUSS ?46)), (ANSWER (LIKESTOEAT PUSS ?46))}

case, of a clause containing only answer literals, that signals a possible specific answer. In order to qualify as a specific answer, there may be no variables and no Skolem functions in the answer literals.

Table 4.2 shows a simple knowledge base from Rich and Knight [1991, p.192] that is used to demonstrate specific answering. With respect to this KB, Rich and Knight [1991, p. 192] pose the question, "What does Puss like to eat?" A logic representation of this question and corresponding question clause are shown in Table 4.3, where the question clause is numbered as clause *7* following the numbering above.

Given the knowledge base shown in Table 4.2 and the question clause shown in Table 4.3, a refutation proof is initiated with steps shown in Table 4.4. Note that variables in clauses have been renamed after each resolution step.

The form of a specific answer is $\overline{P}(\overline{a})$, where $\overline{P}$ are the predicates in the question and $\overline{a}$ are ground terms. In this example, the two specific answers are:

$$(LIKESTOEAT\ PUSS\ CHARLIE)$$

[7] In Table 4.4, R stands for "resolve", and the numbers following R are the numbers of the parent clauses.

Table 4.4: Specific Answers: Refutation Proof Example

Clause Num.	*Clause*	*Justification*	*Substitution*
1.	{(¬ (CAT ?5)),(¬ (FISH ?7)), (LIKESTOEAT ?5 ?7)}	assumption	
2.	{(¬ (CALICO ?11)), (CAT ?11)}	assumption	
3.	{(¬ (TUNA ?15)), (FISH ?15)}	assumption	
4.	{(TUNA CHARLIE)}	assumption	
5.	{(TUNA HERB)}	assumption	
6.	{(CALICO PUSS)}	assumption	
7.	{(¬ (LIKESTOEAT PUSS ?46)), (ANSWER (LIKESTOEAT PUSS ?46))}	question	
8.	{(ANSWER (LIKESTOEAT PUSS ?150)), (¬ (CAT PUSS)),(¬ (FISH ?150)) }	R 1,7 [7]	{ ?46/?7, PUSS/?5 }
9.	{(ANSWER (LIKESTOEAT PUSS ?270)), (¬ (CALICO PUSS)), (¬ (FISH ?270)) }	R 8,2	{PUSS/?11}
10.	{(ANSWER (LIKESTOEAT PUSS ?285)), (¬ (FISH ?285)) }	R 9,6	{ }
11.	{(ANSWER (LIKESTOEAT PUSS ?523)), (¬ (TUNA ?523)) }	R 10,3	{ ?285/?15}
12.	{(ANSWER (LIKESTOEAT PUSS CHARLIE)) } **QED**	R 11,4	{ CHARLIE/?523}
13.	{(ANSWER (LIKESTOEAT PUSS HERB)) } **QED**	R 11,5	{ HERB/?523}

$$(LIKESTOEAT\ PUSS\ HERB)$$

English glosses of these are *Puss likes to eat Herb* and *Puss likes to eat Charlie*, respectively.

With respect to existentially quantified questions, any specific answer entails the original question: all that is required to prove an existentially quantified formula is to find a "witness", which is precisely a specific individual. In this example, we have two specific answers, both of which entail the original question:

$$(LIKESTOEAT\ PUSS\ CHARLIE) \Rightarrow \exists\, x\ (LIKESTOEAT\ PUSS\ x)$$
$$(LIKESTOEAT\ PUSS\ HERB) \Rightarrow \exists\, x\ (LIKESTOEAT\ PUSS\ x)$$

Because specific answers are associated with the existence of a refutation proof, a formal definition of specific answer might be *a clause that contains only answer literals*: such clauses signal the completion of a proof. This is problematic because of the results obtained by Luckham and Nilsson [Luckham and Nilsson, 1971], who also associate the presence of *non-ground* answer literals with a proof. We first define what it means to be a specific answer clause.

Definition 4.3 Specific Answer Clause. *A specific answer clause is a clause with the following properties:*

1. *It contains at least one answer literal.*
2. *It contains no non-answer literals.*
3. *All literals are ground.*
4. *There are no Skolem functions or constants present in literals.*

We will say that specific answer clauses are *associated* with specific answers.

Definition 4.4 Specific Answer. *A specific answer is the disjunction of the arguments of all of the answer literals in a specific answer clause.*

In terms of their role, specific answers *mention* individuals that satisfy a question. Specific answers are useful when it is important to know the identity of these individuals. There are two primary reasons that account for the pervasiveness of specific answers. First, specific answers are associated with the presence of a proof, which is clearly the focus of theorem proving. Question answering in AI emerged from the theorem proving paradigm, and the norms of that paradigm have persisted. Second, existentially quantified questions are extremely common, and specific answers are inextricably tied to existential quantification.

Specific answers persist as the most commonly considered form of answers despite the fact that they are not always appropriate. One problem pointed out by Webber [1986, page 379] is related to answers that are veritable laundry lists of individuals: "... an enumeration of those elements in a set that have property **P** implicates that there are also set elements with property $\neg$ **P**, even though the enumeration is consistent with it covering all the elements." When questions involving universal quantification are of interest, it may be misleading to provide a specific answer unless it is qualified with a specification that the list comprises *all* such individuals. This type of specification is referred to by Belnap and Steel [1976] as a "completeness-claim". This shortcoming of specific answers points to the need to consider other types of answers.

4.4 Generic Answers

If clauses containing only ground answer literals are associated with specific answers, which clauses might be associated with generic answers? The idea of "generics" is that they capture generalities that hold for groups of individuals or classes. Such generalities are formally captured by rules. A clause containing answer literals plus some non-answer literals can be written in rule form by considering the non-answer literals as the rule antecedent and the answer literals as the consequent. These answers are clearly not associated with the presence of a proof, because they are not associated with clauses containing only answer literals. The question is whether all such clauses represent generic answers, and, if not, what distinguishes those clauses that do. In addition, we ask the question of whether there are some clauses associated with generic answers that *are* associated with proof.

In his description of *intensional answering*, Imielinski [1987, page 229] proposes the use of rules to facilitate intelligent querying, stating that, "Conceptually, rules are often more informative and easier to comprehend than the corresponding sets of derived tuples." In the deductive database model, rules can be retrieved from the $\mathcal{IDB}$ (intensional database) and the $\mathcal{IC}$s (integrity constraints) [Reiter, 1978a, Imielinski, 1987, Motro, 1994]. These sets are generally small relative to the size of the $\mathcal{EDB}$ (extensional database), which may contain a huge number of facts [Reiter, 1978a, Imielinski, 1987].

The problem lies in formally characterizing rules that appropriately describe answers to a query. Imielinski's definition is [1987, page 231]:

> Formally, the rules in the answer for a query should be either expressed totally in terms of the *query predicate* i.e., the predicate defined by the query or by the query predicate and some other additional predicates which occur in the database. The choice of these additional predicates may be left to the user. The query predicate must appear both in the consequent and in the body of the rule (otherwise the query definition could trivially form such a rule).

The query predicate to which Imielinski refers is the predicate contained in the user question. Note that conjunctive and disjunctive queries are disallowed by this definition due to the restriction that queries contain a single predicate. Imielinski's definition is directly related to the fact that he is working in a database environment, answering queries with rules from the $\mathcal{IDB}$ and the $\mathcal{IC}$s. In this environment, a rule may be matched and returned as an answer. This differs from the theorem-proving paradigm, where a rule will be *reasoned with*, and the matching components will be resolved away. This means that the query predicate itself will generally *not* appear in the antecedent of a rule given as an answer in a theorem prover.

The following example is taken from Imielinski [1987]. If a query asking which courses Imielinski can teach is posed to a database, the query predicate $\mathcal{Q}$ is *teach Imielinski* (i.e. Imielinski can teach). This would appear initially not as a single predicate, but a predication of *teach* over argument *Imielinski*. However, in this paradigm this information is combined into the single "predicate" *teach Imielinski*. An argument to this *new* predicate would be a course that Imielinski can teach.

If the rule *if someone can teach a course, then he or she can teach all prerequisites for that course* is present in the $\mathcal{IDB}$, then the following is an answer expressed in terms of the query predicate and the rule:

$$[\mathcal{Q}(x) \wedge \text{Prerequisite}(x, y)] \rightarrow \mathcal{Q}(y)$$

A gloss is *if Imielinski can teach x, and y is a prerequisite for x, then Imielinski can teach y*. The query predicate is in evidence in both the body and the consequent of the given rule as mandated by Imielinski's definition.

Cholvy and Demolombe [1986] define intensional answers as those that provide sufficient conditions for tuples to satisfy a query. This is in accord with the types of rules produced as answers in

Table 4.5: Generic Answers and Resolution

Clause Number	*Clause*	*Justification*	*Substitution*
8.	{(ANSWER (LIKESTOEAT PUSS ?150)), (¬ (CAT PUSS)),(¬ (FISH ?150)) }	R 1,7	{ ?46/?7, PUSS/?5 }
9.	{(ANSWER (LIKESTOEAT PUSS ?270)), (¬ (CALICO PUSS)), (¬ (FISH ?270)) }	R 8,2	{PUSS/?11}
10.	{(ANSWER (LIKESTOEAT PUSS ?285)), (¬ (FISH ?285)) }	R 9,6	{ }
11.	{(ANSWER (LIKESTOEAT PUSS ?523)), (¬ (TUNA ?523)) }	R 10,3	{ ?285/?15}

a theorem prover and reflects the fact that their work was done in a theorem-proving environment. The antecedent of a rule can be seen as specifying necessary conditions for class membership, while the consequent (the query) provides a class description. For example, given the query *what barks?* , a rule of the form $dog(x) \rightarrow barks(x)$ defines a sufficient condition for an entity to be a member of the class of barking things, namely, that it is a dog.

We defer a definition of generic answer until some examples have been examined. To illustrate generic answering, consider once again the example from Rich and Knight [1991] from the previous section. Clauses generated during the proof, excepting those associated with specific answers, are shown in Table 4.5.

These clauses contain answer literals, so they are descendents of the question clauses. They contain other literals in addition to the answer literals, so they represent rules. Clauses 8 and 9 are *syntactically subsumed* by 10, thus are neither as informative nor as interesting as 10. Syntactic subsumption between clauses is defined as follows:

Definition 4.5 Syntactic subsumption. *A clause A syntactically subsumes a clause B if there is a substitution* σ *such that the literals in A*σ *are a subset of the literals in B.*

Clauses 10 and 11 are descendents of the question clauses and are not syntactically subsumed by other clauses in Table 4.5. Subsumption is often considered a taxonomic phenomenon [Woods, 1991], but in our discussions it is important to separate syntactic from taxonomic subsumption. Note that clause 10 *taxonomically* subsumes clause 11. We do not consider taxonomic subsumption when determining what *counts* as an answer, though it may be considered in selecting from among answers. The reason for this is that the taxonomy of the questioner may differ from that of the answering agent. A taxonomically subsumed answer may be the one answer that a questioner understands. Without constructing a realistic user model, or making assumptions about the knowledge of a questioner, it is impossible to know which answers provide the best information to a questioner.

Table 4.6 shows the relationship between the clause form of a generic answer and its associated (unquantified) rule *form*. When rewriting a clause as a rule, the consequent of the rule is the disjunction of the arguments of the answer literals, and the antecedent is the conjunction of the negation of the non-answer literals. The rule forms for clauses 10 and 11 are shown in Table 4.6.

Table 4.6: Clauses as Rules

Clause Number	*Clause*	*Rule*
10.	{(ANSWER (LIKESTOEAT PUSS ?285)), (¬ (FISH ?285)) }	(FISH ?285) ⇒ (LIKESTOEAT PUSS ?285)
11.	{(ANSWER (LIKESTOEAT PUSS ?523)), (¬ (TUNA ?523)) }	(TUNA ?523) ⇒ (LIKESTOEAT PUSS ?523)

It is generally accepted that such rules should be written as closed sentences, where all of the variables are universally quantified [Cholvy and Demolombe, 1986, Imielinski, 1987, Motro, 1989, Cholvy, 1990]. Universal quantification is appropriate, because all variables in clauses are implicitly universally quantified. The final forms for clauses 10 and 11 are shown in Table 4.7.

Table 4.7: Rules as Answers

Clause	*Answer*
10.	∀ ?285 [(FISH ?285) ⇒ (LIKESTOEAT PUSS ?285)]
11.	∀ ?523 [(TUNA ?523) ⇒ (LIKESTOEAT PUSS ?523)]

English glosses for these rules are *Puss likes to eat fish* and *Puss likes to eat tuna*, respectively. Each of these rules characterizes a set of specific answers. The first, associated with clause 10, means that any individual that is a fish is a (specific) answer, while the second, associated with clause 11, means that any individual that is a tuna is a (specific) answer. This accords with Cholvy and Demolombe's [1986] definition of generic answers as specifying "sufficient" conditions for specific answer tuples. It does not meet the criterion requiring query predicates to appear in the rule body proposed by Imielinski [1987].

4.4.1 Taxonomic Subsumption

The relationships between rules comprising answers is important. According to the knowledge base introduced in Table 4.2, *all tuna are fish* (a rule in the KB). By this rule, clause 10 *taxonomically* subsumes clause 11. This means that the set of specific answers denoted by clause 11 is a subset of the set of specific answers denoted by clause 10. Despite this denotational redundancy, clauses 10 and 11 convey different information from one another. As previously mentioned, the taxonomic relationship between these rules is in the knowledge base of the *answering agent*; it may not be the case that the *questioner* possesses this knowledge. It is conceivable that a questioner may know something about fish and nothing whatsoever about tuna. Rules that are related by subsumption in a taxonomic hierarchy are at different levels of generality: providing information at different levels of generality is an important part of AI question answering [Cholvy, 1990], and is at the heart of answering in description logic frameworks [Borgida and McGuinness, 1996]. For these reasons we consider information at all levels to be important for question answering.

4.4.2 An Unconstrained Generic Answer

The idea that rules and generic answers are the same, while initially attractive, fails to include everything that is appropriately classified as generic, and includes rules that are clearly not generic. Examples from Nilsson [1980] provide generic answers whose form is the same as the initial question and whose presence is indicated by a proof. This type of answer contains non-ground terms, hence is not considered to be a specific answer. In order to illustrate this type of answer, we examine one of Nilsson's examples [1980, pages 184–185]. Table 4.8 shows the initial knowledge base containing information about parents and children:

Nilsson [1980, page 184] poses the question, "For any x, who is the parent of x?", with respect to this KB. The logic form of the question and corresponding question clause are shown in Table 4.9, where the question clause is numbered as clause 3 following the numbering above. A refutation

[8]Note that the monadic function *p* (lower-case p) from the original text is here rewritten as the function PARENT-OF to distinguish it from *P*, which represents the binary predicate *is a parent of*. The function will return a parent of its argument, while the predicate asserts that first argument is a parent of the second argument.

Table 4.8: Parent and Child

Logic Form	*English Gloss*	*Clause Num.*	*Clause*
$\forall x\ (C\ (x\ ,\ p(x)))$	for all x, x is the child of p(x)	1.	{(C ?4 (PARENT-OF ?4))}[8]
$\forall x\ \forall y\ ((C\ (x\ ,\ y))\ \Rightarrow\ (P\ (y\ ,\ x)))$	for all x and y, if x is the child of y, then y is the parent of x	2.	{(¬ (C ?10 ?12)), (P ?12 ?10)}

proof that answers this question with respect to the the "Parent and Child" knowledge base is shown in Table 4.10.

Table 4.9: Question

Logic Form	*English Gloss*	*Clause Number*	*Question Clause*
$\forall\ x\ \exists\ y\ (P\ (y\ ,\ x))$	For any x, who is the parent of x	3.	{(¬ (P ?26 S24)), (ANSWER (P ?26 S24))}

Table 4.10: Generic Answers: Refutation Proof Example

Clause Number	*Clause*	*Justification*	*Substitution*
1.	{(C ?4 (PARENT-OF ?4))}	assumption	
2.	{(¬ (C ?10 ?12)), (P ?12 ?10)}	assumption	
3.	{(¬ (P ?26 S24)), (ANSWER(P ?26 S24))}	question	
4.	{(ANSWER (P ?60 S24)), (¬ (C S24 ?60))}	R 3,2	{S24/?10, ?12/?26}
5.	{(ANSWER (P (PARENT-OF S24) S24))} **QED**	R 4,1	{(PARENT-OF S24) /?60, S24/?4}

In the example shown, "S24" is a Skolem constant. Luckham and Nilsson [1971] offer a proof that Skolem constants and functions in an answer statement (corresponding to the answer literal) can be replaced by universally quantified variables. The following quote summarizes their result [1971, page 37]:

> Well known inference principles justify the replacement of all terms beginning with

> Skolem functions occurring in the answer statement but not in the axioms by *new* universally quantified variables.

In fact, not only the answer statement, but all clauses related by a reasoning chain to the negated question may be treated in this manner. Nilsson [1980, page 185] states that:

> ... in the answer-extracting process it is correct to replace any Skolem functions in the clauses coming from the negation of the goal wff by new variables.

The important point is that this replacement does not affect the reasoning process. This is achieved by Luckham and Nilsson [1971] by their refutation-tree annotation procedure, which effectively constructs answer literals *after the fact*. In our procedure, answers that are rewritten and returned to a questioner are *not* added back into the reasoner; therefore, the answer clause from Table 4.10 can be expressed replacing the Skolem constants with universally quantified variables:

$$\forall x\ (P(parent\text{-}of\ x)\ x)$$

This closed sentence may be interpreted to mean that, for all x, the parent-of function applied to x yields the parent of x, which is an appropriate answer given the knowledge base and question.

Answers such as this, where an answer literal contains variables at the time a proof is completed and these variables are not shared with a rule antecedent, we term *unconstrained generic answers*. As previously discussed, the antecedent of a generic answer describes sufficient conditions for individuals to be considered as answers to a question. Variables occurring *only* in answer literals are completely unconstrained: in other words, these variables may be instantiated by any individual in the domain.

Sorted Logic and Unconstrained Variables

The notion of a completely unconstrained variable is a theoretical one. In practice we have in mind *some* restricted set of individuals even when we make sweeping generalizations. In sorted logic [Gallier, 1986, Plaisted, 1993] every variable is associated with a sort, which is analogous to a type in a computer programming language. When every variable is constrained in some manner, the notion of unconstrained is construed as unconstrained with respect to a particular set of individuals. When a partial order on sorts is incorporated into the logic there can be subsorts of sorts, allowing for the definition of a taxonomy of sorts. In such a case it is possible to define a sort for which all other sorts are subsorts: this all-inclusive sort would correspond to a top-level ontology category of "entity". In the context of a taxonomy, our notion of unconstrained variable would correspond to the most general type represented in the taxonomy.

4.4.3 A Rule that is Not a Generic Answer

In order to see that there are relevant rules generated in the course of resolution that are *not* generic answers, consider the question *What does Boots like to eat?* in conjunction with the knowledge base in Table 4.2. Note that "Boots" is not mentioned in the knowledge base. Two of the relevant clauses generated are shown in Table 4.11. Rewriting these clauses as rules yields the following:

- $(cat\ Boots) \Rightarrow (likestoeat\ Boots\ Herb)$

  If Boots is a cat then Boots likes to eat Herb.

- $\forall\, x\, [((tuna\ x) \wedge (cat\ Boots)) \Rightarrow (likestoeat\ Boots\ x)]$

  If Boots is a cat then Boots likes to eat tuna.

Neither of these rules has the flavor of a generic, which is meant to provide an intensional set description that can be used to identify individuals in a domain. Lacking variables, the first rule is clearly not generic, yet it contains a non-answer literal, which excludes it from the specific answer category. The second rule contains a variable shared between the argument of the answer literal and one of the other literals, and the piece of information conveyed by those two literals *is* generic. The problem is the presence of the other ground, non-answer literal: its presence means that this answer is not purely generic. In Section 3.5, we will see that these rules are *hypothetical answers*.

Table 4.11: Clauses as Non-Generic Rules

Clause	*Rule*
{(ANSWER (LIKESTOEAT BOOTS HERB), (¬ (CAT BOOTS)) }	(CAT BOOTS) ⇒ (LIKESTOEAT BOOTS HERB)
{(ANSWER (LIKESTOEAT BOOTS ?285)), (¬ (TUNA ?285)),(¬ (CAT BOOTS)) }	((TUNA ?285) ∧ (CAT BOOTS)) ⇒ (LIKESTOEAT BOOTS ?285)

4.4.4 A Characterization of Generic Answers

The purpose of a generic answer is to provide a characterization of individuals that satisfy a given set of criteria. The genericity is indicated by the presence of variables in answer literals: these unbound variables are akin to arbitrary individuals [Fine, 1983, Fine, 1985]. Non-answer literals in a clause can be viewed as additional specifications of what must hold in order for the arguments of the answer literals to constitute answers.

Imielinski's definition of "rules as answers" given previously [Imielinski, 1987] posits *predicate sharing* as a way of connecting conditions on an answer and the answer itself. We have already seen that predicate sharing does not necessarily accompany a generic answer when answering a question in a resolution theorem prover.

It seems reasonable to identify non-answer literals that should count as the antecedents in a generic answer by looking at their variables. These variables should describe properties of individuals satisfying a question; that is, they should *match* the variables in the answer literals. Identifying generic answers with literals containing variables from the original question can be viewed as an issue of relevance: the literals that are *relevant* to a generic description of the arbitrary individuals mentioned in the answer literals are those that mention the individuals. Anderson and Belnap [1975, pp.32–33] discuss relevance in a logic of entailment as depending on *variable sharing*:

> So we propose as a *necessary*, but by no means sufficient, condition for the relevance of $\mathcal{A}$ to $\mathcal{B}$ in the pure calculus of entailment, that $\mathcal{A}$ and $\mathcal{B}$ must share a variable.

The notion of variable sharing has to be clarified. The following examples show answer literals that contain variables, not all of which overlap the variables in the answer literal. The rule base consists of the following fact and rule:

1. (university Michigan)
2. ∀ x ∀ y ∀ z [((university x) ∧ (sport y) ∧ (athlete z) ∧ (playsWell z y))
   ⇒ (providesScholarshipFor x z)]

Table 4.12: Question

Logic Form	*Clause*
$\exists\, x\ (providesScholarshipFor$ $Michigan\ x)$	{(¬ (PROVIDESSCHOLARSHIPFOR MICHIGAN ?22)), (ANSWER (PROVIDESSCHOLARSHIPFOR MICHIGAN ?22))}

Using these two pieces of information, a question about scholarships may be asked: *For whom does Michigan provide scholarships?* . The logical representation and clause form for the question are shown in Table 4.12. The answer in clause form is:

{(¬(ATHLETE ?123)),(¬(PLAYSWELL ?123 ?125)),(¬(SPORT ?125)),

(ANSWER (PROVIDESSCHOLARSHIPFOR MICHIGAN ?123))}

The answer written in rule form is:

$$\forall x \forall y\, [((athlete\ x) \wedge (playsWell\ x\ y) \wedge (sport\ y))$$
$$\Rightarrow (providesScholarshipFor\ Michigan\ x))]$$

The variable x is the only variable in the answer literal: it occurs in two of the literals in the antecedent of the rule and not the other. This rule has the "feel" of a generic because it describes conditions related to the (somewhat) *arbitrary* individual satisfying the question: this individual must be an athlete who plays a sport well. While y, the variable representing any sport, does not occur in the answer literal, it occurs in a non-answer literal *with* x. The criterion for including non-answer literals must account for those containing variables that occur *with* variables from the answer literals. The non-answer literals that are included in a generic answer are in a set that we term *the closure of variable sharing of the answer literals*. In order to define this set, some preliminary results are required. We begin by defining the *variable sharing set of a literal*.

Definition 4.6 Variable Sharing Set of a Literal. *The variable-sharing set of a literal l with respect to a set of literals C, is a set of literals $\mathcal{VS}(l)$ such that each literal in $\mathcal{VS}(l)$ is an element of C and shares at least one variable with l.*

For example, given the clause $\mathcal{C} = \{R(x,z),\ P(x),\ Q(z)\}$, we have $\mathcal{VS}(P(x)) = \{R(x,z)\}$, and $\mathcal{VS}(R(x,z)) = \{P(x), Q(z)\ \}$.

The definition of the variable sharing set of a literal is made with respect to a single clause (i.e. a set of literals): because variables in different clauses are renamed apart from one another, there is no variable sharing between clauses. By this definition, the variable sharing set of a ground literal is the empty set. We compute the variable sharing set of the answer literals in a clause in order to determine the form of the answer associated with a clause. It is important that the variable sharing set of a literal with respect to a particular set of literals is unique.

Theorem 4.3 (Uniqueness of the variable sharing set of a literal) *The variable sharing set of a literal with respect to a clause $\mathcal{C}$ is unique.*

Proof: Assume that there are sets $\mathcal{VS}(l)$ and $\mathcal{VS}'(l)$ such that $\mathcal{VS}(l) \neq \mathcal{VS}'(l)$, and both $\mathcal{VS}(l)$

and $\mathcal{VS}'(l)$ are variable sharing sets of literal l with respect to clause $\mathcal{C}$. There must be some literal l_k such that either $l_k \in \mathcal{VS}(l)$ and $l_k \notin \mathcal{VS}'(l)$, or vice versa. By definition, l_k shares at least one variable in common with l. But that means that l_k is in the variable sharing set of l. Therefore, either $\mathcal{VS}(l)$ or $\mathcal{VS}'(l)$ is *not* a variable sharing set of l, which contradicts our assumption. □

Definition 4.7 Variable Sharing Set of a Set of Literals. *The variable sharing set of a finite set of literals* $\mathcal{S} = \{l_1, l_2, \ldots, l_n\}$ *is the union of the variable sharing sets for each of the literals:* $\mathcal{VS}(\mathcal{S}) = \bigcup_{l_i \in \mathcal{S}} \mathcal{VS}(l_i)$.

For example, with respect to the clause $\mathcal{C} = \{R(x,z),\ P(x),\ Q(z)\}$, the variable sharing set for the literals $R(x,z)$ and $P(x)$ is the union of their variable sharing sets, namely, $\{R(x,z)\} \cup \{P(x), Q(z)\}$, which is $\{R(x,z),\ P(x),\ Q(z)\}$: the entire clause $\mathcal{C}$. Given any finite set of literals, the variable sharing set can be computed.

The next notion to be introduced is the *closure of variable sharing* of a set of literals. Intuitively, this includes not only the literals from the variable sharing set of the set of literals, but also the literals from the variable sharing set of the literals in that variable sharing set, and so on.

Definition 4.8 Closure of Variable Sharing of a Set of Literals. *The closure of variable sharing of a set of literals* $\mathcal{L}$, *denoted* $\mathcal{CVS}(\mathcal{L})$, *is the fixed point of the* $\mathcal{VS}$ *operator:* $\mathcal{VS}(\mathcal{CVS}(\mathcal{L})) = \mathcal{CVS}(\mathcal{L})$. *It is inductively defined as follows:*

$\mathcal{CVS}_0 = \mathcal{VS}(\mathcal{L})$

$\mathcal{CVS}_1 = \mathcal{VS}(\mathcal{CVS}_0)$

$\vdots$

$\mathcal{CVS}_n = \mathcal{VS}(\mathcal{CVS}_{n-1})$

$\mathcal{CVS}_{n+1} = \mathcal{CVS}_n$

At this point in the computation, a fixed point has been reached, and no more literals can be added to the set. The maximum number of literals in this set is the number of literals in the clause $\mathcal{C}$ *with respect to which* $\mathcal{CVS}(\mathcal{L})$ *is computed.*

Note that $\mathcal{VS}(\mathcal{L})$ is not always the same as $\mathcal{CVS}(\mathcal{L})$. For example, given the following clause:

$$\mathcal{C} = \{R(x,z),\ P(x),\ Q(z,y),\ S(y)\}$$

we have $\mathcal{VS}(R(x,z)) = \{P(x), Q(z,y)\}$ and $\mathcal{CVS}(R(x,z)) = \{P(x), Q(z,y),\ S(y)\}$.

Based on the definition of the closure of variable sharing of a set of literals, we define the *closure of variable sharing of the answer literals*. This definition is used to identify clauses that are associated with generic answers.

Definition 4.9 Closure of Variable Sharing of the Answer Literals. *The closure of variable sharing of the answers literals $\mathcal{A}$ in a clause $\mathcal{C}$ is $\mathcal{CVS}(\mathcal{A})$ with respect to the clause $\mathcal{C} \setminus \mathcal{A}$.*

In other words, only non-answer literals may be included in the closure of variable sharing of the answer literals.

At this point, we can note the following characteristics of clauses associated with generic answers: the answer literals cannot all be ground, the non-answer literals must contain variables, and *all* non-answer literals must be in the closure of variable sharing of the answer literals.

What of the case where the variable sharing between answer and non-answer literals is partial? One such case shows why there is no requirement for every variable in the answer literals to appear in every non-answer literal. Consider a rule representing that if x is green and y is blue then x loves y:

$$\forall\, x\, \forall\, y\, [((green\ x)\ \wedge\ (blue\ y))\ \Rightarrow\ (loves\ x\ y)]$$

Given this rule, and the question *Is there is someone who loves someone?* , written in FOPC as $\exists\, x\, \exists\, y\, (loves\ x\ y)$, what answer is found? The clause representation of the rule and question are as follows:

1. {(¬(GREEN ?6)), (¬(BLUE ?8)), (LOVES ?6 ?8)}

2. {(¬(LOVES ?17 ?19)), (ANSWER (LOVES ?17 ?19))}

The answer clause is:

3. {(ANSWER (LOVES ?54 ?56)), (¬(GREEN ?54)), (¬(BLUE ?56))}

The rule form of the answer is $\forall\, x\, \forall\, y\, [((green\ x)\ \wedge\ (blue\ y))\ \Rightarrow\ (loves\ x\ y)]$. While two variables appear in the answer literal, ?54 and ?56, the non-answer literals $(\neg(green$?54$))$ and $(\neg(blue$?56$))$ each contain only one of the variables. As long as a single variable in a non-answer literal is shared with the answer literals that is sufficient to connect them, which accords with our definition of the closure of variable sharing of the answer literals.

Another case of partial overlap of the variables in the answer and non-answer literals is found when there are variables in the answer literals that do not occur in the non-answer literals. According to the definition of the variable sharing of the answer literals, not every variable in the answer literals is required to appear in the non-answer literals. Given the rule *everyone who loves Bob loves everyone*, and the question *Is there someone who loves everyone?* , the answer is *everyone who loves Bob loves everyone*. The clause representation of the rule, question, and answer are as follows:

1. {(¬(LOVES ?6 BOB)), (LOVES ?6 ?8)}
2. {(¬(LOVES ?17 ?19)), (ANSWER (LOVES ?17 ?19))}
3. {(ANSWER (LOVES ?56 ?58)),(¬(LOVES ?56 BOB))}

In this case, the variables ?56 and ?58 appear in the answer literal, but only ?56 appears in the non-answer literal.

Finally, there may be variables in the non-answer literals that are not found in the answer literals. A question asked about a single object with a particular property may evoke an answer that involves a long chain of variable sharing, where there is little overlap between the variable in the question and those in the antecedent of the rule that comprises the answer. The following example illustrates not only how this may come about, but also the way in which complex object descriptions may constitute generic answers. Given the question *What is valuable?* , a possible answer is *Items in a locked cabinet where the key is held by a senior manager are valuable*. A logic representation of this question and answer is as follows:

Question: $\exists\, x\ (valuable\ x)$

Answer: $\forall\ (x\ y\ z\ k\ l)$

$$\begin{aligned}&[((item\ x)\ \wedge\ (cabinet\ y)\ \wedge\ (senior\text{-}manager\ z)\\&\wedge\ (key\ k)\ \wedge\ (lock\ l)\ \wedge\ (locks\ l\ y)\\&\wedge\ (key-to\ k\ l)\ \wedge\ (held\text{-}by\ k\ z)\ \wedge\ (in\ x\ y))\\&\qquad \Longrightarrow\ (valuable\ x)]\end{aligned}$$

The closure of variable sharing of the answer literals for this example, where there is a single answer literal, $ANSWER(valuable\ x)$, is constructed as follows:

$$\begin{aligned}\mathcal{CVS}_0 &= \mathcal{VS}(\{ANSWER(valuable\ x)\})\\&= \{(item\ x),\ (in\ x\ y)\}\end{aligned}$$

$$
\begin{aligned}
\mathcal{CVS}_1 &= \mathcal{VS}(\mathcal{CVS}_0) \\
&= \{(item\ x),\ (in\ x\ y),\ (locks\ l\ y),\ (cabinet\ y)\} \\
\mathcal{CVS}_2 &= \mathcal{VS}(\mathcal{CVS}_1) \\
&= \{(item\ x),\ (in\ x\ y),\ (locks\ l\ y),\ (cabinet\ y),\ (lock\ l),\ (key-to\ k\ l)\} \\
\mathcal{CVS}_3 &= \mathcal{VS}(\mathcal{CVS}_2) \\
&= \{(item\ x),\ (in\ x\ y),\ (locks\ l\ y),\ (cabinet\ y),\ (lock\ l),\ (key-to\ k\ l)\ ,(key\ k), \\
&\quad\ (held-by\ k\ z)\} \\
\mathcal{CVS}_4 &= \mathcal{VS}(\mathcal{CVS}_3) \\
&= \{(item\ x),\ (in\ x\ y),\ (locks\ l\ y),\ (cabinet\ y),\ (lock\ l),\ (key-to\ k\ l)\ ,(key\ k), \\
&\quad\ (held-by\ k\ z),\ (senior-manager\ z)\}) \\
\mathcal{CVS}_5 &= \mathcal{VS}(\mathcal{CVS}_4) \\
&= \{(item\ x),\ (in\ x\ y),\ (locks\ l\ y),\ (cabinet\ y),\ (lock\ l),\ (key-to\ k\ l)\ ,(key\ k), \\
&\quad\ (held-by\ k\ z),\ (senior-manager\ z)\}) \\
&= \mathcal{CVS}_4
\end{aligned}
$$

The closure has been computed at this point because the last iteration has added no new literals to the set. $\mathcal{CVS}_4$ is the fixed point. The closure of variable sharing of the answer literal includes all of the non-answer literals; thus, this represents a generic answer. The answer can be written as follows:

$$
\begin{aligned}
&\forall\ (x\ y\ z\ k\ l) \\
&\qquad [((item\ x)\ \wedge\ (cabinet\ y)\ \wedge\ (senior-manager\ z) \\
&\qquad \wedge\ (key\ k)\ \wedge\ (lock\ l)\ \wedge\ (locks\ l\ y) \\
&\qquad \wedge\ (key-to\ k\ l)\ \wedge\ (held-by\ k\ z)\ \wedge\ (in\ x\ y)) \\
&\qquad\qquad \Longrightarrow\ (valuable\ x)]
\end{aligned}
$$

The literals in the antecedent give a set of sufficient conditions for an object to be considered valuable.

We are now in a position to define what it means for a clause to be considered a generic answer clause.

Definition 4.10 Generic Answer Clause. *A generic answer clause is a clause with the following properties:*

1. *There is at least one answer literal that contains a variable, a Skolem constant, or a Skolem*

function.

2. *All non-answer literals must be in the closure of variable sharing of the answer literals.*

We will say that generic answer clauses are *associated* with generic answers. Note that this definition encompasses unconstrained generic answers as well as the various answers that include partial overlap between the variables in the answer and non-answer literals. A question that arises here is whether sharing of Skolem constants or functions between answer and non-answer literals should be considered as a type of variable sharing. The answer is no, and a full discussion of this issue is presented in the section of this chapter that describes hypothetical answers.

Definition 4.11 Generic Answer. *A generic answer is a universally quantified formula associated with a generic answer clause. The (possibly empty) antecedent is the conjunction of the negations of all of the non-answer literals in the clause. The consequent is the disjunction of all of the arguments of the answer literals in the clause. Skolem constants and functions are replaced by variables. Variables local to the consequent are universally quantified with narrow scope. All other variables are universally quantified with wide scope over the entire formula.*

Note that the variables local to the antecedent could be narrowly scoped with an existential quantifier: this would simply be a logically equivalent variant form. The form using all universal quantifiers was selected as best capturing the idea of a *generic*.

4.4.5 Discussion of Generic Answers

Generic answers *describe* individuals satisfying a question, capturing generalities and referring to categories or kinds: as such, they are often desirable. In a system that provides generic answers, the need to introduce individuals just so that an answer can be found (e.g., a particular fox, a particular wolf, etc., in formulations of Schubert's Steamroller [Stickel, 1985]) is obviated. There are many circumstances under which a generic answer is preferable to a specific answer. Even when there is not a preference, a generic answer may be given in the absence of a specific answer, conveying information that would otherwise be lost. When backward chaining is employed in the search for specific answers, generic answers may be discovered "along the way" to these answers, meaning that they require less time or resources to generate. Generic answers may be more succinct than lists of individuals. They are more appropriate than specific answers when a question is universally quantified [Webber, 1986].

In addition to computational efficiency gains provided by generic answers, they are important in terms of human cognition: generic answers are ubiquitous in human conversation. Given the question, "What barks?", the generic answer "Dogs bark." is a more appropriate answer than a litany of individual dog names. In fact, an exhaustive list of dogs is not an answer most people would even consider giving. It isn't difficult to come up with many questions for which the most natural answers correspond to categories rather than lists of individuals. Work by Rosch showed that people associate large amounts of information with basic-level categories [Rosch and Mervis, 1975]. Specific answers do not provide information about categories, while generic answers are capable of doing so. Categories form the basis for description logics: a KR formalism capable of providing generic descriptions in response to questions. of knowledge and are capable of handling generic descriptions. The ability to provide generic answers at a variety of taxonomic levels is a strength of description logics from the standpoint of question answering. Note that there are different possible representations of generic information about categories of individuals in different KR&R systems. Rather than representing a category restriction as the antecedent of a rule, it might be attached as a constraint to a slot that is filled by a satisfying individual. It might also be implicit if variables are typed.

In the context of databases, Cholvy and Demolombe [1986] point out the persistence of intensional answers in contrast to extensional answers. In most databases, the rules specifying relationships among data and constraints on data are relatively static, while the facts (corresponding to extensional answers) are frequently updated. This is clearly another advantage of generic answering, particularly when a set of facts is large and subject to frequent updates.

Thus, generic answers are an important, under-appreciated class of answers that are (for all practical purposes) *already being generated* by resolution refutation theorem provers. In many cases, these answers are simply ignored. There may be circumstances under which they should be acknowledged, even when the goal of reasoning is to find a proof of a theorem. In combination with specific answers they comprise the set individuals and their descriptions that can be used to satisfy universally and existentially quantified questions.

We have clearly distinguished generic answers from the set of all rules by defining the *closure of* *variable sharing of the answer literals*. In the past, overlap of variables and/or predicates has been suggested as ways in which generic answers can be identified. However, we have just demonstrated that overlap is not sufficient in defining generic answers.

Generic and specific are not the only possible answer classes. Section 3.4.3 demonstrated two clauses that are neither generic nor specific according to our definitions. The next section describes how these, along with all other clauses relevant to a question but not accounted for by the classes specific and generic, fall into the class of *hypothetical answers*. It is clear that there may be disjunctive generic answers. Rather than including a separate discussion of disjunctive answers in the context of the different primary answer classes, the topic is treated in Section 4.8 of the dissertation.

4.5 Hypothetical Answers

A specific answer is a witness that proves the truth of an existential hypothesis: it is a ground answer to a question. A generic answer is a rule that expresses general properties of classes or groups of objects that answer a question. A hypothetical answer is a rule that is *not* a generic answer. Given the question *What barks?* , *all dogs bark* is a generic answer and *if rover barks then all dogs bark* is a hypothetical answer.

A hypothetical answer comprises two parts: one part is either a specific or generic answer, and the other is a condition that must hold in order for the accompanying answer to *be considered an answer*. In the above example, the condition is *rover barks* and the accompanying generic answer is *all dogs bark* . The condition is termed the *hypothetical component* of the answer, and the rest of the answer is termed the *answer component*. Hypothetical answers are divided into specific and generic hypothetical answers, based on the type of the answer component. The hallmark of the hypothetical answer is the way in which information belonging to the hypothetical component is identified: literals *not* in the closure of variable sharing of the answer literals comprise the hypothetical component of the hypothetical answer.

Demolombe provides a definition of a *conditional answer* [Demolombe, 1992] in terms of logic databases. Given the following:

- $\mathcal{EDB}$ is a set of ground formulas
- $\mathcal{IDB}$ is a set of formulas

- F(x) is a query
- Q = $(\forall\, x(F(x) \Rightarrow q(x)))$
- T = $\mathcal{IDB} \cup \mathcal{EDB} \cup Q$ is a theory
- all formulas are represented in clause form and clauses do not contain function symbols.

this is a definition of a conditional answer, where c is a clause in T.

> Let q be a positive literal; the conditional answer to the query q is the set of clauses:
> $\{q\sigma \vee c \mid T \vdash q\sigma \vee c$, and $q\sigma \vee c$ is minimal wrt subsumption$\}$

According to this definition, *any* answer in rule form is a conditional answer, whether generic or hypothetical. Demolombe partitions conditional answers into two classes: Intensional and Extensional. Intensional Conditional Answers are derived from the IDB, and Extensional Conditional Answers are derived from the EDB. This partition is related to the division of intensional and extensional information in the database model to which he refers. Because they contain *no* variables, Demolombe's Extensional Conditional Answers are a subset of our hypothetical answers. Demolombe's Intensional Conditional Answers include generic answers as well as non-ground hypothetical answers.

In a later paper, Demolombe [1997, pp.104–105] restricts his definition of conditional answers to those that are all ground, namely, those he terms Extensional Conditional Answers in his previous paper. As mentioned above, these answers represent the subset of our hypothetical answer class that is characterized by its lack of variables. Note that this new definition *excludes* generic answers:

> The conditional answer to a query $q(x)$ is a set of ground formulas of the form
> $$cond(a, b) \rightarrow q(a)$$
> where $cond(a, b)$ represented the *additional* information necessary to derive the answer $q(a)$. A general definition of conditional answers is
> $$\{cond(a, b) \rightarrow q(a) \mid DB \vdash cond(a, b) \rightarrow q(a)\}.$$

The form of this answer is similar to our notion of hypothetical answers. However, Demolombe's definition restricts conditional answers so as to exclude terms containing variables. Note that $cond(a, b)$ maps into what we term the *hypothetical component* of the answer, and $q(a)$ is a *part* of what we term the *answer component*: it is the answer component when the answer is *specific*, that is, all ground. This definition makes clear our extension of Demolombe's work on conditional answers.

To illustrate hypothetical answers, we use an example from the previous sections. First, the KB to be considered is that presented in Table 4.2 (from [Rich and Knight, 1991, p.192]). The question asked was given in the previous section on generic answers, namely, *What does Boots like to eat?* . Note once again that the individual *Boots* is not mentioned in the KB: nothing whatsoever is known about Boots. Under ordinary circumstances, a query about an unknown individual would fail. However, when all relevant clauses are considered as answers, Table 4.13 shows the results.

Table 4.13: Hypothetical Answers

Clause	*Rule*
1. {(ANSWER (LIKESTOEAT BOOTS HERB), (¬ (CAT BOOTS)) }	$(CAT\ BOOTS) \Rightarrow (LIKESTOEAT\ BOOTS\ HERB)$
2. {(ANSWER (LIKESTOEAT BOOTS CHARLIE), (¬ (CAT BOOTS)) }	$(CAT\ BOOTS) \Rightarrow (LIKESTOEAT\ BOOTS\ CHARLIE)$
3. {(ANSWER (LIKESTOEAT BOOTS HERB), (¬ (CALICO BOOTS)) }	$(CALICO\ BOOTS) \Rightarrow (LIKESTOEAT\ BOOTS\ HERB)$
4. {(ANSWER (LIKESTOEAT BOOTS CHARLIE), (¬ (CALICO BOOTS)) }	$(CALICO\ BOOTS) \Rightarrow (LIKESTOEAT\ BOOTS\ CHARLIE)$
5. {(ANSWER (LIKESTOEAT BOOTS ?285)), (¬ (TUNA ?285)),(¬ (CALICO BOOTS)) }	$(CALICO\ BOOTS) \Rightarrow (\forall x\ [(TUNA\ x) \Rightarrow (LIKESTOEAT\ BOOTS\ x)])$
6. {(ANSWER (LIKESTOEAT BOOTS ?270)), (¬ (FISH ?270)),(¬ (CALICO BOOTS)) }	$(CALICO\ BOOTS) \Rightarrow (\forall x\ [(FISH\ x) \Rightarrow (LIKESTOEAT\ BOOTS\ x)])$
7. {(ANSWER (LIKESTOEAT BOOTS ?150)), (¬ (TUNA ?150)),(¬ (CAT BOOTS)) }	$(CAT\ BOOTS) \Rightarrow (\forall x\ [(TUNA\ x) \Rightarrow (LIKESTOEAT\ BOOTS\ x)])$
8. {(ANSWER (LIKESTOEAT BOOTS ?523)), (¬ (FISH ?523)),(¬ (CAT BOOTS)) }	$(CAT\ BOOTS) \Rightarrow (\forall x\ [(FISH\ x) \Rightarrow (LIKESTOEAT\ BOOTS\ x)])$

English glosses for the rules in Table 4.13 are given below:

1. If Boots is a cat then Boots likes to eat Herb.
2. If Boots is a cat then Boots likes to eat Charlie.
3. If Boots is a calico then Boots likes to eat Herb.
4. If Boots is a calico then Boots likes to eat Charlie.
5. If Boots is a calico then Boots likes to eat tuna.
6. If Boots is a calico then Boots likes to eat fish.

7. If Boots is a cat then Boots likes to eat tuna.
8. If Boots is a cat then Boots likes to eat fish.

Answers 1–4 are termed *specific hypothetical answers* because the associated answer component is specific. Similarly answers 5–8 are termed *generic hypothetical answers* because the associated answer component is generic.

The rules in Table 4.13 show the separation of non-answer literals into two sets. When present, non-answer literals in the closure of variable sharing of the answer literals comprise the antecedent to a generic answer; thus, they appear in the *answer component* of the rule. Any other non-answer literals are part of the hypothetical component of the rule. There are only two distinct hypothetical components across the eight answers: *Boots is a cat* and *Boots is a calico*. Because all calicos are cats, these hypotheses are related taxonomically: if Boots is a calico, then, based on the information in the KB, Boots is a cat. As mentioned in the discussion of generic answers, this taxonomic relation may not exist in the mind of the questioner: a questioner might know that Boots is a calico and know nothing about the relationship between calicos and cats. In this case (which is just one example of this phenomenon), it would clearly be detrimental to deny "answer" status to an answer with hypothetical component *Boots is a calico* simply because it is taxonomically subsumed by *Boots is a cat*.

Not all hypothetical components of answers are ground. The following example shows a universally quantified hypothetical component with an unconstrained generic answer as a consequent. Given a knowledge base with one rule:

1. $\forall$ x ((dog x) $\Rightarrow$ (barks x))

and the question *Does everything bark?* , the answer generated by our theorem prover is, in clause form, {(¬(DOG S11)), (ANSWER (BARKS S11))}. The corresponding answer in rule form is (DOG S11) $\Rightarrow$ (BARKS S11). As previously discussed, Skolem constants and functions can be replaced by universally quantified variables in clauses *outside* the context of an ongoing proof. When should this replacement take place: should it be *before* or *after* the answer literals are divided into the hypothetical, and generic or specific answer component of a hypothetical answer? If it takes place beforehand, it seems to indicate that S11 in both literals should be replaced with the same variable, thus yielding a generic answer. This would conflict with our definition of the closure of variable sharing, which stipulates that the literal (DOG S11) — an all-ground literal, which does not share variables with the answer literal (ANSWER (BARKS S11)) — should appear in the hypothetical position of the answer. A consideration of the original question, and the semantics of both the question and answer, leads to clarification on this issue. The original question asked whether *everything* barks. If S11 is replaced by the same variable in both clauses, thus becoming a shared, universally quantified variable in a generic answer, the answer would be $\forall\, x\, [(dog\ x)\ \Rightarrow\ (barks\ x)]$. The meaning of this rule is that if something is a dog then it barks; it makes no claim about whether or not the arbitrary individual barks. On the other hand, by first separating the literals into the two components of the answer, and replacing S11 by a universally quantified variable with *local scope* in the two answer components, the answer would be $\forall\, x\ (dog\ x)\ \Rightarrow\ \forall\, y\ (barks\ y)$. The meaning of this rule is that if everything is a dog then everything barks. This is clearly the right answer. The only way for an answer literal to contain a Skolem function is if a question contains a universally quantified variable. Typically, the *wh-questions* (*who, what, where, when, why*) are associated with existentially quantified variables representing individuals. In the case of *who*, an agent is sought; for *what* questions, an event or entity; *where* questions seek a location; *when* questions ask for a time; *why* questions are associated with reasons.

Thus, not only must translation from Skolem functions and constants to variables take place outside the scope of reasoning, but it must be delayed until the literals in a clause have been properly separated into the hypothetical and answer components.

4.5.1 Quantification in the Hypothetical Component of the Answer

At this point, an algorithm for translating from clause form to FOPC representation of an answer is nearly in hand. There is one other consideration concerning quantification in the hypothetical component of the answer: should the variables in the hypothetical component be existentially or universally quantified? Another example will help elucidate both the problem and the solution.

Table 4.14 shows a knowledge base with information that leads to more interesting hypothetical answers.

The question to be asked is *What does Muffy like?* . The following is a list of answers in rule form along with English glosses:

1. $(LIKES\ MUFFY\ COWCREAMER)$

Table 4.14: Knowledge Base for Hypothetical Answer Example

Logic Form	*English Gloss*	*Cl. Num*	*Clause*
$\forall\, x\, [(American\ x) \Rightarrow (consumer\ x)]$	Americans are consumers	1.	{(¬ (AMERICAN ?11)), (CONSUMER ?11)}
$(\forall\, x\, (\forall\, y\, [((consumer\ x) \wedge (bargain\ y)) \Rightarrow (likes\ x\ y)]))$	consumers like bargains	2.	{(¬ (CONSUMER ?5)), (¬ (BARGAIN ?7)), (LIKES ?5 ?7)}
$(\forall\, x\, (\forall\, y\, [((consumer\ x) \wedge (junk\ y)) \Rightarrow ((\neg\, (likes\ x\ y))]))$	consumers don't like junk	3.	{(¬ (CONSUMER ?12)), (¬ (JUNK ?14)), (¬((LIKES ?12 ?14))}
$(American\ Muffy)$	Muffy is an American	4.	{(AMERICAN MUFFY)}
$(bargain\ cowCreamer)$	The cow creamer is a bargain	5.	{(BARGAIN COWCREAMER)}
$(bargain\ cowPitcher) \vee (junk\ cowPitcher)$	The cow pitcher is a bargain or the cow pitcher is junk	5.	{(BARGAIN COWPITCHER), (JUNK COWPITCHER)}

Muffy likes the cow creamer.

2. $(\forall\, x\, ((BARGAIN\ x) \Rightarrow (LIKES\ MUFFY\ x)))$

Muffy likes bargains.

3. $(\neg\, (JUNK\ COWPITCHER)) \Rightarrow (LIKES\ MUFFY\ COWPITCHER)$

If the cow pitcher isn't junk then Muffy likes it.

Answer 1 is a specific answer, 2 is a generic answer, and 3 is a (ground) specific hypothetical answer. Two other answers are found, shown initially in clause form:

4. {(¬(CONSUMER ?22)), (¬(LIKES ?22 COWPITCHER)),
(LIKES MUFFY COWPITCHER)}

5. {(¬(AMERICAN ?24)), (¬(LIKES ?24 COWPITCHER)),
(LIKES MUFFY COWPITCHER)}

These clauses correspond to hypothetical answers because of the lack of variables in the answer literals: when the closure of variable sharing of the answer literals is empty, there is no generic answer.

Variables in clauses are implicitly universally quantified. Recall that, during conversion to clause form, all quantifiers are moved out of the matrix, giving them scope over the entire formula that will ultimately be rewritten as a clause or set of clauses. The last step in conversion to clause form is to simply drop the universal quantifiers. Reversing the process leads to universal quantification of variables in the clause. By *definition*, the variables in the hypothetical component of the answer occur *only* in that component: they are not shared with the generic or specific answer component. This leads to the following possible representation of answer 4, above:

4a. $\forall\, x\, [((consumer\ x)\ \wedge\ (likes\ x\ cowPitcher))\ \Rightarrow\ (likes\ Muffy\ cowPitcher)]$

An alternate, logically equivalent representation is:

4b. $[\exists\, x\, ((consumer\ x)\ \wedge\ (likes\ x\ cowPitcher))]\ \Rightarrow\ (likes\ Muffy\ cowPitcher)$

The hypothetical component *always* appears as the antecedent of a rule: any variables in the hypothetical component appear *only* in that component and nowhere else in the formula. In this context, the wide-scope universal quantifier is equivalent to a narrow-scope existential quantifier.

We opt for narrow-scope existential quantification in our proposed representation, because it reflects the *semantics* of the hypothetical component: the hypothetical can be viewed *in and of itself* as an unanswered question, and as such it should be easily *detachable* from the rest of the answer in case the answering agent wishes to subsequently pose it.

First, we define what it means for a clause to be considered a hypothetical answer clause.

Definition 4.12 Hypothetical Answer Clause. *A hypothetical answer clause is a clause with the following properties:*

1. *There is at least one answer literal.*
2. *There is at least one non-answer literal that is not in the closure of variable sharing of the answer literals.*

We will say that hypothetical answer clauses are *associated* with hypothetical answers. An example of a hypothetical answer clause (from Table 4.13) is:

{(ANSWER (LIKESTOEAT BOOTS HERB), (¬ (CAT BOOTS))}.

Definition 4.13 Hypothetical Answer. *A hypothetical answer is a rule associated with a hypothetical answer clause. The antecedent of the rule is the conjunction of the negations of the non-answer literals in the clause that are not in the closure of variable sharing of the answer literals. Any variables in the antecedent are existentially quantified with narrow scope over the antecedent. Any Skolem functions in the antecedent are replaced by universally quantified variables with narrow scope over the antecedent. The remaining literals in the clause correspond either to a specific or a generic answer that comprises the consequent of the rule.*

The hypothetical answer associated with the above clause (from Table 4.13) is:

$(CAT\ BOOTS) \Rightarrow (LIKESTOEAT\ BOOTS\ HERB)$.

The antecedent of this rule is $(CAT\ BOOTS)$, which is the hypothetical information. The consequent, representing a specific answer, is $(LIKESTOEAT\ BOOTS\ HERB)$. The antecedent describes conditions that must hold in order for the associated answer to be true, whether that answer is generic or specific. This example conveys the information that *if Boots is a cat then Boots likes to eat Herb*. In contrast to the antecedent of a generic answer, which describes *conditions on the individuals* that satisfy the question, the antecedent of a hypothetical answer describes *conditions on the answer itself*.

4.5.2 Hypothetical Answers and Assumption-Based Truth Maintenance Systems

The hypothetical component of the answer is similar to what Reiter and de Kleer call a *support* for a clause [Reiter and de Kleer, 1987, page 184]:

A clause $\mathcal{S}$ is a *support* for $\mathcal{C}$ with respect to Σ iff $\Sigma \not\models \mathcal{S}$ and $\Sigma \models \mathcal{S} \supset \mathcal{C}$

"... the conjunction of literals $\mathcal{S}$ is an hypothesis which, if known to Σ (and hence to the Reasoner) would sanction the conclusion $\mathcal{C}$." Note that this statement makes the assumption that $\mathcal{S}$ and $\mathcal{C}$ are independently closed formulae.

One of the characteristics of question answering is the fact that some hypothetical answers discovered in the process of reasoning are later found *not* to be hypothetical answers. Some of the answers we term hypothetical may at some point further along in the reasoning process be subsumed: this would occur if the hypothetical component is discovered to either be in or to follow from the KB. It is concerning this point that the analogy between hypothetical answers and assumption-based truth maintenance systems fails to hold. The potential ephemeral nature of hypothetical answers is noted by Demolombe [1992], who uses the term *Extended Conditional Answers* to refer to *all* hypothetical answers, including those whose hypothetical component may at some point in the future be subsumed. Demolombe reserves the term *Conditional Answer* for those hypotheticals that are never subsumed.

4.5.3 Hypothetical Answers and Abduction

Hypothetical answering and abduction are very similar, as noted in the previously presented literature survey. As we shall demonstrate, it is the goal of reasoning that differentiates between them.

Some differences can be immediately noted between abduction and the production of hypothetical answers to questions. The observations or facts comprising the input to the abductive reasoning process are assumed by the reasoner to be *true*, whereas the status of the propositional content of a question posed to a question answerer with regard to truth or falsity is, initially, *unknown*. An abductive reasoner is able to learn something new by assuming the truth of an abductive hypothesis: it represents an explanation for what is known. A question answering agent would have no reason to make assumptions based on a question whose propositional status remains unknown after reasoning.

Note that this difference reflects the difference discussed by Magnani [1998]. The generation and assumption of abductive hypotheses corresponds to his second definition of abduction: "inference to the best explanation." Hypothetical answering, on the other hand, corresponds to his first definition of abduction: "abduction that only generates plausible hypotheses". One of his terms for this is *selective abduction*.

The existence of multiple explanations in abduction is a problem: often these explanations are *competing*, and there is a notion of a *best* explanation. However, the existence of multiple hypothetical answers is *not* a problem. All such answers are implied by the knowledge base, and their status is not provisional. Unless the answering agent has an extensive model of the questioner, there is no way for the answering agent to infer which answer is *best*. Table 4.15 shows a comparison between abductive reasoning and the generation of hypothetical answers to a question.

As shown in Table 4.15, abduction begins with observable facts and reasons towards provisional hypotheses. The observable facts are known to be true and are added to the knowledge base of the

Table 4.15: A Comparison between the Generation of Abductive Hypotheses and the Generation of Hypothetical Answers

	Abduction	*Hypothetical Q/A*
Start State of Reasoning Agent KB	$\mathcal{KB}$	$\mathcal{KB}$
Information "Trigger" for Reasoner Example:	Fact/Observation The sky is blue	Question Is the sky blue?
Representation of Information	$\mathcal{P}$	$\mathcal{P}$ (propositional content of question)
KB of Reasoning Agent	$\mathcal{KB}' = \mathcal{KB} \cup \mathcal{P}$	$\mathcal{KB}$
Reasoning Proceeds With:	$\mathcal{KB}'$ (or $\mathcal{KB} \cup \neg\mathcal{P}$ depending on how reasoner operates)	$\mathcal{KB} \cup \neg\mathcal{P}$ (refutation proof)
Purpose of Reasoning	Find an explanation for $\mathcal{P}$	Determined the validity of $\mathcal{P}$
Conclusion of Reasoning Example:	$\mathcal{KB}' \vdash \mathcal{H} \supset \mathcal{P}$ If it is not raining then the sky is blue	$\mathcal{KB} \vdash \mathcal{H} \supset \mathcal{P}$ If it is not raining then the sky is blue
Interpretation of Results	$\mathcal{H}$ is an abductive hypothesis (assume that $\mathcal{H}$ is in the set of abducibles)	$\mathcal{H} \supset \mathcal{P}$ is a hypothetical answer $\mathcal{H}$ is a new question that is unanswerable by this reasoner
Action following Reasoning	Adopt $\mathcal{H}$ as a defeasible hypothesis	Return $\mathcal{H} \supset \mathcal{P}$ as a hypothetical answer
KB of Reasoning Agent Action:	$\mathcal{KB}'' = \mathcal{KB}' \cup \mathcal{H}$ Assume: It is not raining	$\mathcal{KB}$ New answer: If it is not raining then the sky is blue

abductive reasoner. The reasoner then searches for possible explanations for the observations. In contrast, question answering begins with a proposition that is neither known to be true nor known to be false to a reasoner. The goal of reasoning is to determine whether the proposition is true. In the absence of the ability to do that, the reasoner may produce *hypothetical answers*.

Abduction involves accepting hypotheses as true that are neither provably true nor provably false. The hypothetical component of an answer to a question is *not* assumed to be true by an answering agent. Rather, the hypothetical answer is added to a set of answers from which an answering agent may draw in order to generate responses. The answering agent may *choose* to offer the hypothetical component of an answer, which *formally* corresponds to an abductive hypothesis, back to the questioner as a follow-up question, but clearly the purpose and effect is different.

Bromberger [1992] defines abductive reasoning as the search for explanations. Not all why-questions are requests for explanations, but, when they are, Bromberger identifies an "inner question" that is presumed true. For example, the why-question *Why is the sky blue?* has as an inner question *Is the sky blue?* . Inner questions in this context function as presuppositions. The analogy between Bromberger's observations and our discussion of abduction and question answering is clear. In response to the (observation/fact) *the sky is blue*, an abductive reasoner effectively asks itself the question *Why is the sky blue?* . In response to a question whose propositional content is *the sky is blue*, a question answerer attempts to determine the validity of the proposition. An abductive reasoner is designed to make *assumptions* that are not entailed by its knowledge, while a question answerer is expected to respond with information that is true according to its knowledge.

Abductive reasoning provides a way to handle the failure to prove a proposition $\mathcal{P}$. Each failed branch of the proof tree in a resolution theorem prover corresponds to a non-empty set of clauses. The clauses on such a branch do entail $\mathcal{P}$; the problem is that such clauses are neither known to be true nor are they known to be false. Above it was mentioned that this is precisely the status of the propositional content of a question. A questioner will take a proposition whose truth is unknown and *pose* it as a question to another agent, whereas an abductive reasoner will simply make an assumption about its truth. In the context of question answering, an answering agent may in turn derive a hypothetical answer with its associated "question" (hypothetical component), namely, a proposition whose truth is unknown. In this context, the hypothetical component of an answer makes an ideal follow-up question that the answering agent may (if circumstances permit) pose to the questioner. Alternatively, in response to a hypothetical answer, a *questioner* can try to get yet another answering agent to verify the hypothetical component of the answer. Like abduction, question answering provides a way to handle the failure to prove a proposition $\mathcal{P}$: know that you don't know by the presence of a hypothetical answer, or ask another agent.

The desire for obtaining a "minimal", non-redundant, basic explanation for an observation determines what a system will consider to be an abducible hypothesis. The question answering task involves no such requirement: in fact, it is important in question answering to be able to provide answers at *many levels of specificity*. This is because the questioner's knowledge may not include the same concepts as those of the answering agent, particularly in terms of specificity/generality of concepts. Thus, no *a priori* assumptions about what will comprise a satisfactory answer to questioner may be made in the absence of a large amount of knowledge about the questioner's knowledge, which is generally impractical. For question answering, not only the clauses at the *ends* of the failed branches of a proof tree are interesting, but all intermediate clauses along the branches comprise answers, excepting the root, which is simply the negation of the original question.

Abduction is held in sharp contrast to induction [Peirce, 1932], and abductive reasoning proceeds in order to explain particular observations [Dimopoulos and Kakas, 1996], not to generalize over situations. For this reason, most work on abduction involves reasoning from observed facts to (all ground) abductive hypotheses. There is, however, some work on expanding the notion of abduction in order to support reasoning from "general observations" to general hypotheses [Michalski, 1993]. For example, the general observation *all the students here are wearing pink* corresponds to a universally quantified description. It might provide the basis for reasoning toward a general rule rather than a typical abductive hypothesis. In such cases, it is a rule rather than a fact that is abductively inferred, and the claim [Michalski, 1993] is that this combines inductive as well as abductive reasoning.

In our question answering framework, it *is* possible to express generalizations in the hypothetical component of an answer: Skolem constants are taken to be universally quantified in the hypothetical component, and there is nothing to prevent a universally quantified rule from appearing here. Note also that existentially quantified variables in the hypothetical component of an answer may *lead* to generalizations if the hypothetical component is verified. For example, the following was shown in the previous section as an example of a hypothetical answer:

$$[\exists\, x\ ((consumer\ x)\ \wedge\ (likes\ x\ cowPitcher))]\ \Rightarrow\ (likes\ Muffy\ cowPitcher).$$

If the hypothetical component, namely, $[\exists\, x\ ((consumer\ x)\ \wedge\ (likes\ x\ cowPitcher))]$ proves to be the case, then it is also true (based on the example knowledge base shown above) that

$$[\forall\, x\ ((consumer\ x)\ \Rightarrow\ (likes\ x\ cowPitcher))]$$

because, if any consumer likes the cow pitcher, it follows that the cow pitcher is a bargain: hence, it is liked by all consumers. This generalization follows from the discovery of a single instance that satisfies the hypothetical component of the answer. In short, because we have not limited the hypothetical component of an answer to ground literals, the hypothetical component may represent a hypothesis that either corresponds to a generalization or may *lead* to a generalization. Thus, hypothetical answers represent a broader class of information than mainstream abductive hypotheses.

4.5.4 Limiting the Number of Hypothetical Answers

As previously mentioned, when a question is posed to a large knowledge base the number of hypothetical answers generated may be intractible. A simple characterization of the interesting hypothetical answers is the following: *Interesting hypothetical answers are those for which either the questioner or the answering agent can "settle" the hypothesis*. It may not be practical to search for a set of literals that may be resolved against all of the hypothetical literals in an answer clause, in which case an agent (whether questioner or answerer) is incapable of determining whether a particular hypothetical answer is interesting by this criterion alone.

The failure of an answering agent to find a literal opposite to one of the hypothetical literals in an answer clause is an indication that the clause will never be subsumed by a specific or generic answer. These unresolvable literals might be cached and used to help prune the search space. The "forever hypothetical" clauses associated with these literals may only prove interesting in the absence of other types of answers. If a resolvent containing a hypothetical literal that matches an "unresolvable" is generated, further search can be halted by discarding the clause.

A variation of unit preference can be employed so that the fewer hypothetical literals a clause contains the better it is, as has been suggested in Chalupsky and Russ [2002].

If the answering agent and questioner are capable of switching roles, it may be useful for the answering agent to pose the hypothetical component of a clause as a question to the questioner. If the questioner succeeds in verifying one of the hypotheses, this information can be added to the user model of the answering agent and incorporated into reasoning. Whether the information should become part of the knowledge base of the answering agent is a question involving many other considerations, including the believability of the questioner. The assumption here is that the information does not contradict anything in the knowledge base of the answering agent, which in practice can not be guaranteed. Involving the questioner in the process of supporting or contradicting hypothetical literals means that the set of interesting hypothetical answers is determined *relative to a questioner*.

4.5.5 Circumstances under which Hypothetical Answers Arise

In a back-chaining reasoner, hypothetical answers are discovered "along the way" as the search for specific and generic answers proceeds. Most hypothetical answers function merely as resolution intermediates and are ultimately subsumed. When the hypothetical component of a hypothetical answer is subsumed, there are two possible outcomes. One possibility is that the subsuming clause is a hypothetical answer clause. The other possibility is that the subsuming clause is either a generic or a specific answer clause. In the latter case, we may say that the hypothesis presented by the hypothetical answer clause has been settled, whereas in the former case it has simply been replaced by a new hypothesis.

In general, when a hypothetical answer is not subsumed, it stands on its own as both relevant and informative. It is relevant, because it is connected to the question by a chain of reasoning. It is informative in that it provides the questioner with new information, namely, a question for which the reasoner has no answer. This question comprises the hypothetical part of the answer. When a hypothetical answer clause is converted from clause form into standard FOPC, the hypothetical part is an independent, closed formula H. The rest of the clause, namely, the answer part that is associated with either a generic or a specific answer, is itself an independent, closed formula A. We express the relationship between the hypothetical and answer parts of the clause as $\mathcal{H} \Rightarrow \mathcal{A}$. $\mathcal{H}$ is a new question that has arisen and must be addressed by the reasoner, if possible.

How does this differ, though, from a generic answer, which apparently has the same general form as a hypothetical answer: $\mathcal{G} \Rightarrow \mathcal{Q}$? If a generic answer is *all dogs bark*, isn't the antecedent also associated with a potentially unanswerable question for a reasoner, namely, *is everything a dog?* . The answer to this question is no, because the antecedent in the formula for a generic answer is *not* detachable; it is not an independent, closed formula. Rather, it is tied to its consequent through the property of variable sharing. In addition, the antecedent of a generic answer does not question whether all known entities are members of some category; rather, it describes a category to which a property is ascribed by its consequent. The question *is everything a dog?* which was discussed previously with regard to Skolem functions, arises from a hypothetical answer where *everything is a dog* appears as a closed, independent, fully detachable antecedent.

Hypothetical answers are less informative than specific and generic answers in that they neither present a satisfying instance for a question nor a general description of a satisfying instance, but they do provide information about what needs to be known in order for an answer to be forthcoming.

In the examples shown earlier in this section, it is seen that *all* answers are hypothetical when there is a complete lack of information about individuals mentioned in a question. Hypotheses about properties of such individuals must be made in order to provide answers. The only situation under which relevant answers can be found regarding a completely unknown individual is in the presence of *unconstrained generic information*. For example, if a knowledge base contains a universally quantified formula such as $\forall\, x\ (P\ x)$, then $(P\ a)$ can be inferred for any individual.

Hypothetical answers will be present when there is *disjunctive* information in a knowledge base. A disjunction, where neither disjunct is known to be true or known to be false, represents a lack of information. If $\mathcal{A} \vee \mathcal{B}$ is known, and neither $\mathcal{A}$ nor $\mathcal{B}$ is known to be true, then hypothetical answers will arise, as shown in the example given in Table 4.14.

It is clearly not always practical, in the face of lack of information, to generate *all* possible hypothetical answers: this can lead to a combinatorial explosion of unwanted information. Recent work with large knowledge bases demonstrates the gravity of this problem [Chalupsky and Russ, 2002]. We have offered a short discussion of some ways in which certain hypothetical answers may be considered more interesting than others. In particular, if a class of hypothetical answers is deemed *uninteresting* then this information can be used to help prune the search space of an answering agent. If the only available responses from a particular answering agent are hypothetical answers, it is likely that a questioner will go elsewhere in hope of better results. Unlike the abductive reasoner, which must select from among a potential deluge of possibilities, the answering agent bears no such burden: the onus is on the questioner to perform the selection and to terminate the question answering interaction.

In summary, hypothetical answers arise naturally in the course of backward reasoning, and often function solely as resolution intermediates. When hypothetical answers persist, they indicate a lack of information.

4.5.6 Summary of Hypothetical Answers

In summary, hypothetical answers provide a way of giving a questioner information in the face of incomplete knowledge of an answering agent. They share some properties of abductive hypotheses, but there are important differences. The following points can be made about hypothetical answers:

- Hypothetical answers whose only difference is the level of generality are *all* interesting.
- The hypothetical component of an answer is an independent, closed formula that represents a question that is currently unanswerable by the answering agent.
- The hypothetical component of an answer may contain existentially quantified, locally scoped variables.
- Hypothetical answers have a purely syntactic definition that is domain-independent and does not rely on a predefined set of predicates.
- The practical problem of which hypothetical answers are interesting remains an open research issue.

4.6 A Complete Characterization of Answers

Question answering in a resolution theorem prover is initiated when the set of question clauses $\mathcal{QC}$ associated with a question $\mathcal{Q}$ is combined with an existing knowledge base $\mathcal{K}$. It is from the set $\mathcal{QC} \cup \mathcal{K}$ that answer clauses are derived. Figure 4.1 shows the important subsets derived from the theory $Th(\mathcal{K} \cup \mathcal{QC})$, which is the closure of the set of clauses that an answering agent may derive.

The theory is partitioned into two sets: $Th(\mathcal{K})$ and $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$. Clauses in $Th(\mathcal{K})$ are derived using only information from $\mathcal{K}$, while clauses in $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ include those descended solely from the question clauses as well as those descended from the question clauses *and* clauses in $\mathcal{K}$.

The set $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$, which contains all the clauses descended from the question clauses plus the knowledge base, is partitioned into three subsets labeled specific, generic, and hypothetical for the three answer clause types described in this chapter. There is a further division of this set into subsumed and non-subsumed clauses. The subsumed clauses shown in the outer ring of the inner set in Figure 4.1 are subsumed by clauses in the inner circle of that set. In other words, all of the subsumed clauses can be derived from clauses in the non-subsumed clause set.

In Figure 4.1 the clauses descended from $\mathcal{QC}$ are members of the set $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$: these are precisely the relevant clauses we defined earlier in this chapter. The next result establishes that the set $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ is precisely the set of relevant clauses.

Theorem 4.4 ($Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ is the set of relevant clauses) *A clause $\mathcal{C}$ is relevant if and only if it is a member of the set $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$.*

Proof: A clause $\mathcal{C}$ is relevant if and only if it is a descendent of $\mathcal{QC}$ by Theorem 4.2. The set $Th(\mathcal{K} \cup \mathcal{QC})$ is the entire set of clauses available to the answering agent, where $\mathcal{K}$ is the knowledge of the answering agent and $\mathcal{QC}$ is the set of question clauses. The set $Th(\mathcal{K} \cup \mathcal{QC})$ may be partitioned into the following subsets:

- $Th(\mathcal{K})$: This is the set of clauses that are derived solely from clauses in $\mathcal{K}$. None of these clauses contain answer literals because no clause in $\mathcal{K}$ contains an answer literal.
- $Th(\mathcal{QC})$: This is the set of clauses that are derived solely from clauses in $\mathcal{QC}$. All of these clauses contain answer literals by Theorem 4.2.
- $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K}) \setminus Th(\mathcal{QC})$: This is the set of the rest of the clauses derivable from $\mathcal{K}$ and $\mathcal{QC}$. Because clauses derived solely from $\mathcal{K}$ are excluded, as are clauses derived solely from $\mathcal{QC}$, then clauses in this set are those with ancestors in both $\mathcal{K}$ and $\mathcal{QC}$. Since they have at least one ancestor in $\mathcal{QC}$, they will *all* contain answer literals by Theorem 4.2.

All of the clauses in two of the three subsets comprising this partition contain clauses descended from $\mathcal{QC}$: only $Th(\mathcal{K})$ contains no clauses descended from $\mathcal{QC}$. Therefore the set:

$$[Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K}) \setminus Th(\mathcal{QC})] \bigcup Th(\mathcal{QC})$$

is precisely the set of relevant clauses. Simplifying the expression yields $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ as *the* set of relevant clauses. $\square$

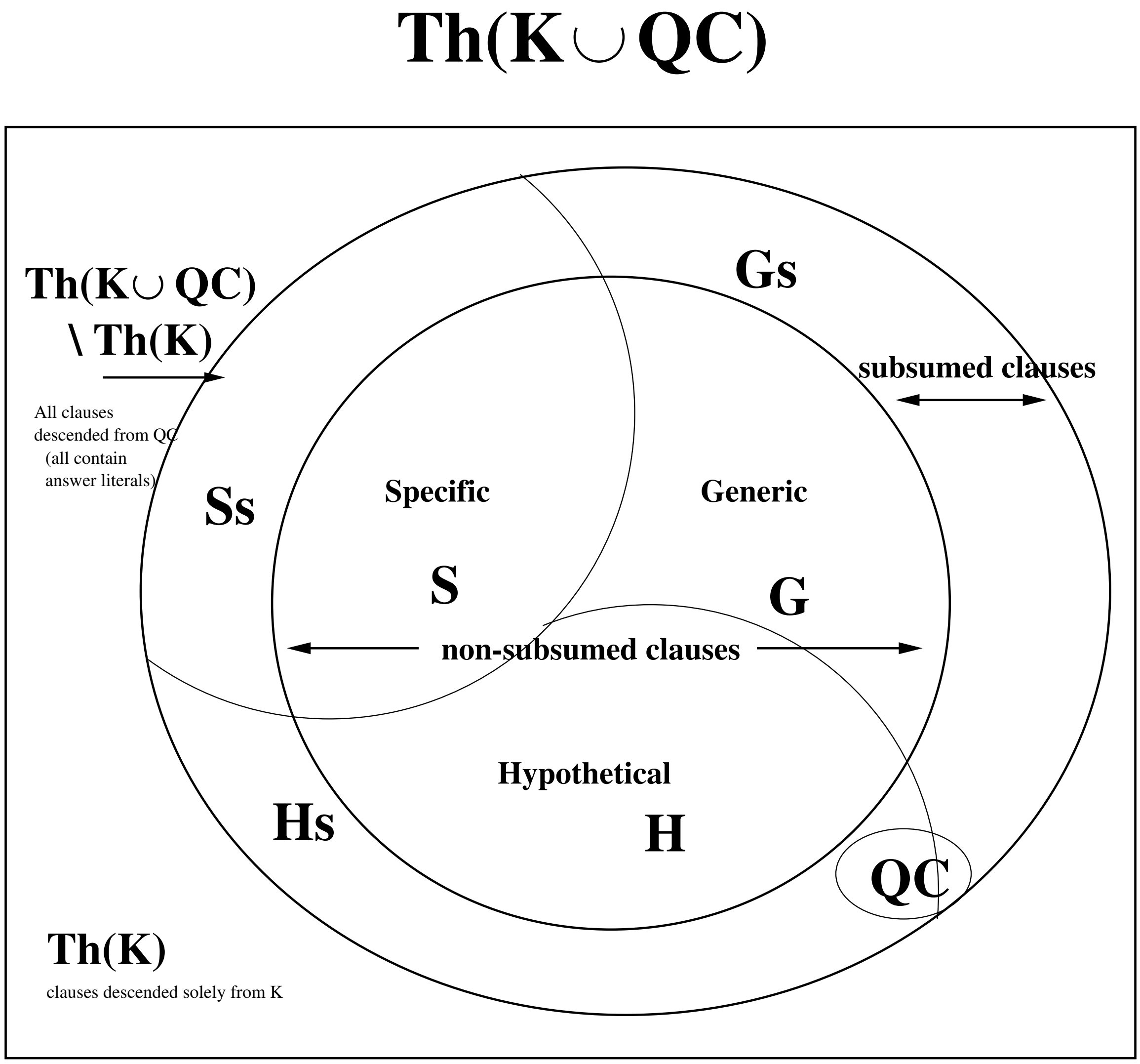

Figure 4.1: The theory of $(\mathcal{K} \cup \mathcal{QC})$: $\mathcal{K}$ is the clauses comprising the knowledge base of the reasoner; $\mathcal{QC}$ is the set of question clauses; $Th(\mathcal{K})$ is the set of clauses descended solely from $\mathcal{K}$; $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ is the set of clauses descended from both $\mathcal{K}$ and $\mathcal{QC}$; $\mathcal{S}$, $\mathcal{G}$, and $\mathcal{H}$ are the sets of specific, generic, and hypothetical answer clauses respectively; $\mathcal{S}_s$, $\mathcal{G}_s$, and $\mathcal{H}_s$ are subsumed specific, generic, and hypothetical answer clauses.

As Figure 4.1 shows, the set $Th(\mathcal{K} \cup \mathcal{QC}) \setminus Th(\mathcal{K})$ is partitioned into three parts: specific, generic, and hypothetical. The form of a clause determines the partition of which it is a member. In considering the form of a clause, it must be remembered that clauses are comprised of literals. Literals may either be answer or non-answer literals. By the previous results, every clause in this set contains at least one answer literal. The following is an exhaustive list of possible clause forms in this set and their associated answer clause types.

- **Specific:** all literals are ground answer literals and do not contain Skolem constants or functions
- **Generic:**
  - **either:** all literals are answer literals and at least one contains a variable or Skolem term
  - **or:** there are non-answer literals that all contain variables, and all such literals are in the closure of variable sharing with the answer literals
- **Hypothetical:** there is at least one non-answer literal that is *not* in the closure of variable sharing with the answer literals

Theorem 4.5 (Specific, Generic, and Hypothetical Clauses Partition the Set of Relevant Clauses) *Every clause descended from the question clauses is either a specific, generic, or hypothetical answer clause based on its form.*

Proof: The proof is divided into two cases based on the form of clauses. As shown above, all clauses in question contain at least one answer literal. Case 1 covers clauses *all* of whose literals are answer literals, while Case 2 covers clauses that contain at least one non-answer literal. Clearly all clauses fall into one of these two cases.

- **Case 1: Clauses contain only answer literals**

  We partition the set of clauses containing only answer literals into two sets based on their form. These clauses *either* contain all ground terms, none or which are Skolem functions or constants, *or* contain at least one non-ground or Skolem term.

  1. **Answer literals contain only ground terms with no Skolem constants or functions**

     By definition this is a *specific* answer clause.

2. **Answer literals contain at least one variable or a Skolem constant or function**

   When a clause contains only answer literals with at least one variable or Skolem constant or function, it is a *generic* answer clause. In particular, it is associated with what we have referred to as an *unconstrained* generic answer because it lacks the category restrictions that might be placed on universally quantified variables in a rule antecedent.

Clearly this is an exhaustive description of clauses containing only answer literals, and each such clause belongs to either the class of specific or the class of generic answer clauses.

- **Case 2: Clauses contain one or more non-answer literals**

  If clauses do not contain only answer literals, they obviously contain at least one non-answer literal. We partition these clauses into two groups: (1) At least one non-answer literal contains only ground terms and (2) there is at least one variable in each of the non-answer literals. Clearly all clauses containing non-answer literals fall into exactly one of these two categories.

  1. **At least one non-answer literal contains only ground terms**

     When some non-answer literal contains only ground terms it obviously contains no variables, hence is not in the closure of variable sharing of the answer literals. This means it is part of the hypothetical component of an answer, thus is part of a *hypothetical* answer clause.

  2. **There is at least one variable in each non-answer literal**

     This set of clauses contain non-answer literals none of which is pure ground, meaning each non-answer literal contains at least one variable. We partition this set into two subsets: (1) all non-answer literals are in the closure of variable sharing of the answer literals and (2) some non-answer literal is not in the closure of variable sharing of the answer literals.

     (a) **All non-answer literals are in the closure of variable sharing of the answer literals**

     This is the definition of the set of *generic* answer clauses.

     (b) **Some non-answer literal is not in the closure of variable sharing of the answer literals**

     These clauses are by definition hypothetical answer clauses.

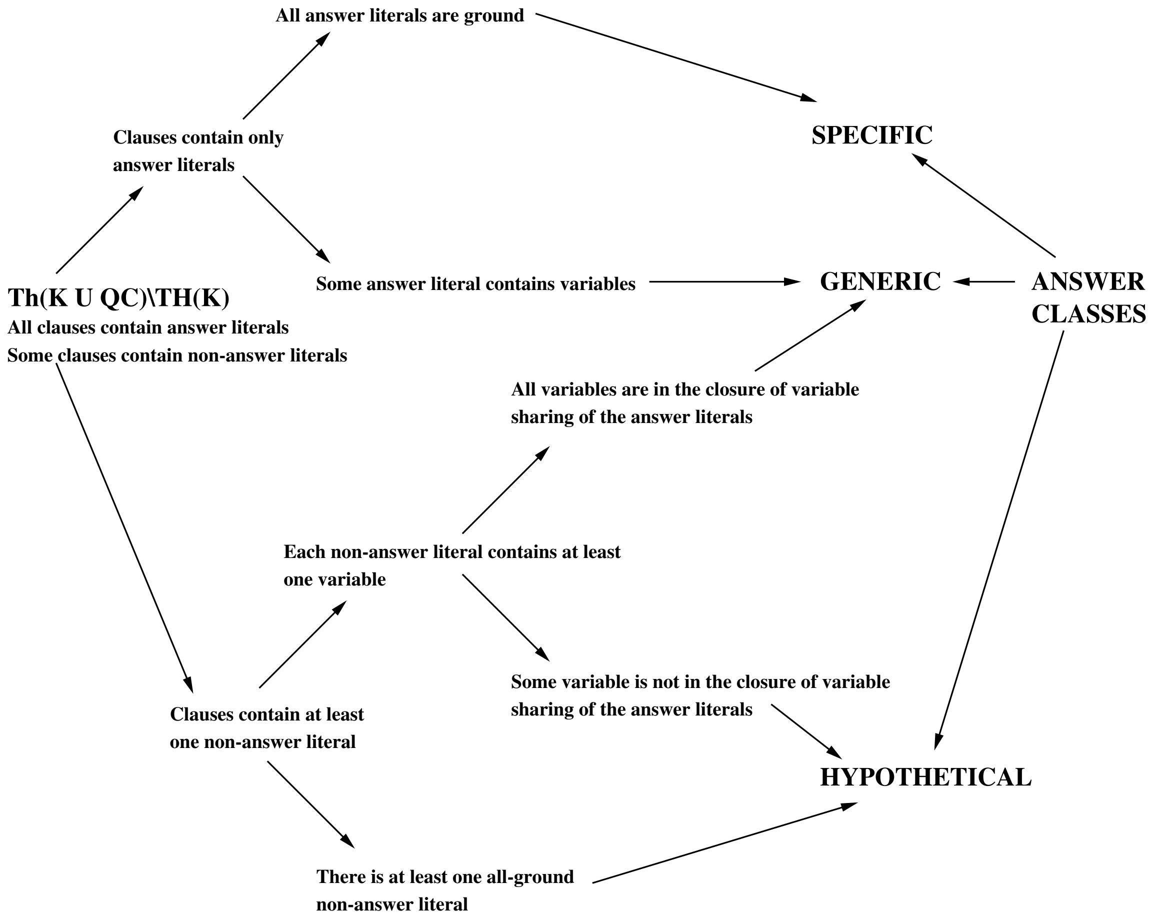

Figure 4.2: Answer Classes and their relation to Clause Form

We have shown that all relevant clauses are either specific, generic, or hypothetical answer clauses by partitioning the set of relevant clauses based on form. Each partition and subpartition described has clearly accounted for *all* clauses in the set on which the partition is defined. Each partition has been associated with one of the three answer classes as stated.

□

Figure 4.2 shows a graphical representation of the partitions of the set of relevant clauses and the relationship of these partitions to the three answer classes.

At this point we have constructed a solid description of the set of relevant clauses, and have assured ourselves that *all* such clauses are associated with answers from one of the three answer classes introduced: specific, generic, or hypothetical. The following formula provides a complete characterization of answers in a resolution theorem prover in accordance with our previous results:

$$\exists(w_1 \dots w_{n_1})\mathcal{H}\,(w_1 \dots w_{n_1},\, a_1 \dots a_{m_1})$$
$$\Rightarrow\ [\forall(x_1 \dots x_{n_2},\ y_1 \dots y_{n_3})\ [\mathcal{G}\,(x_1 \dots x_{n_2},\ y_1 \dots y_{n_3},\ b_1 \dots b_{m_2})$$
$$\Rightarrow\ \forall(z_1 \dots z_{n_4})\ \mathcal{Q}\,(x_1 \dots x_{n_2},\ z_1 \dots z_{n_4},\ c_1 \dots c_{m_3})]]$$

This formula can be schematized as $\mathcal{H} \Rightarrow [\mathcal{G} \Rightarrow \mathcal{Q}]$ which makes clear the three components of an answer. The antecedent, $\mathcal{H}$, is what we referred to in the previous section on hypothetical answers as the *hypothetical component* of an answer. It is *optional*: when it is present, it indicates a hypothetical answer. The consequent of the formula is what we referred to as the *answer component*. All answers have a non-empty answer component. The answer component in turn has two parts. $\mathcal{G}$ is the *generic component* of the answer. It is *optional*: its presence indicates a generic answer. Finally, $\mathcal{Q}$ is the *question component*, which is comprised of answer literals. *All* answers must contain a question component; it is the only required part of the formula. An all ground question component indicates a specific answer, and a question component that contains one or more variables indicates a generic answer. Whenever the hypothetical component is present, it indicates a hypothetical answer. When a proof is required in order for there to be an answer (Green [1969a, 1969b], Luckham and Nilsson [1971], Prolog, Otter [McCune, 1994], and theorem provers in general) the answer will *always* consist of only the question component. This limits these approaches to question answering to providing either specific or unconstrained generic answers.

4.6.1 The Hypothetical Component

The existentially quantified antecedent of the formula above is the *hypothetical component*, and has the following form:

$$\exists(w_1 \dots w_{n_1})\mathcal{H}\,(w_1 \dots w_{n_1},\, a_1 \dots a_{m_1})$$

The parts of the hypothetical component of the formula are as follows:

- $\mathcal{H}$ stands for the predicates in the hypothetical component.

- The w_i are variables not shared with the answer component. They may be present or absent, depending on the form of the hypothetical. If they are present then the hypothetical has the form of a rule or generic; if the hypothetical contains no variables then it falls into the *conditional* answer category defined by Demolombe [Demolombe, 1992, Demolombe, 1997] as well as having the flavor of an abducible hypothesis.

- The a_{m_i} are constants. As with variables, a hypothetical component may or may not contain constants, depending on its form.

4.6.2 The Generic Component

The antecedent $\mathcal{G}$ of the answer component is the *generic component*, and has the following form:

$$\forall(x_1 \ldots x_{n_2},\; y_1 \ldots y_{n_3})\; \mathcal{G}\; (x_1 \ldots x_{n_2},\; y_1 \ldots y_{n_3},\; b_1 \ldots b_{m_2})$$

The parts of the generic component of the answer are as follows:

- $\mathcal{G}$ stands for the predicates in the generic component.

- The x_i are variables that *overlap* with the variables in the question component, that is, the variables shared by both the generic and question component of the answer. There must be shared variables between the generic and question components.

- The y_i are are other variables that are in the *closure of variable sharing* of the answer literals but do not *coincide* with the answer literal variables. Therefore they only appear in the generic component. This set of variables may be empty.

- The b_{m_i} are constants, and may or may not be present, depending on the form of the generic. When constants occur with variables in generic answers, these answers coincide with what Motro has termed *mixed answers* [Motro, 1994].

4.6.3 The Question Component

The consequent $\mathcal{Q}$ of the answer component is the *question component* of an answer. The question component has the following form:

$$\forall(z_1 \ldots z_{n_4})\; \mathcal{Q}\; (x_1 \ldots x_{n_2},\; z_1 \ldots z_{n_4},\; c_1 \ldots c_{m_3})$$

The parts of the question component of the formula are as follows:

- $\mathcal{Q}$ stands for the predicates in the question component.

- The z_i are variables contained only in the question component. When present they signal a generic answer.

- The x_i are variables shared with the generic component of the answer.

- The c_{m_i} are constants in the question component, which may or may not appear.

4.7 Multiple Answer Clauses

When more than one answer clause is present, each such clause is associated with an answer. As indicated by the semantics of clause form, these answers are implicitly conjoined, thus representing a conjunctive answer.

When a question-answering system returns only specific answers, multiple answers are not problematic. This is due to the fact that specific answers are all at the same level of generality. Once generic and hypothetical answers are incorporated into a question-answering system, however, this no longer holds. Above, we mentioned the problem of potentially misleading a questioner when multiple generic answers are allowed. A conjoined answer such as *Bob likes tuna and Bob likes fish*, where there is a taxonomic relationship between the categories mentioned in the two conjuncts, may be misleading if the questioner fails to understand the fact that *tuna* is a type of *fish*. A specific and a generic answer that, taken together, mention an instance and the category to which the instance belongs, may similarly mislead. For example, if Ming Teh is a Chinese restaurant, then the answer *Bob likes Ming Teh and Bob likes Chinese restaurants* may lead a questioner to believe that there is no relationship between *Ming Teh* and *Chinese restaurants*.

Thus, the presence of answers at different levels of generality presents a challenge to the question-answering agent in generating responses. While not a topic considered in this dissertation, it is clearly an important problem in discourse analysis.

4.8 Disjunctive Answers

Any of the three answer types may contain more than one answer literal. As indicated by the semantics of clause form, multiple answer literals are disjoined, indicating a disjunctive answer. The general characterization of answer that we previously presented includes those with multiple answer literals. To clarify, an answer involving multiple answer literals has the following general form:

$$\mathcal{H} \Rightarrow [\mathcal{G} \Rightarrow (Q_1 \vee Q_2 \vee \ldots \vee Q_n)]$$

Clauses containing multiple answer literals have been termed *disjunctive answers*, and are discussed in the early question answering literature by Green [1969a], as well as in the subsequent literature on question answering. A disjunctive answer presents different possible answers, at least one of which is true. We distinguish here between hypothetical answers and disjunctive answers by identifying disjunctive answers with clauses containing *multiple answer literals*. Of course, a hypothetical answer might be written in the form of a disjunction, but in a disjunctive answer, there is more than one literal that has the same structure as the original question. In fact, for each of the answer literals contained in a disjunctive answer, the literal can be unified with the literal that represents the original question (provided the original question maps to a single literal in clause form). This observation allows us to distinguish between true hypothetical and disjunctive answers, and leads to the notion of a canonical form for disjunctive answer clauses.

We have described how to partition the literals in a *clause* into three sets corresponding to the hypothetical, generic, and question components of an answer based on the property of variable sharing. When there is more than one answer literal, it might be asked whether the answer literals as a group can be partitioned in a manner that consolidates related information. Some of the answers that are disjoined may share variables, while others don't. Similarly, some of the answers may share constants. We will suggest a way of partitioning disjunctive answers into related groups later in this section. First, we will look at some examples of disjunctive answers.

4.8.1 Examples of Disjunctive Answers

Green [1969a, page 222] uses the automaton shown in Figure 4.3 to illustrate disjunctive answering. The letters along the edges indicate actions, and the letters at the vertices represent states. The goal state is labeled d. The following are the rules and facts given by Green to describe the automaton:

M1. $AT(a,\ s_0)$

M2. $(\forall s)[AT(a,\ s) \rightarrow (AT(b,i(s)) \vee AT(c,i(s)))]$

M3. $(\forall s)[AT(b,\ s) \rightarrow AT(d,j(s))]$

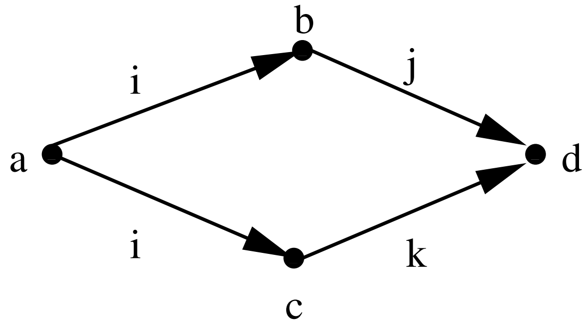

Figure 4.3: Automaton used to Illustrate Specific and Generic Disjunctive Answers

M4. $(\forall s)[AT(c,\ s)\ \rightarrow\ AT(d,k(s))]$

The question posed in Green's paper is whether the goal, d, is reachable; represented as $(\exists s)AT(d,\ s)$. The answer given by Green is the following:

Yes, $s\ =\ j(i(s_0))$ or $s\ =\ k(i(s_0))$

This disjunctive specific answer conveys the information that, starting from state s_0, either action i followed by action j *or* action i followed by action k will result in reaching the goal state d.

The intermediate steps of the proof presented by Green contain the following example of a disjunctive generic answer (written using our notation for representing generic answers):

$\forall s[(AT\ a\ s)\ \rightarrow\ [(AT\ d\ j(i(s)))\ \vee\ (AT\ d\ k(i(s)))]]$

Note that this is simply a generalization of the specific answer given by Green. The information conveyed by this answer is that, from state a, either action i followed by action j or action i followed by action k will result in reaching the goal state d.

Disjunctive hypothetical answers may also be found. Consider Figure 4.4, which shows a second automaton that is an extension of the automaton presented by Green. This new automaton adds the state e, which is the new goal state. State e is reachable from state d by taking either action l or action m. The question asked is whether the goal state e is reachable. One of the disjunctive hypothetical answers produced is the following:

$(\neg(at\ b\ (i(s_0))))\ \rightarrow\ [(at\ e\ (m(k(i(s_0)))))\ \vee\ (at\ e\ (l(k(i(s_0)))))]$

This answer indicates that, if it is *not* the case that action i from state s_0 resulted in moving to state b, then either action i followed by k, then m, or action i followed by k, then l, will result in moving to the goal state e. While this is a disjunctive specific hypothetical answer, there are also, obviously,

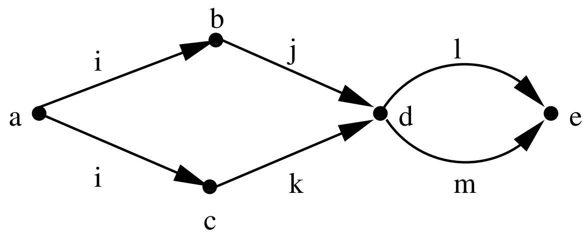

Figure 4.4: Automaton used to Illustrate Hypothetical Disjunctive Answers

disjunctive generic hypothetical answers.

A further example demonstrates the problem of different answer clauses that give rise to logically equivalent answers. In this case, there is one answer clause comprising two answer literals, and two other answer clauses each with one literal in the hypothetical component and one answer literal. The knowledge base consists of the following single fact:

$((LIKES\ BOB\ FRED)\ \vee\ (LIKES\ BOB\ RITA))$

which is represented in clause form as:

$\{(LIKES\ BOB\ FRED),\ (LIKES\ BOB\ RITA)\}$

The question to be asked is whether there is something Bob likes, written as:

$\exists x(LIKES\ BOB\ x)$

corresponding to the question clause:

$\{(\neg(LIKES\ BOB\ x)),\ (ANSWER(LIKES\ BOB\ x))\}$

As a proof proceeds, the following three resolvents, all answer clauses, are produced:

1. $\{((ANSWER(LIKES\ BOB\ FRED)),\ (LIKES\ BOB\ RITA))\}$
2. $\{((ANSWER(LIKES\ BOB\ RITA)),\ (LIKES\ BOB\ FRED))\}$
3. $\{((ANSWER(LIKES\ BOB\ FRED)),\ (ANSWER(LIKES\ BOB\ RITA)))\}$

These clauses cannot be unified with one another. Written in standard FOPC according to our answer form, the answers corresponding to these answer clauses are:

1. $(\neg(LIKES\ BOB\ RITA))\ \rightarrow\ (LIKES\ BOB\ FRED)$
2. $(\neg(LIKES\ BOB\ FRED))\ \rightarrow\ (LIKES\ BOB\ RITA)$
3. $((LIKES\ BOB\ RITA)\ \vee\ (LIKES\ BOB\ FRED))$

The first two appear to be hypothetical specific answers, while the last is a disjunctive specific answer. With the "protective" *ANSWER* predicate removed in the process of converting the clauses to answer form, it is apparent that the three answers generated by the answer clauses are logically equivalent.

The following example illustrates disjunctive generic answers. Consider the following knowledge base, represented in clause form (variables are preceded with a question mark) with accompanying English glosses:

1. $\{(LIKES\ MOTHERTHERESA\ ?4),\ (LIKES\ DALAILAMA\ ?4)\}$

   Everyone is liked by either Mother Theresa or by the Dalai Lama

2. $\{(LIKES\ ?8\ JOE),\ (LIKES\ ?8\ BOB)\}$

   Everyone likes either Joe or Bob

3. $\{(\neg(STUDENT\ ?12)),\ (LIKES\ ?12\ HOMEWORK),\ (LIKES\ ?12\ PARTIES)\}$

   Students like either homework or parties

The question asked is whether the likes relation holds between some two entities, the FOPC form of which is $(\exists x(\exists y(likes\ x\ y)))$, with the following associated question clause:

$$\{(\neg(LIKES\ ?27\ ?29)),\ (ANSWER(LIKES\ ?27\ ?29))\}$$

Three "clusters" of equivalent answers are found, each corresponding to one of the three (disjunctive) clauses in the original knowledge base. One of these clusters is the following:

1. $\{(LIKES\ DALAILAMA\ ?45),\ (ANSWER(LIKES\ MOTHERTHERESA\ ?45))\}$

2. $\{(LIKES\ MOTHERTHERESA\ ?49),\ (ANSWER(LIKES\ DALAILAMA\ ?49))\}$

3. $\{(ANSWER(LIKES\ MOTHERTHERESA\ ?666)),\ (ANSWER(LIKES\ DALAILAMA\ ?666))\}$

These clauses are clearly not equivalent, but consider the answers corresponding to these clauses, shown in FOPC below:

1. $\forall ?x0[(\neg(LIKES\ DALAILAMA\ ?x0)) \rightarrow (LIKES\ MOTHERTHERESA\ ?x0)]$

2. $\forall ?x0[(\neg(LIKES\ MOTHERTHERESA\ ?x0)) \rightarrow (LIKES\ DALAILAMA\ ?x0)]$

3. $\forall ?x0[(LIKES\ MOTHERTHERESA\ ?x0) \vee (LIKES\ DALAILAMA\ ?x0)]$

These are, clearly, logically equivalent. This illustrates the need for recognizing disjunctive answers *even when their associated clause form is that of a hypothetical or generic answer*.

4.8.2 Origins of Disjunctive Answers

The above examples have clearly demonstrated that, in the presence of disjunctive information in a knowledge base, disjunctive answers may arise. There is another way in which disjunctive answers may arise, namely, when a disjunctive question is asked. For example, if you ask the question whether there is something that is a dog or a cat ($\exists x[(dog\ x) \ \vee \ (cat\ x)]$), and the knowledge base contains the fact that Fluffy is a cat ($(cat\ Fluffy)$), then the disjunctive answer *Fluffy is a dog or Fluffy is a cat* arises.

4.8.3 Structure of Disjunctive Answers

It is clear that the above examples demonstrate the way in which distinct answer clauses may map to a single *answer* in the presence of disjunctive information. This general phenomenon arises when the following condition is met: at least one non-answer literal in a clause "looks like" the literal corresponding to the original question. More formally, at least one non-answer literal in a clause unifies with the literal corresponding to the original question. These "unifying literals", along with all of the arguments to the $ANSWER$ predicates, share a common structure. The literal corresponding to the question is the *most general* representative of this structure. Recognizing different answer clauses that correspond to the same answer is important: absent this, the number of equivalent answers that may be generated corresponds to all of the possible ways of arranging subsets of the disjuncts between the antecedent and consequent minus one: one is subtracted because there must be at least one answer literal present (according to our definition). Thus, for n disjuncts there are $(2^n - 1)$ equivalent answers. When not appearing as arguments to $ANSWER$ predicates, some of the literals may appear in the generic component of the answer while others may be found in the hypothetical component. Because no answer literal can appear in *both* of these components, the number of possible arrangement of the answer literals is the same in spite of this complication. While a traditional proof may not generate all of the equivalent answer clauses, a question-answering agent designed to continue reasoning regardless of the presence of a proof will, unfortunately, find all $(2^n - 1)$ of them given sufficient resources.

In considering where the literals with a potential dual role – as arguments of $ANSWER$ predicates or as part of the generic or hypothetical component of an answer – should be placed, there are two extreme possibilities. Namely, all literals excepting one, which appears in the question component of the answer, may be placed in the appropriate antecedent, or all literals may be placed in the question component as arguments to $ANSWER$ predicates. Between these extremes there is a range.

There are clearly two separate groups of literals that may be involved in a disjunctive answer: those that align with the hypothetical component, and that that align with the generic component. Consider the case where some of the literals that could, logically, end up as arguments to $ANSWER$ predicates might alternatively appear in the hypothetical component of the answer. As previously discussed, the hypothetical component of an answer is associated with a detachable question. However, there are no *a priori* criteria for deciding *which* of the potential arguments of $ANSWER$ predicates should, instead, be incorporated into the hypothetical answer component. There is a strong chance that, absent constructing all possible variant hypothetical answers, a questioner might be mislead by this form.

Consider the following example, where all disjuncts are present in the question component. A disjunctive answer to the question *Where is John?* might be *John is at Bill's house or John is at Bob's house* . While we typically think of disjunctive answers as information-lacking, this may be as informative as is necessary: knowing that John is at *someone's* house may completely satisfy the questioner. On the other hand, the logically equivalent answer *If John isn't at Bill's house then John is at Bob's house* , with detachable hypothetical component *John isn't at Bill's house* and associated question *Is it the case that John is not at Bill's house?* , may not be as useful. Notice that the commonality among the disjuncts is lost when one or more are presented as a hypothesis to be settled instead of as part of a larger answer.

Rather than literals that may appear in either the hypothetical or the question component of an answer, consider literals that may appear either in the generic or the question component of an answer. Unlike the hypothetical component, which has a "life of its own" apart from the other answer components, the generic component is inextricably bound to the question component. This means that, even if some of the literals that might ultimately be part of the question component appear in the generic component of an answer, at least all of these literals will appear together as a cohesive answer. Still, it can be argued that grouping these literals in the question component helps to capture generalities that they share. For example, in response to the question *Does anyone like anyone* (assume that the universe of discourse is restricted to people for purposes of illustration), the answer *If it's not the case that Mother Theresa likes someone then the Dalai Lama likes him* , and an alternative formulation *Everyone is liked either by Mother Theresa or by the Dalai Lama* , convey the same information. However, the second answer better expresses that everything is liked by *something*. This example is similar to the earlier example: there is a potential for a subset of the disjuncts comprising an answers to fall into the same category. In this sense, a disjunctive answer may function in the same manner as a single, nondisjunctive answer. Obviously, the recognition that different terms represent the same category is a function of the questioner, not the answering agent. This type of processing on the part of a questioner may render a disjunctive answer more informative than its surface structure indicates.

Whether the literals that may appear as arguments to an *ANSWER* predicate are presented as arguments to *ANSWER* predicates or appear in the hypothetical or generic component of an answer, there is clearly no benefit to maintaining multiple clauses that map to the same answer in an answer set. Because the *clauses* in question are not logically equivalent, we are not suggesting any changes in terms of the reasoning process and the knowledge base. Rather, once an answer is found it can be transformed into a canonical form so that there is a one-to-one mapping between answer clauses and answers.

There are compelling reasons to place all potential arguments to *ANSWER* predicates in the question component of the answer. These literals match the question asked, thus, placing them in the *question component* makes sense. Placing these literals in the question component clearly separates disjunctive answers from hypothetical answers, in addition to saving the questioner from unnecessary hypothetical answers. Finally, when grouped together these literals may evince commonalities that would not be evident if they were placed in separate parts of the answer.

Thus, we propose a canonical form for answer clauses that maximizes the number of literals in the question component. A literal not initially in the question component may be moved there as long as it unifies with the original question. "Moving it there" involves rewriting it as the single argument to an *ANSWER* predicate. Note that the rewritten clause is *not* logically equivalent to the original clause, but that the answer associated with the rewritten clause is logically equivalent to the answer associated with the original clause. Thus, this rewriting process can not occur in the context of reasoning, and must occur after an answer has been discovered but before it is compared to the answers in the current answer set.

For example, given the following clause:

$$\{(LIKES\ DALAILAMA\ ?x0),\ (ANSWER(LIKES\ MOTHERTHERESA\ ?x0))\}$$

and initial question $(\exists x(\exists y(likes\ x\ y)))$, with corresponding clause form (for the question only) of $(likes\ x\ y)$, we see that $(LIKES\ DALAILAMA\ ?x0)$ unifies with the original question. Thus, the clause shown can be rewritten as:

$$\{(ANSWER(LIKES\ MOTHERTHERESA\ ?x0)),\ (ANSWER(LIKES\ DALAILAMA\ ?x0))\}$$

before it is checked against the current answer set.

Definition 4.14 Answer Clause in Canonical Form. *An answer clause C is in canonical form if all of its literals that unify with the literals in the original question clauses are arguments of ANSWER predicates.*

The procedure described above can be used to put an answer clause into the canonical form. It might be argued that there is no need to put clauses into this form since, eventually, a reasoner will produce the canonical form on its own. This objection is easily overcome. First, it may be a long time before the canonical form is discovered, and in the meanwhile the answer set will be filling up with answer clauses that map to the same answer. Second, in the spirit of anytime reasoning, we would like to have a better answer available as early as possible.

We can now examine the two sets of equivalent answers shown above and note which are in canonical form according to this definition. In the following list of answer clauses, the third clause is in canonical form and the other two are not.

1. $\{((ANSWER(LIKES\ BOB\ FRED)),\ (LIKES\ BOB\ RITA))\}$

2. $\{((ANSWER(LIKES\ BOB\ RITA)),\ (LIKES\ BOB\ FRED))\}$

3. $\{((ANSWER(LIKES\ BOB\ FRED)),\ (ANSWER(LIKES\ BOB\ RITA)))\}$

Another example mentioned above involved the following three answer clauses:

1. $\{(LIKES\ DALAILAMA\ ?50)),\ (ANSWER(LIKES\ MOTHERTHERESA\ ?50))\}$

2. $\{(LIKES\ MOTHERTHERESA\ ?89),\ (ANSWER(LIKES\ DALAILAMA\ ?89))\}$

3. $\{(ANSWER(LIKES\ MOTHERTHERESA\ ?666)),\ (ANSWER(LIKES\ DALAILAMA\ ?666))\}$

In this case the third answer clause is the only one given in canonical form.

This form is defined for questions that map to a single literal. A complex question may be itself a disjunction, in which case two literals would have to match the question rather than one. Alternatively, a question may be conjunctive, in which case any attempt to "collapse" answer clauses that map to equivalent answers may involve different clauses. A further exploration of these issues is not undertaken in this dissertation.

4.8.4 Relationship between Disjuncts

Another issue regarding the structure of a disjunctive answer concerns the way in which different parts of the disjunction are related. We know that all of the arguments to the $ANSWER$ predicates have the same predicate structure. Thus, relationships between subsets of these literals may be seen by comparing their arguments, which may be variables, functions, or simple ground terms.

Consider, for example, a question involving the two-place predicate *likes*: $(\exists x(\exists y(likes\ x\ y)))$. A possible disjunctive answer might be *John likes Mary or John likes Sue or students like John or students like parties or everyone likes baseball or dogs like bones or dogs like cats* . This is obviously an unusual answer, but it will serve to illustrate the way in which various parts of a disjunctive answer may be separately grouped.

First, let's examine this answer in clause form, listing the glosses for each part of the answer along with their associated literals. Note that some literals share variables, thus are included together in the same group.

1. John likes Mary – (ANSWER(likes John Mary))

2. John likes Sue – (ANSWER(likes John Sue))

3. students like John or students like parties – (¬(student x)),(ANSWER(likes x John)),
   (ANSWER(likes x parties))

4. everyone likes baseball – (ANSWER(likes baseball y)))

5. dogs like bones or dogs like cats – (¬(dog z)),(ANSWER(likes z bones)),(ANSWER(likes z cats))

We recognize two non-answer literals corresponding to the generic component of the answer: (¬(student x)) and (¬(dog z)). The rest of the literals are *ANSWER* literals, comprising the question component of the answer. Thus, the general answer form dictates that the answer be written as follows:

$$(\forall x(\forall z[((student\ x) \wedge (dog\ z)) \rightarrow$$
$$((likes\ John\ Mary) \vee (likes\ John\ Sue) \vee (likes\ x\ John) \vee (likes\ x\ parties)$$
$$\vee (\forall y(likes\ baseball\ y)) \vee (likes\ z\ bones) \vee (likes\ z\ cats))]))$$

Unfortunately, this answer fails to associate what are clearly related parts of the answer. A more familiar grouping of terms might lead to a phrasing of the answer as follows: *John likes Mary or Sue, or students like John or parties, or everyone likes baseball, or dogs like bones or cats* . This grouping might be written using the following logical representation:

$$[(likes\ John\ Mary) \vee (likes\ John\ Sue)]$$
$$\vee [\forall x((student\ x) \rightarrow ((likes\ x\ John) \vee (likes\ x\ parties)))]$$
$$\vee [\forall y(likes\ baseball\ y)]$$
$$\vee [\forall z((dog\ z) \rightarrow ((likes\ z\ bones) \vee (likes\ z\ cats))))]$$

Predicates with like arguments may naturally be grouped together: all of the entities that John likes might be grouped, all of the entities everyone likes, all of the entities liked by students, and so on. The notion of these groups of answers is related not only to variable sharing among literals, but also to *constant sharing*.

While literals in a clause were selected for the generic component of an answer if they were in the variable sharing set of the answer literals, meaning they share a variable with *at least one* answer literal, in the case of disjunctive answers we may want to divide these literals apart based, once again, on variable sharing. This changes the form of the answer component of a disjunctive answer (recall that the answer component is the generic and the question components taken together) from:

$$\mathcal{G} \rightarrow (\mathcal{Q}_1 \vee \mathcal{Q}_2 \vee \ldots \vee \mathcal{Q}_n)$$

to the following logically equivalent form:

$$(\mathcal{G}_1 \rightarrow \mathcal{Q}_1) \vee (\mathcal{G}_2 \rightarrow \mathcal{Q}_2) \vee \ldots \vee (\mathcal{G}_n \rightarrow \mathcal{Q}_n)$$

where it is understood that the generic component in each of the above disjuncts, as is true for the answer in the original form, is *optional*. Thus, while some of the disjuncts are generic answers, others may be unconstrained generics, as is *everyone likes baseball*, while others may be specific answers. The latter two of these lack an antecedent. Note that the $\mathcal{G}_i$ may contain more than one variable and more than one literal, similarly the Q_i may comprise a set of literals.

Answer literals (and associated generic literals) may be grouped, as mentioned above, according to the way in which they share variables and constants. There are likely to be a number of choices of these groupings. The problem of grouping a set of predicates based on variable sharing and overlap has been considered by Choi, whose work on *P-trees* [1993] is related to the *Rete algorithm* [Forgy, 1982]. The P-tree problem is concerned with constructing a tree from a list of patterns. Adjacent pairs of patterns that share variables are grouped together, and processing of the pattern list continues recursively, adding additional patterns at different levels of the tree until pairwise checking is complete. The Rete algorithm joins patterns to common nodes in a network based on "inter-element features", namely, variable sharing among pairs of patterns. Both algorithms help to avoid maintaining duplicate representations of variables by grouping patterns that share one or more variables together. The most important effect of the grouping is to enable fast matching of objects to patterns: Rete is designed specifically for comparing a set of patterns to a very large set of objects to check for matches. The patterns considered are the left-hand sides of production rules, and a match triggers the right-hand side of the rule.

There are differences between the problem addressed by P-trees and the Rete algorithm. Most importantly, an answer stands on its own: it is not stored for repeated use as a pattern to match against objects. Thus, the need to minimize the number of nodes representing shared variables in a tree or network to help optimize repeated processing to computer matches is not present in the case of question answering. The calculation of variable sharing is similar, but, as we have previously shown, the variable sharing set is unique. There is no need to choose between variable sharing sets because all literals with overlapping variables are grouped together, and variable sharing sets are disjoint. It is not necessarily the case that the partition of the answer literals into disjoint variable sharing sets will match the output of the Rete or P-tree algorithms. For example, Choi [1993, page 83] shows a tree whose leaf nodes do not correspond to disjoint variable sharing sets: rather, non-adjacent nodes include shared variables that are represented in distinct sets. The difference in our approach is that we consider variable sharing with respect to a set rather than with respect to adjacent pairs in an ordered list.

While variable sharing provides a methodology for partially partitioning the answer literals, the problem of ground answer literals remains. Ground literals can be partitioned based on user preferences that indicate which objects are of interest, or, absent such preferences, the P-tree algorithm could be used to create a partition of the answer literals that would be likely to group similar literals together. Rather than variable overlap, the P-tree algorithm would group together literals that exhibit overlap of constants and predicates.

A further consideration is the mismatch between the typical statement *John likes Mary or Sue* and the logical $(likes\ John\ Mary) \vee (likes\ John\ Sue)$, which might be better represented as $(likes\ John\ (Mary \vee Sue))$. In other words, the non-shared arguments to the predicate *like* might be considered as a collection, namely, the collection of entities that stand in the relationship *likes* to *John*. A scheme such as this may provide a way of collapsing an answer into smaller groupings that appear to better match the sorts of answers with which we are familiar. As was the case for the grouping rewrite of a disjunctive answer described above, we have not considered collecting terms in this manner in our implementation, but leave it instead for future work. While collecting terms in this manner might be viewed as a knowledge representation task, and incorporated into the structure of the answer set, it might, alternatively, be embodied in the answer communication task. Either way, it may provide a way of recognizing equivalence classes of terms, and ultimately lead to more informative and succinct answers.

4.8.5 Summary of Disjunctive Answers

In conclusion, disjunctive answers are important. While traditionally viewed as less informative than non-disjunctive answers, this may not be the case. The informativeness of a disjunctive answer depends in part upon the taxonomy of the questioner. In addition, the informativeness of a disjunctive answer depends upon grouping shared terms together, and upon presenting terms as collections to help the questioner discover equivalence classes into which subsets of terms may fall.

In characterizing a canonical form for disjunctive answers that requires moving literals into the question component when possible, we recognize the potential for capturing commonalities among disjuncts. In addition, the canonical form allows the answering agent to avoid the problem of storing and returning redundant disjunctive answers, and it clearly differentiates disjunctive and hypothetical answers. We have demonstrated that clauses that are logically distinct may map onto answers that are *not* logically distinct. Thus, transforming answer clauses into the canonical form must occur outside of the reasoning process. In specifying the canonical form and the procedure for rewriting a clause in the form, we have assumed that the original *question* corresponds to a single literal. Beyond the scope of this dissertation is an exploration of problems that arise when the original question has a complex structure: in other words, when the structure of the question maps onto more than one literal. Also relegated to future work are a thorough explication of a grouping scheme for the parts of a disjunctive answer, and a representation of some terms as collections.

4.9 Summary

This chapter has presented an in-depth exploration of question answering in a resolution theorem prover. Theorem provers have traditionally associated the notion of answer with the presence of a proof (Green [1969a, 1969b], Luckham and Nilsson [1971], Prolog, Otter [McCune, 1994], and theorem provers in general). When a proof is required, answers are limited to either specific or unconstrained generic.

When the notion of answer was expanded, several different results were seen. Work by Cholvy and Demolombe on generic answering in a theorem prover [1986, 1990] classified rules that were derived during the reasoning process as intensional answers. Work in databases expanded upon the traditional notion of answers as (ground) facts. Reiter [1986] described the use of null values in relational databases whose use in answers is analogous to Skolem constants. Luckham and Nilsson [1971] showed that these constants correspond to arbitrary individuals in answers, and that they may be replaced by universally quantified variables. A number of researchers working in logic databases [Reiter, 1978a, Imielinski, 1987, Motro, 1989, Motro, 1994] showed that information from the *intensional* database ($\mathcal{IDB}$) and *integrity constraints* ($\mathcal{IC}$s) could be used to construct rules that represent *intensional* answers. Motro [1989, 1994, 1996] identified answers containing both extensional and intensional information as *mixed* answers. Imielinski [1987] encoded extensional information in his rules by rewriting some predicates: for example, when the question arose about which classes Imielinski could teach, his query predicate was *teach Imielinski(x)*, a unary predicate, rather than *teach(Imielinski,x)*, a binary predicate, with *Imielinski* as one of the arguments to the predicate.

Work proceeded from intensional answers to conditional answers, characterized by Sergot [1983] and Demolombe [1992, 1997] and identified by others interested in question answering [Vasey, 1986, Wolstenholme, 1988, Gilbert, 1988]. With the exception of Michalski [1993] most of the work on conditional answers has focused on *ground* answers. The hypothetical component of such answers is seen to coincide in many cases with an *abducible hypothesis* [Kakas *et al.*, 1998]. We have elucidated the similarities and differences between abductive reasoning and hypothetical question answering.

The work presented in this chapter is the core of this dissertation, providing the foundation for the explication of answer sets in the next chapter. Taking into account previous work on question answering in a resolution theorem prover, we began a thorough investigation of resolvents as answers. Existing definitions of answer literals proved too narrow to accommodate all questions expressible in FOPC. Our response to this problem was to broaden the definition of answer literal and to include the full predicate structure of the question therein. In doing so, we are able to equate our approach to answer construction with that of Luckham and Nilsson. The advantage of our approach is that there is no need to store and annotate a proof tree: we generate the answers as resolution proceeds.

An early observation was that some resolvents fail to fit into any existing characterization of answers. Identifying formally defined answer types into which resolvents may be placed enables a principled classification scheme for answers. In particular, the explication of hypothetical answers and their relationship to abduction connects question answering and hypothesis generation in a fundamental manner. The oft-presented characterization of rules as generic answers was revealed as insufficient once we considered resolvents that obviously did not accord with generic answers, yet had the form of rules. The need to clearly distinguish between generic and hypothetical answers led to the foundational definition of the variable sharing of a literal, and an explication of why it is not appropriate to consider sharing of Skolem constants and functions among literals. The incorporation of Luckham and Nilsson's [1971] insights about Skolem constants and functions and their replacement by universally quantified variables allowed us to ascribe a semantics to previously uninterpretable clauses containing both answer and non-answer literals. Note that their work encompassed clauses containing only answer literals.

Our complete characterization of relevant resolvents as answers, and the partition of that set into specific, generic, and hypothetical answer clauses, closes the chapter on question answering in a resolution theorem prover. We have provided a semantics for every resolvent reachable using a set-of-support strategy. Our work builds the foundation necessary for those interested in theorem proving to understand failed proofs. We provide a clear interpretation for failed branches of a proof tree.

We have shown that our answer typology expands upon previously defined classes of answers, in particular developing the largely ignored class of hypothetical answers. In addition, we have delineated the class of underconstrained generic answers as open answer forms that are associated with proof. While these forms had been recognized as answers in the past, their role and relationship to other generic answers was not well defined. Finally, we presented a discussion of disjunctive answers that clarifies the difference between the inner structure of the set of answer literals, comprising the question component of an answer, and the set of all of the literals in a clause taken as a whole. The answer literals are distinguished as those predicates of interest to the questioner. Answer literals may be partitioned into groups, where not only variable sharing is important, but sharing of constants is also of interest. We described a canonical form for answer clauses that is applicable outside the context of reasoning. This form is used in order to prevent distinct answer clauses that map onto logically equivalent answers from being incorporated into the answer set. The canonical form reflects a bias towards maximizing the number of answer literals in a clause.

Our formal characterization of all answers in a resolution theorem prover demonstrates the relationships between the different answer classes and the form of answers. Now that we have provided a complete characterization of answers as relevant information, we examine the problem of how to determine when a clause is *informative*.

Chapter 5

Answer Sets

5.1 Introduction

In the previous chapter, we identified *answer clauses* as those clauses *relevant* to a question. Relevance of a clause to a question is based on both the question and the knowledge base with respect to which the question is answered. It is clear that relevant clauses may not all be equally good answers: some may be more *informative* than others. For example, the clause:

$\{(ANSWER\ (\mathcal{LIKES}\ FRED\ \ WILMA))\}$

corresponding to the specific answer *Fred likes Wilma*, is more informative than:

$\{(ANSWER\ (\mathcal{LIKES}\ FRED\ \ WILMA)),\ (ANSWER\ (\mathcal{LIKES}\ FRED\ \ BETTY))\}$

corresponding to the specific answer *Fred likes Wilma or Fred likes Betty*. In this case the relative informativeness of the two clauses is based on the syntactic subsumption relation that holds between them: the first subsumes the second.

This chapter addresses the problem of organizing answer clauses based on their *informativeness*. The more informative a clause is, the better the answer it represents. We define informativeness with respect to a *set of answer clauses*, taking into account the answer class to which an answer clause belongs (specific, generic, or hypothetical). While all answer clauses are relevant, they are not necessarily *informative*.

The function of an answering agent is to provide the *most informative answers* available. According to our theory, finding answers can be viewed as the process whereby an answering agent discovers and keeps track of answers based on their informativeness. The "keeping track" as we define it involves maintaining an *answer set* whose members are answer clauses. At any point during the search for answers, an answering agent may be called upon to *produce* an answer. In order to provide the best answer possible, the answer set should contain the most informative answer clauses that are *currently* available. Thus, an answer set in a question answering scenario is temporally related to the reasoning process. In addition to the current answer set, we introduce the notion of the *complete answer set* for a given question and knowledge base. This potentially infinite set is a subset of the complete set of answer clauses described in the previous chapter. Regardless of whether the complete answer set is finite or infinite, there is neither a requirement nor a guarantee that a particular answering agent will actually generate this set. What is important is that, at any point during the reasoning process, the answering agent is able to return the most informative answers at that point in time.

It is important to distinguish between our notion of *answer set* and the term *answer set* introduced by Gelfond and Lifschitz [1991] in the context of logic programming. Answer set semantics extends the notion of stable model semantics [Gelfond and Lifschitz, 1988] to apply to extended logic programs, namely, those containing classical negation. Briefly, a stable model of a logic program is a minimal Herbrand model for the program. A program with a unique stable model is said to have a stable model semantics. The interest in stable model semantics and answer set semantics has to do with finding an answer to a ground query, which, obviously, has either a yes or a no answer. Given a logic program with a stable model, the answer to a query is yes if it belongs to the stable model; otherwise, the answer is no. Note that negation by failure is assumed. An answer set (as defined by Gelfond and Lifschitz) is analogous to a stable model, only if it models a program that may contain explicit, or classical, negation. Thus, there are three possible answers to a ground query with respect to answer set semantics: yes, if the query is a member of the answer set; no, if the negation of the query is a member of the answer set; and unknown otherwise. Answer set programming has emerged as an important topic in AI, and was the focus of a AAAI Spring 2001 Symposium entitled *Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*. This is an entirely separate endeavor from our work: we are not interested in Herbrand models for logic programs; rather, we have a particular interest in the semantics of non-ground questions and answers.

5.2 Definitions and Terminology

This section serves to introduce the basic definitions and terminology that will be employed in this chapter. Syntactic subsumption, defined in the previous chapter, forms the basis for determining when an answer clause is considered to be *informative*. Recall that a clause A syntactically subsumes a clause B if there is a substitution σ such that the literals in $A\sigma$ are a subset of the literals in B. We represent that clause c_i syntactically subsumes clause c_j as $\mathcal{SUBS}(c_i,\ c_j)$. We extend this definition to subsumption between a clause and a set of clauses.

Definition 5.1 Syntactic subsumption of a clause by a set of clauses. *A set of clauses $\mathcal{S}$ syntactically subsumes a clause c, written $\mathcal{SUBS}(\mathcal{S},\ c)$, if, for some clause $c_i \in \mathcal{S}$, $\mathcal{SUBS}(c_i,\ c)$.*

Syntactic subsumption is an appropriate measure of informativeness of a clause with respect to a set of clauses. However, it is important to carefully circumscribe the set of clauses with respect to which subsumption is computed. If subsumption were computed with respect to all answer clauses, generic answers could be subsumed by specific answers. This would contradict our contention that answers at different levels of generality are *all* interesting. What is needed is a method that employs syntactic subsumption yet preserves answers at different levels of generality. We accomplish this by partitioning the set with respect to which answer clauses are judged informative, and selecting appropriate partitions of that set with respect to which syntactic subsumption is determined.

We assume that all answer clauses are converted into the canonical form described in the previous chapter with respect to disjunctive answer.

As previously described, a set of answer clauses may be partitioned according to the classes to which its constituent answer clauses belong. We use the following terms to refer to the partitions of $\mathcal{S}$, where $\mathcal{S}$ is any set of answer clauses.

- $\mathcal{S}_s$ – the specific answer clauses in $\mathcal{S}$
- $\mathcal{S}_g$ – the generic answer clauses in $\mathcal{S}$
- $\mathcal{S}_h$ – the hypothetical answer clauses in $\mathcal{S}$
- $\mathcal{S}_{h_s}$ – the hypothetical specific answer clauses in $\mathcal{S}$
- $\mathcal{S}_{h_g}$ – the hypothetical generic answer clauses in $\mathcal{S}$

The following relations hold between the answer set partitions:

- $\mathcal{S}_s \cup \mathcal{S}_g \cup \mathcal{S}_h = \mathcal{S}$
- $\mathcal{S}_{h_s} \cup \mathcal{S}_{h_g} = \mathcal{S}_h$

5.2.1 Informativeness of Answer Clauses

We are now in a position to define what it means for an answer clause to be considered *informative* with respect to a set of answer clauses. Specifically, we have the following:

Definition 5.2 Informative Specific Answer Clause. *A specific answer clause c_s is informative with respect to a set of answer clauses $\mathcal{S}$ if* $\neg(\mathcal{SUBS}(\mathcal{S}_s,\ c_s))$.

Informativeness for specific answer clauses is determined solely with respect to the specific answer clauses in a set. A specific answer clause is informative with respect to a set of answer clauses as long as it is not subsumed by a specific answer clause in that set.

Definition 5.3 Informative Generic Answer Clause. *A generic answer clause c_g is informative with respect to a set of answer clauses $\mathcal{S}$ if* $\neg(\mathcal{SUBS}(\mathcal{S}_g,\ c_g))$.

Informativeness for generic answer clauses is determined by checking subsumption solely with the generic answer clauses in a set, which obviates the problem of subsumption of generic answer clauses by specific answer clauses.

Hypothetical answer clauses may be subsumed by other hypothetical answer clauses or by specific or generic answer clauses. In the same way that specific answer clauses do not preclude the informativeness of generic answer clauses, the partition of the set with respect to which subsumption is computed for hypothetical answer clauses is based both on their subtype (specific or generic) as well as the fact that they are hypothetical. Hypothetical specific answer clauses are checked for subsumption by hypothetical specific answer clauses and (non-hypothetical) specific answer clauses.

Definition 5.4 Informative Hypothetical Specific Answer Clause. *A hypothetical specific answer clause c_{h_s} is informative with respect to a set of answer clauses $\mathcal{S}$ if* $\neg(\mathcal{SUBS}((\mathcal{S}_s \cup \mathcal{S}_{h_s}),\ c_{h_s}))$.

Subsumption relations for hypothetical generic answer clauses are computed with respect to hypothetical generic answer clauses and generic answer clauses.

Definition 5.5 Informative Hypothetical Generic Answer Clause. *A hypothetical generic answer clause c_{h_g} is informative with respect to a set of answer clauses $\mathcal{S}$ if* $\neg(\mathcal{SUBS}((\mathcal{S}_g \cup \mathcal{S}_{h_g}), c_{h_g}))$.

Combining the previous two definitions, a hypothetical answer clause is informative with respect to a set of answer clauses if it is either an informative hypothetical specific answer clause or an informative hypothetical generic answer clause.

Definition 5.6 Informative Hypothetical Answer Clause. *A hypothetical answer clause c_h is informative with respect to a set of answer clauses $\mathcal{S}$ if it is either an informative hypothetical specific answer clause with respect to $\mathcal{S}$ or an informative hypothetical generic answer clause with respect to $\mathcal{S}$.*

Now that informativeness for each type of answer clause has been defined, we can define what it means *in general* for an answer clause to be informative.

Definition 5.7 Informative Answer Clause. *An answer clause c is an informative answer clause with respect to a set of answer clauses $\mathcal{S}$, written $\mathcal{INF}(c, \mathcal{S})$, if c is either an informative specific answer clause, an informative generic answer clause, or an informative hypothetical answer clause with respect to $\mathcal{S}$.*

The following set of answer clauses will be used to illustrate the concept of informativeness of an answer clause.
Let $\mathcal{S}$ comprise the following clauses, with glosses given below each clause:

1. $\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$
   Rover barks or fido barks.

2. $\{$(ANSWER (barks *x*)), $\neg$(dog *x*)$\}$
   All dogs bark.

3. $\{$(ANSWER (barks monty)), $\neg$(barks jake)$\}$
   If jake barks then monty barks.

4. $\{$(ANSWER (barks *x*)), $\neg$(cat *x*), $\neg$(barks fluffy)$\}$
   If fluffy barks then all cats bark.

The specific answer clause $\{$(ANSWER (barks rover))$\}$ is informative with respect to $\mathcal{S}$ because it subsumes clause 1. The clause $\{$(ANSWER (barks rover)), (ANSWER (barks fido)), (ANSWER (barks fifi))$\}$ is not informative with respect to $\mathcal{S}$ because it is subsumed *by* clause 1.

The generic answer clause $\{$(ANSWER (barks *x*)), ¬(dog *x*), ¬(cat *x*)$\}$ is not informative with respect to $\mathcal{S}$ because it is subsumed by clause 2. However, $\{$(ANSWER (barks *x*)), ¬(seal *x*)$\}$ is informative with respect to $\mathcal{S}$.

The hypothetical specific answer clause $\{$(ANSWER (barks tootles)), ¬(dog tootles)$\}$ is informative with respect to $\mathcal{S}$ while $\{$(ANSWER (barks monty)), (ANSWER (bites monty)), ¬(barks jake)$\}$ is not informative with respect to $\mathcal{S}$ because it is subsumed by clause 3.

Finally, the hypothetical generic answer clause $\{$(ANSWER (barks *x*)), ¬(barks fluffy)$\}$ is informative with respect to $\mathcal{S}$, whereas $\{$(ANSWER (barks *x*)), (ANSWER (bites *x*)), ¬(cat *x*), ¬(barks fluffy)$\}$ is not informative with respect to $\mathcal{S}$ because it is subsumed by clause 4.

A useful concept related to informativeness of an answer clause with respect to a set of answer clauses is that of a clause that is an *informative member* of a set of answer clauses. While the notion of an answer clause being informative *relative* to a set of answer clauses captures the fact that it conveys information not currently in that set, the notion of an answer clause as an informative *member* of a set of answer clauses indicates that the set would be *less informative* without that particular answer clause. We propose the following definition:

Definition 5.8 Informative Member. *An answer clause c is an informative member of a set of answer clauses $\mathcal{S}$, written $\mathcal{INFM}(c, \mathcal{S})$, if $\mathcal{INF}(c, (\mathcal{S} \setminus \{c\}))$.*

In general, a set of answer clauses may contain some informative members and some uninformative members.

5.3 Answer Sets

We are now ready to formally define what it means for a set of answer clauses to be considered an *answer set*. What distinguishes an *answer set* from sets of answer clauses in general is the fact that all members of an answer set are *informative members*, formally expressed as follows:

Definition 5.9 Answer Set. *An answer set $\mathcal{S}$ is a set of answer clauses c_i such that*
$\forall c_i \in \mathcal{S}, \mathcal{INFM}(c_i, \mathcal{S})$.

It is useful to define an operation on sets of answer clauses that will turn them into answer sets, namely, an operation to remove all but the informative members of the set. We term this operation $\mathcal{CLEAN}$, where it is defined as follows:

Definition 5.10 $\mathcal{CLEAN}$**.** *For a set $\mathcal{S}$ of answer clauses, $\mathcal{CLEAN}(\mathcal{S}) = \mathcal{S}'$ where $\mathcal{S}'$ is the set of the informative members of $\mathcal{S}$.*

The set $\mathcal{S}$ = {{(ANSWER (barks rover)), (ANSWER (barks fido))}, {(ANSWER (barks x)), ¬(dog x)}, {(ANSWER (barks rover))} is not an answer set: {(ANSWER (barks rover)), (ANSWER (barks fido))} is not an informative member of $\mathcal{S}$. $\mathcal{CLEAN}(\mathcal{S})$ = {{(ANSWER (barks rover))}, {(ANSWER (barks x)), ¬(dog x)}} is an answer set.

Answer sets may be created and updated using the following two operations:

- Create($\mathcal{A}$): $\mathcal{A} = \emptyset$

  This is a straightforward operation whereby a new object named $\mathcal{A}$ of type answer set is created and initialized to the empty set. An empty answer set indicates the absence of answer clauses.

- Update($\mathcal{A}$, c): $\mathcal{A} \times c \longrightarrow \mathcal{A}'$, where c is an answer clause, $\mathcal{A}$ and $\mathcal{A}'$ are answer sets, and $\mathcal{A}' = \mathcal{CLEAN}(\mathcal{A} \cup \{c\})$

  Note that if c is not informative with respect to $\mathcal{A}$ then Update($\mathcal{A}$, c) = $\mathcal{A}$

Two clauses are considered equivalent if each subsumes the other. When computing with answer sets it is important to recognize equivalent clauses. When calculating the intersection of two answer sets, one of each pair of equivalent clauses across the two sets must appear in the intersection. Given the answer sets $\mathcal{S}$ = {{(ANSWER (barks x))}} and $\mathcal{T}$ = {{(ANSWER (barks y))}}, their intersection contains a single clause. Depending on the direction in which subsumption is calculated, we have either $\mathcal{S} \cap \mathcal{T}$ = {{(ANSWER (barks x))}} or $\mathcal{S} \cap \mathcal{T}$ = {{(ANSWER (barks y))}}. There is no difference between the resulting sets other than the variable name.

While answer sets are clearly closed under intersection and set minus, the union of two answer sets is not necessarily an answer set. For example, if $\mathcal{S}$ = {{(ANSWER (barks rover)), (ANSWER (barks fido))}, {(ANSWER (barks x)), ¬(dog x)}}, and $\mathcal{T}$ = {{(ANSWER (barks rover))}, {(ANSWER (barks y)), ¬(dog y)}}, $\mathcal{S}$ and $\mathcal{T}$ are answer sets, but $\mathcal{S} \cup \mathcal{T}$ is not an answer set because $\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$ is a non-informative member: it is subsumed by $\{$(ANSWER (barks rover))$\}$. The appropriate way to achieve something akin to the union of two answer sets is to perform repeated updates of one set with the clauses in the other. While all of the clauses in the resultant set will have been members of at least one of the two starting sets, the resulting set may have neither of the starting sets as a subset.

It is useful to consider whether two answer sets are *equivalent*, and whether one answer set is *more informative* than another. It is clear that the relative informativeness of answer sets can not always be determined. For example, is $\{\{$(ANSWER (barks rover))$\}\}$ more informative than $\{\{$(ANSWER (barks fido))$\}\}$? Or are they *equally informative*? By our definition we shall say that the relationship between these answer sets can not be determined.

Equivalence of answer sets will hold under the following circumstances:

Definition 5.11 Equivalence of Answer Sets. *Two answer sets are equivalent if their clauses are equivalent.*

In other words, $\{\{$(ANSWER (barks x)), $\neg$(dog x)$\}\}$ and $\{\{$(ANSWER (barks y)), $\neg$(dog y)$\}\}$ are equivalent answer sets, because they differ only in the names of the variables found in their constituent clauses. In both cases, the answer set comprises a single answer clause corresponding to the generic answer *all dogs bark*. On the other hand, the answer sets $\{\{$(ANSWER (barks fido))$\}\}$ and $\{\{$(ANSWER (barks x))$\}\}$ are *not* equivalent; neither are $\{\{$(ANSWER (barks fido))$\}\}$ and $\{\{$(ANSWER (barks rover))$\}\}$. When two answer sets are equivalent, we say that they are *as informative as one another*.

We propose an informativeness relation that defines a partial ordering on answer sets: for many answer sets, the informativeness relation cannot be computed.

Definition 5.12 Informativeness Relation. *An answer set $\mathcal{S}$ is more informative than an answer set $\mathcal{T}$ if the following hold: for all clauses $c_s \in (\mathcal{S} \setminus \mathcal{T})\}$, it is the case that $\mathcal{INF}(c_s, \mathcal{T})$, and, for all clauses $c_t \in (\mathcal{T} \setminus \mathcal{S})$, it is that case that $\neg\mathcal{INF}(c_t, \mathcal{S})$.*

Informally, an answer set $\mathcal{S}$ is more informative than an answer set $\mathcal{T}$ if all of the clauses that appear in $\mathcal{S}$ and *not* in $\mathcal{T}$ are informative relative to $\mathcal{T}$, and if none of the clauses that appear in $\mathcal{T}$ and not in $\mathcal{S}$ are informative with respect to $\mathcal{S}$. If an answer set $\mathcal{S}$ is either equivalent to an answer set $\mathcal{T}$ or is more informative than $\mathcal{T}$, then we say that $\mathcal{S}$ *is at least as informative as* $\mathcal{T}$.

Definition 5.13 Informativeness of the Empty Set. *The empty set is uninformative with respect to all answer sets.*

The empty set is a set containing no clauses, therefore there are no clauses with which to compare informativeness to another set. Clearly, any clause is informative with respect to the empty set.

The following examples demonstrate the informativeness relation between answer sets.

- If $\mathcal{S} = \{\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$, $\{$(ANSWER (barks x)), $\neg$(dog x)$\}\}$, and $\mathcal{T} = \{\{$(ANSWER (barks rover))$\}$, $\{$(ANSWER (barks y)), $\neg$(dog y)$\}\}$, $\mathcal{T}$ is more informative than $\mathcal{S}$. The only clause in $\mathcal{S} \setminus \mathcal{T}$ is $\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$, which is not informative with respect to $\mathcal{T}$: it is subsumed by the clause $\{$(ANSWER (barks rover))$\}$. The sole clause in $\mathcal{T} \setminus \mathcal{S}$ is $\{$(ANSWER (barks rover))$\}$, which is informative with respect to $\mathcal{S}$.

- If $\mathcal{S} = \{\{$(ANSWER (barks rover))$\}\}$, and $\mathcal{T} = \{\{$(ANSWER (barks fido))$\}\}$, no informativeness relation holds between $\mathcal{S}$ and $\mathcal{T}$. The only clause in $\mathcal{S} \setminus \mathcal{T}$ is $\{$(ANSWER (barks rover))$\}$, which is informative relative to $\mathcal{T}$. Similarly, the only clause in $\mathcal{T} \setminus \mathcal{S}$ is $\{$(ANSWER (barks fido))$\}$, which is informative with respect to $\mathcal{S}$. Since both sets contain clauses that are informative relative to each other, one can not be said to be more informative than the other.

- If $\mathcal{S} = \{\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$, $\{$(ANSWER (barks fifi))$\}\}$, and $\mathcal{T} = \{\{$(ANSWER (barks fido))$\}$, $\{$(ANSWER (barks rover)), (ANSWER (barks fifi))$\}\}$, each contains a clause not in their intersection that is informative with respect to the other. In addition, each contains a clause not in their intersection that is not informative with respect to the other. Considering $\mathcal{S}$ relative to $\mathcal{T}$, $\{$(ANSWER (barks rover)), (ANSWER (barks fido))$\}$ is uninformative and $\{$(ANSWER (barks fifi))$\}$ is informative. Considering $\mathcal{T}$ with respect to $\mathcal{S}$,$\{$(ANSWER (barks fido))$\}$ is informative and $\{$(ANSWER (barks rover)), (ANSWER (barks fifi))$\}$ is not informative. Therefore neither $\mathcal{S}$ nor $\mathcal{T}$ can be said to be more informative than the other.

When an update is performed on an answer set, the resultant is at least as informative as the initial answer set. The update function is nondecreasing, monotonic in terms of the informativeness of the resulting answer set.

Theorem 5.1 (Informativeness of an Updated Answer Set) *An answer set $\mathcal{A}'$ that results from performing an Update operation on an answer set $\mathcal{A}$ is at least as informative as $\mathcal{A}$.*

Proof: There are two possibilities when an update is performed on an answer set $\mathcal{A}$ given clause c: c is informative with respect to $\mathcal{A}$ or it isn't.

1. $\neg\mathcal{INF}(c, \mathcal{A})$

   If c is not informative with respect to $\mathcal{A}$, then by definition Update($\mathcal{A}$, c) = $\mathcal{A}$. Since $\mathcal{A}' = \text{Update}(\mathcal{A}, c)$, we have $\mathcal{A}' = \mathcal{A}$. Because these two sets are equivalent, $\mathcal{A}'$ is *at least as informative* as $\mathcal{A}$.

2. $\mathcal{INF}(c, \mathcal{A})$

   There are two possibilities:

   (a) $\mathcal{A}'$ = Update($\mathcal{A}$, c) = $\mathcal{A} \cup \{c\}$

   In this case none of the clauses from $\mathcal{A}$ are removed as a result of performing the $\mathcal{CLEAN}$ operation on $\mathcal{A} \cup \{c\}$. According to the definition of the informativeness relation, $\mathcal{A}'$ is more informative than $\mathcal{A}$ if the following two conditions hold:

   i. All clauses in $\mathcal{A}' \setminus \mathcal{A}$ are informative with respect to $\mathcal{A}$. This is obvious: $\mathcal{A}' \setminus \mathcal{A} = \{c\}$, and our assumption is that $\mathcal{INF}(c, \mathcal{A})$.

   ii. All clauses in $\mathcal{A} \setminus \mathcal{A}'$ are not informative with respect to $\mathcal{A}'$. Since all clauses in $\mathcal{A}$ are contained in $\mathcal{A}'$, $\mathcal{A} \setminus \mathcal{A}' = \emptyset$. The empty set is uninformative with respect to all sets, thus this condition is satisfied.

   Since both conditions are satisfied, $\mathcal{A}'$ is more informative than $\mathcal{A}$.

   (b) $\mathcal{A}'$ = Update($\mathcal{A}$, c) $\neq \mathcal{A} \cup \{c\}$

   In this case some of the clauses from $\mathcal{A}$ are not present in $\mathcal{CLEAN}(\mathcal{A} \cup \{c\})$. Let $\mathcal{S}_c$ be the set of clauses c_i such that $c_i \in \mathcal{A}$ and $c_i \notin \mathcal{CLEAN}(\mathcal{A} \cup \{c\})$. According to the definition of the informativeness relation, $\mathcal{A}'$ is more informative than $\mathcal{A}$ if the following two conditions hold:

   i. All clauses in $\mathcal{A}' \setminus \mathcal{A}$ are informative with respect to $\mathcal{A}$. This is obvious: $\mathcal{A}' \setminus \mathcal{A} = \{c\}$, and our assumption is that $\mathcal{INF}(c, \mathcal{A})$.

ii. All clauses in $\mathcal{A} \setminus \mathcal{A}'$ are not informative with respect to $\mathcal{A}'$. We stated above that this set difference is nonempty, namely, $\mathcal{A} \setminus \mathcal{A}' = \mathcal{S}_c$. Assume that some clause $c_i \in \mathcal{S}_c$ is informative with respect to $\mathcal{A}'$. By definition, $\mathcal{A}' = \mathcal{CLEAN}(\mathcal{A} \cup \{c\})$, which includes all of the informative members of $\mathcal{A} \cup \{c\}$. But $c_i \notin \mathcal{CLEAN}(\mathcal{A} \cup \{c\})$, thus c_i is not informative with respect to $\mathcal{A}'$, contradicting our assumption. Therefore, there is no clause $c_i \in \mathcal{S}_c$ that is informative with respect to $\mathcal{A}'$, and this condition is satisfied.

Since both conditions are satisfied, $\mathcal{A}'$ is more informative than $\mathcal{A}$. □

Theorem 5.2 (Answer sets and their finite subsets) *If all of the finite subsets of a given set $\mathcal{S}$ of answer clauses are answer sets, then $\mathcal{S}$, which is the union of all of these finite subsets, is itself an answer set. Conversely, given an answer set $\mathcal{S}$, it is the case that all finite subsets of $\mathcal{S}$ are answer sets.*

Proof: By contradiction: Assume that all finite subsets of a set $\mathcal{S}$ of a answer clauses are answer sets, but that their union is not. Thus, there must be at least one pair of clauses $c_i,\ c_j \in \mathcal{S}$ such that either c_i is not informative with respect to $\{c_j\}$ or vice versa. Assume the former case without loss of generality. $\mathcal{S}_i = \{c_i,\ c_j\}$ is a finite subset of $\mathcal{S}$. According to our assumptions, c_i is not an informative member of $\mathcal{S}_i$, therefore $\mathcal{S}_i$ is not an answer set, and this contradicts the assumption that all finite subsets of $\mathcal{S}$ are answer sets.
For the converse, assume that $\mathcal{S}$ is an answer set and that there is at least one finite subset $\mathcal{S}_i \subseteq \mathcal{S}$ that is not an answer set. This means that there are at least two clauses $c_i,\ c_j \in \mathcal{S}_i$ such that either c_i is not informative with respect to $\{c_j\}$ or vice versa. Assume the former case without loss of generality. These clauses are both members of $\mathcal{S}$, thus, c_i is not an informative member of $\mathcal{S}$, which means that $\mathcal{S}$ is not an answer set. But this contradicts our original assumption that $\mathcal{S}$ is an answer set. □

The point made here is that, once you have an answer set, you are guaranteed properties of its subsets and their unions. However, if you simply take the union of two arbitrary answer sets, there is *no guarantee* that the result will itself be an answer set.

We usually discuss answer sets with respect to a particular question $\mathcal{Q}$, a knowledge base $\mathcal{KB}$, and a control structure $\mathcal{CS}$. The set of possible answers is determined by $\mathcal{Q}$ and $\mathcal{KB}$, while the control structure determines the order in which answer clauses will be generated. We make the following assumptions about the control structure:

- Reasoning proceeds as it does in a resolution theorem prover (as described in the previous chapter).
- Set-of-support or an analogous strategy is employed so that all resolvents generated in the course of reasoning are answer clauses, thus are relevant to the question.

In this context, we refer to the i^{th} answer clause and the i^{th} answer set, which contains the informative subset of those answers numbered one through i. Without loss of generality, we assume that each reasoning step is associated with the generation of a single answer. An answer *clause* that is derived with respect to a question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$ will be denoted $a_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}$, where i is the index indicating the order in which the answer was derived. An answer *set* with respect to a question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$ will be denoted $\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}$, where i is an index indicating that the answer set contains the informative answers from among the first i answers generated . The following is an inductive definition of an answer set with respect to a question, knowledge base, and control structure.

Definition 5.14 Answer set with respect to question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$.

- $\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},0]} = \emptyset$
- $\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]} = Update(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i-1]},\ a_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}),\ \forall\ i > 0.$

We now define the $\mathcal{GENANS}$ function that takes as input the i^{th} answer set and and outputs the $i+1^{st}$.

Definition 5.15 The $\mathcal{GENANS}$ function. *$\mathcal{GENANS}$ is defined with respect to question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$ as follows:*

$\mathcal{GENANS}(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}) = \mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i+1]}$

We are finally in a position to define an algorithm to find answers, thereby building a sequence of answer sets.

Algorithm: Find Answers:

Given a question $\mathcal{Q}$, a knowledge base $\mathcal{KB}$, and a control structure $\mathcal{CS}$, proceed as follows:

$$\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},0]} = \emptyset$$
$$\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},1]} = \mathcal{GENANS}(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},0]})$$
$$\vdots$$
$$\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},n+1]} = \mathcal{GENANS}(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},n]})$$

When there are an infinite number of reasoning steps, this is an infinite sequence. The answer set containing all of the informative answers, whether finite or infinite, is termed the *complete answer set*. In a reasoning agent the number of iterations of $\mathcal{GENANS}$ is resource limited, and the fixed point may not be found. We term the fixed point of the $\mathcal{GENANS}$ function the *complete* answer set for a given question and knowledge base:

Definition 5.16 Complete Answer Set. *The complete answer set for a question $\mathcal{Q}$ and a knowledge base $\mathcal{KB}$ is the fixed point of the $\mathcal{GENANS}$ function, where $\mathcal{GENANS}$ is associated with any appropriate control structure for reasoning. We denote the complete answer set $\mathcal{A}_{[\mathcal{Q},\mathcal{KB}]}$.*

There is no guarantee that there are a finite number of informative answers, in which case no answering agent will generate the complete answer set. Even when the complete answer set is finite, its generation is not the goal of question answering as we envision it. Rather, the fact that at any point in time during the answering process there is a set of answers from which to choose is what we consider the most important feature of our formalism.

A notable feature of the way in which we have defined answer sets is that the clauses in *an answer set* for a given question and knowledge base may not be in the *complete answer set* for the same question and knowledge base: there may be other answer clauses that subsume them that have not *yet* been found by the answering agent as it reasons. This underscores the temporal nature of answer sets and the answering process. An answer set contains *the most informative answers discovered thus far*, not the most informative answers that might possibly be found in a complete search (in case the search space is finite). This accords not only with the way an answering agent reasons in search of answers, but also with the way people answer questions. How many times have you thought how much better you might have answered a question if you had been given more time? Regardless of whether or not answers *improve* over time, we have shown that lengthening the time alloted for answering a question never results in *less* informative answers. It is this characteristic of question answering that makes it an anytime reasoning process.

The following example demonstrates the way in which sequential answer sets are generated using the **Find Answers** algorithm. Consider the following list of the clauses that comprise the knowledge base of an answering agent:

1. {(CALICO FLUFFY)}

   Fluffy is a calico.

2. {(LIKES FLUFFY ROVER)}

   Fluffy likes Rover.

3. {(¬(CALICO ?20)), (CAT ?20)}

   Calicos are cats.

4. {(¬(CAT ?6)), (¬(FISH ?8)), (LIKES ?6 ?8)}

   Cats like fish.

5. {(¬(CAT ?14)), (¬(MOUSE ?16)), (LIKES ?14 ?16)}

   Cats like mice.

   A question is posed to the agent. The question clause with accompanying gloss is shown below:

6. {(¬(LIKES FLUFFY ?50)), (ANSWER (LIKES FLUFFY ?50))}

   What does fluffy like?

One possible sequence of answer sets generated by the **Find Answers** algorithm is listed below along with explanatory text. For simplicity we drop the subscript letters used to designate the question, knowledge base, and control structure with respect to which the answer sets are computed: we have listed the knowledge base and question above, and the control structure is implicit in the answer set sequence below. Clause numbers are noted in parentheses next to each clause below, and new clauses are numbered beginning with 7 following along from the knowledge base and question clauses.

- $\mathcal{A}_0 = \emptyset$

  The initial answer set is the empty set.

- $\mathcal{A}_1 = \{\{$(ANSWER (LIKES FLUFFY ROVER))$\}$ (7)$\}$

  The specific answer clause $\{$(ANSWER (LIKES FLUFFY ROVER))$\}$ has been generated in step 1 by resolving clauses 2 and 6.

- $\mathcal{A}_2 = \{\{$(ANSWER (LIKES FLUFFY ROVER))$\}$ (7),

  $\{$(ANSWER (LIKES FLUFFY ?171)), (¬(CAT FLUFFY)), (¬(FISH ?171))$\}$ (8)$\}$

  Step 2 of reasoning produces the hypothetical generic answer clause $\{$(ANSWER (LIKES FLUFFY ?171)), (¬(CAT FLUFFY)), (¬(FISH ?171))$\}$, with hypothetical component (¬(CAT FLUFFY)). This clause is produced by resolving clauses 4 and 6. By examining the knowledge base we can see that fluffy is a cat, which means that eventually this hypothetical generic answer will be subsumed by a generic answer: the hypothetical component will be "settled". At this point in time, however, if a questioner wanted *all answers* the answering agent would have to provide the following answers (shown in English rather than logic form): *fluffy likes rover* and *if fluffy is a cat then fluffy likes fish*.

- $\mathcal{A}_3 = \{\{$(ANSWER (LIKES FLUFFY ROVER))$\}$ (7),

  $\{$(ANSWER (LIKES FLUFFY ?171)), (¬(CAT FLUFFY)), (¬(FISH ?171))$\}$ (8),

  $\{$(ANSWER (LIKES FLUFFY ?186)), (¬(CAT FLUFFY)), (¬(MOUSE ?186))$\}$ (9)$\}$

  Step 3 of reasoning produces another hypothetical generic answer clause, $\{$(ANSWER (LIKES FLUFFY ?186)), (¬(CAT FLUFFY)), (¬(MOUSE ?186))$\}$, by resolving clauses 5 and 6. The hypothetical component of this answer is the literal (¬(CAT FLUFFY)). For the reasons mentioned above, this hypothetical generic answer will eventually be subsumed by a generic answer. At this point in time, however, if the questioner wanted *all answers* the answering agent would have to provide the following three answers: *fluffy likes rover*; *if fluffy is a cat then fluffy likes fish*; and *if fluffy is a cat then fluffy likes mice*.

- $\mathcal{A}_4 = \{\{$(ANSWER (LIKES FLUFFY ROVER))$\}$ (7),

  $\{$(ANSWER (LIKES FLUFFY ?171)), (¬(CAT FLUFFY)), (¬(FISH ?171))$\}$ (8),

  $\{$(ANSWER (LIKES FLUFFY ?186)), (¬(CAT FLUFFY)), (¬(MOUSE ?186))$\}$ (9),

  $\{$(ANSWER (LIKES FLUFFY ?614)), (¬(CALICO FLUFFY)), (¬(FISH ?614))$\}$ (10)$\}$

  In step 4 an additional hypothetical generic answer clause is added when clauses 3 and 8 are resolved. While clause 10 *taxonomically* subsumes clause 8, recall that we only take into account syntactic subsumption with respect, in this case, to hypothetical generic and generic answer clauses. The answers now available are: *fluffy likes rover*; *if fluffy is a cat then fluffy likes fish*; *if fluffy is a cat then fluffy likes mice*; *if fluffy is a calico then fluffy likes fish*.

- $\mathcal{A}_5$ = {{(ANSWER (LIKES FLUFFY ROVER))} (7),
  {(ANSWER (LIKES FLUFFY ?171)), (¬(CAT FLUFFY)), (¬(FISH ?171))} (8),
  {(ANSWER (LIKES FLUFFY ?186)), (¬(CAT FLUFFY)), (¬(MOUSE ?186))} (9),
  {(ANSWER (LIKES FLUFFY ?614)), (¬(CALICO FLUFFY)), (¬(FISH ?614))} (10),
  {(ANSWER (LIKES FLUFFY ?421)), (¬(MOUSE ?421)), (¬(CALICO FLUFFY))} (11)}
  Step 5, in a manner analogous to step 4, adds a hypothetical generic answer clause that is *taxonomically* more specific than a clause currently in the answer set: in this case, clause 11 is more specific than clause 9. {(ANSWER (LIKES FLUFFY ?421)), (¬(MOUSE ?421)), (¬(CALICO FLUFFY))} is generated when clause 3 is resolved with clause 9. The answers currently available are: *fluffy likes rover*; *if fluffy is a cat then fluffy likes fish*; *if fluffy is a cat then fluffy likes mice*; *if fluffy is a calico then fluffy likes fish*; *if fluffy is a calico then fluffy likes mice*.

- $\mathcal{A}_6$ = {{(ANSWER (LIKES FLUFFY ROVER))} (7),
  {(ANSWER (LIKES FLUFFY ?186)), (¬(CAT FLUFFY)), (¬(MOUSE ?186))} (9),
  {(ANSWER (LIKES FLUFFY ?421)), (¬(MOUSE ?421)), (¬(CALICO FLUFFY))} (11),
  {(ANSWER (LIKES FLUFFY ?1086)), (¬(FISH ?1086))} (12)}
  In step 6 the generic answer clause {(ANSWER (LIKES FLUFFY ?1086)), (¬(FISH ?1086))} is generated when clauses 1 and 10 are resolved. This clause syntactically subsumes clauses 8 and 10, so they are removed as a result of the $\mathcal{CLEAN}$ operation. Clauses 8 and 10 are hypothetical generic answer clauses whose generic answer components are equivalent to clause 12. The current set of answers comprises the following: *fluffy likes rover*; *if fluffy is a cat then fluffy likes mice*; *if fluffy is a calico then fluffy likes mice*; *fluffy likes fish*.

- $\mathcal{A}_7$ = {{(ANSWER (LIKES FLUFFY ROVER))} (7),
  {(ANSWER (LIKES FLUFFY ?1086)), (¬(FISH ?1086))} (12),
  {(ANSWER (LIKES FLUFFY ?845)), (¬(MOUSE ?845))} (13)}
  In step 7, the final reasoning step, the generic answer clause {(ANSWER (LIKES FLUFFY

?845)), (¬(MOUSE ?845))} is added to the answer set. This clause results when clauses 1 and 11 are resolved. In a manner analogous to the update of the answer set described in step 6, this new generic answer clause syntactically subsumes the remaining two hypothetical generic answer clauses, numbered 9 and 11. These clauses are thus removed as a result of the $\mathcal{CLEAN}$ operation performed on $\mathcal{A}_6 \cup$ {(ANSWER (LIKES FLUFFY ?845)), (¬(MOUSE ?845))}. The final, and *complete* answer set for the given question and knowledge base contains three answers whose English glosses are *fluffy likes rover*, *fluffy likes fish*, and *fluffy likes mice*.

5.4 The "Discharging Hypotheticals" Search Strategy for Answer Set Generation

Different search strategies ultimately lead to the same complete answer set, but might yield different intermediate answer sets. In a top-down search, the answers first found will be those that can immediately be resolved with a question clause. Such answers are likely to be hypothetical, requiring further inference to discover generic or specific answers. In a database of pure facts, these would all be specific answers. Resolving clauses most recently generated leads to depth-first search, which favors faster discovery of specific answers in a top-down search. When clauses are queued up in order of generation rather than pushed onto a stack, a top-down, breadth-first search, which favors generic answers, is accomplished. Bottom-up strategies favor immediate discovery of specific answers. Clauses may also be ordered by total length. Ordering clauses shortest to longest amounts to unit preference with a weighting factor of clause length, and selects for shorter answers.

While search strategies implicitly encode preferences for particular types of answers, they do not in general guarantee that answers will be produced exactly in the order desired. This is because it can't be assumed that certain types of answers exist at similar levels of a taxonomic hierarchy. Consider the following set of facts and rules: *cats like dogs*, *rover is a dog*, *fluffy is a cat*, and *fluffy likes tweety*. There is an asymmetry in the way the information about what fluffy likes is structured. The fact *fluffy likes tweety* stands on its own, while discovering that fluffy likes rover requires chaining through intermediate clauses that are not themselves specific answers. This simple example makes it clear that a preference for a particular type of answer does not guarantee that only the preferred type of answer will be produced in the course of reasoning.

Even if answers can be generated in nearly the order preferred by a user, there is still the problem that early answers may be subsumed by later answers. When this happens, early answers are redundant, and may even be misleading if they are used to generate responses to a questioner in addition to subsequent, more succinct answers. Clearly, it is easy to order answers *after* they have been generated, because such an ordering is guaranteed to reflect questioner or answering agent preferences. Thus, it is not surprising that current question answering systems tend to employ a sequential strategy of finding all answers (when possible), then ordering them.

We propose a new search strategy, termed *discharging hypotheticals*. This involves pushing a hypothetical answer clause onto the front of the stack of resolvents in the set of support as soon as it is generated. In other words, as soon as a question in the form of a hypothetical component of an answer clause presents itself, the answering agent tries to answer *that* intermediate question. It seems cognitively reasonable to attempt to settle doubts as quickly as possible, but, as previously discussed, an agent may not have the information necessary to resolve a hypothetical answer component. Rather than search within its own knowledge base for an answer, an answering agent could surface a hypothetical component as an immediate follow-up question to the questioner: a definitive answer to *that* question would allow reasoning to proceed immediately in the answering agent. Both strategies can be combined, allowing whichever method is faster to control the search time. Of course, the questioner may also not be able to answer the question posed by a hypothetical answer component, in which case a search must be resource limited to prevent agents from engaging in a potentially infinite, unprofitable search. Once a hypothetical has been discharged, this information may be cached for easy later retrieval. Note that this is analogous to the indexing of predicates performed by Prolog. The following is an outline of the search strategy:

Choose a clause to resolve from the front of the list of supported clauses
If the clause is a hypothetical answer clause
    Then check the cached information about hypotheticals already resolved to see if the
        hypothetical portion of the clause can be immediately discharged
        If the hypothetical component can be discharged
            Then do so, placing the resolvent(s) in the list of supported clauses
        Else try to resolve just the hypothetical portion of the clause with other clauses
            in the rule base and/or pose a follow-up question to the questioner based on
            the hypothetical portion of the clause
            If hypothetical is successfully resolved
                Then place the resolvent back in the list of suppported clauses as described

above for use in subsequent resolution
Else proceed with resolution as usual
EndIf
EndIf
Else proceed with resolution as usual
EndIf

The cached information, namely, what has been employed to discharge hypotheticals, may itself be interesting. That is, knowing that the fact that fluffy was a calico was critically important in answering a particular question might help in the formation of future queries, including possibly reformulating questions so that not as many hypotheticals are generated.

5.5 Taxonomic Relationships among Answer Clauses

While we have partitioned answer sets according to the form of answer clauses, we have not discussed the way in which answer sets might represent taxonomic information about answer clauses contained therein. One of the issues about taxonomic information is that it is not usually requested explicitly as part of a question. If an assumption is made that a questioner and answering agent have the same ontology, then conveying taxonomic information is unnecessary. We have suggested that this is not a reasonable assumption, nor should it be a requirement for an answering agent to provide good answers. Webber [1986] notes the importance of properly conveying sub- and super-class relations when giving answers. It is not merely that providing answers at different levels of generality may be confusing: sometimes a *questioner* has a misconception about taxonomic relationships between categories that might be cleared up by an answering agent. For example, if, in response to a question about what sorts of entities like to eat fish, an answering agent is able to provide the generic answers *cats like to eat fish* and *calicos like to eat fish*, it is important to convey the relationship between cats and calicos. A questioner with no prior knowledge of these categories might think them unrelated. On the other hand, a questioner may have heard of calicos and not cats, or vice versa. There is no guarantee that a questioner shares even the same set of basic categories [Rosch and Mervis, 1975]: in general it may be the case that one agent's ontology contains important "holes" when compared with that of another agent. It is only when an answer fits into what an agent already knows that it is useful. Conveying sub- and super-class information about answers helps to fulfull this goal.

There are several issues to be addressed: How should sub- and super-class relations among *answers* be determined; how should these relations be represented *in an answer set*; and how and when should they be conveyed to a questioner? First order predicate calculus is not an ideal knowledge representation and reasoning system for working with taxonomic relations. Description logics are better suited, and Borgida and McGuinness's [1996] work on classifying query *concepts* places queries directly in a taxonomic hierarchy so that answers are found above (super-concepts) and below (sub-concepts) the query concept in an existing hierarchy, and their positions and relationships are intrinsic to the representation. The knowledge representation and reasoning system ANALOG [Ali and Shapiro, 1993, Ali, 1994] is designed to capture the similarity between statements about individuals and classes, and thus facilitates representing taxonomic information. On the other hand, in response to a question, ANALOG returns a set of taxonomically related answers without clarifying the taxonomic relations therein, which may be confusing (see Section 2.1).

In order to demonstrate the utility of maintaining taxonomic information in an answer set, consider the answer set $\mathcal{A}$ as an example of what might be generated in response to the question *what barks?* .

1. {(ANSWER (barks rover))}

2. {(ANSWER (barks joe))}

3. {(ANSWER (barks x)), (dog x)}

4. {(ANSWER (barks y)), (poodle y)}

The primary partitions of $\mathcal{A}$ based on answer-clause type are $\mathcal{A}_s$ = {(ANSWER (barks rover))}, {(ANSWER (barks joe))}}; $\mathcal{A}_g$ = {{(ANSWER (barks x)), (dog x)}, {(ANSWER (barks y)), (poodle y)}}; and $\mathcal{A}_h = \emptyset$. If information to the effect that *rover is a poodle* and *all poodles are dogs* is available to the answering agent, it may be used to partition $\mathcal{A}$ based on taxonomic relations: $\mathcal{A}_1$ = {{(ANSWER (barks joe))}}; $\mathcal{A}_2$ = {(ANSWER (barks rover))}, {(ANSWER (barks y)), (poodle y)}, {(ANSWER (barks x)), (dog x)}}. For purposes of illustration the order of clauses in $\mathcal{A}_2$ conveys an ordering from most specific to most general, and this example ignores problems of tangled hierarchies, multiple inheritance, and so on. We raise the issue of taxonomic information because it is an important part of question answering. It is not, however, the focus of this dissertation.

5.6 Comparing Answers: Which Is Better?

Our definition of an answer set makes no mention of the *goodness* of answers. While it may appear that specific and generic answers are always better than hypothetical answers, having no attached contingencies, this is not necessarily the case. Consider the situation where the questioner has nothing in its ontology that allows it to relate to the specific and generic answers provided by an answering agent, but it does possess information needed to settle a hypothetical answer component in addition to having the knowledge needed to make sense of the accompanying generic or specific answer. For example, consider a question that asks which objects are blue. Assume that an answering agent provides the specific answer *The sky is blue.* , the generic answer *All dishes made by Delft are blue.* , and the hypothetical answer *If Mary is wearing her blue dress today then Mary's socks are blue.* . Further, assume that the questioner has neither knowledge of the sky nor knowledge about dishes, but that the questioner does know about Mary, dresses, and socks. In this case the hypothetical answer is a best answer. Thus, even when a questioner is not able to settle a hypothetical answer component, a hypothetical answer may "make sense" with respect to its ontology. In addition, the questioner may know of another agent to whom it may pose the question associated with the hypothetical answer component in case it can not be directly settled. In this case, the questioner might need to query another answering agent in order to find out whether or not Mary is wearing her blue dress today.

An observation about question answering systems is that definitions of answer goodness are varied and highly subjective. Most question answering systems are designed to return a particular type of answer, thus implicitly encode a definition of goodness in the control structure. When user preferences are considered, these have more to do with what a user knows than a preference for a particular answer type. This isn't surprising given the pervasive lack of recognition of different answer types.

While finding the *best answer* seems to be a holy grail of question answering systems, perhaps a better goal is to simply comply with Gricean maxims in attempting to be *as informative as possible* [Grice, 1975] within certain allowable time constraints imposed on the answering process. Our approach of sequential answer set generation is in line with Grice at each step of the reasoning process.

5.7 Complete Answers

There are several reasons to be concerned with completeness of answers. From a theoretical standpoint, completeness in general is a desirable property of systems. The use of a closed world assumption is a neat solution to the problem of incomplete information, which leads to incomplete answers. In general, however, the closed world assumption is impractical: the world is fundamentally incomplete. There are of course problems with partial answers, which may lead a questioner astray. Leaving out crucial information may cause a questioner to make poor decisions. Practically, it is difficult to ensure completeness.

A complete specific answer is simply the list of all satisfying individuals with respect to a question. Once generic answers are allowed, it is a bit trickier. There may not be a complete, pure generic answer to a question. Mixed answers (to borrow Motro's terminology [1994]) such as *all of the managers except for John Smith* and *all dogs plus Marky* may be required. In Motro's work, when there is a generic answer, it is viewed as a containment: it "contains" a set of specific individuals, and the generic is employed for reasons of succinctness and clarity. It is a *more general* way of referring to individuals in a database and, like an intensional set representation, is more likely to be correct over time as the membership in the set it denotes changes. If generic answers with "empty containments" are permitted, a generic is no longer "standing in" for a list of individuals. In fact, no individuals are known. Whether or not such answers are considered valid depends on philosophical considerations, but we aver that there is no reason in principle why such answers are not valid.

Cholvy and Demolombe [1986, 1990] defined a generic answer to a query $Q(x)$ as the complete set of of generic answers. This definition is problematic when generic answers at different levels of generality are provided together. This is the same problem encountered in ANALOG. For example, if cats eat fish, and tuna are fish, then two generic answers (cats eat fish, cats eat tuna) may be generated. To list both seems to imply that fish and tuna are disjoint categories of objects. Thus, a complete generic answer *must* also provide taxonomic information in order to avoid misleading a questioner. Cholvy and Demolombe's rule base did not include any specific answers; thus, their complete answer is naturally all generic.

The only way to ensure the production of complete answers is to employ a traditional sequential model of answering, where all answers are first discovered, then ordered and "cleaned up", and finally returned to a questioner. One question to ask is whether this is more important, or whether a sequence of partial answers is more useful. This depends on the preferences of the questioner, the knowledge base of the answering agent, and the purpose of the question. Knowing *why* a question is asked is clearly important in a question answering system. If a hospital needs to identify patients with a critical medical condition, a partial answer is unacceptable. If you are trying to get some idea of which of your friends attended a party, a complete answer may be unnecessary and pedantic.

Hypothetical answers are fundamentally incomplete. The question is whether or not a hypothetical answer should be included as part of a complete answer. If the purpose of a complete answer is to be *most helpful*, consider which of the following is more helpful:

John is at the park, and if Mike is there so is Pete.

John is at the park.

Which of these is more helpful depends on the questioner. Thus, we aver that, from a theoretical standpoint, hypothetical answers are not properly part of a complete answer. Considering the purpose of question answering, they are an important piece of what might be termed *complete information*. Returning a large large number of hypothetical answers to a question may be problematic: it may be annoying, time consuming, and may detract from the information-rich specific and generic answers. For example, there are limits to the usefulness of an answer such as *if Mike is at the park then Joe is at the park, and if Ray is at the park then Bill is at the park, and ..., and if Ron is at the park then Mark is at the park.* Thus, the notion of *complete answer* is correlated with the conjunction of a set of answers from which *all* other answers may be derived, and does not include hypothetical answers. The notion of *complete information* is broader, and includes hypothetical answers. Which hypothetical answers, and how many, are dependent upon the context in which a question is asked, which is an important research topic in the area of discourse understanding.

Another concern in question answering is defining the most general and most specific answer that might be provided by an answering agent. There are two reasons why this is not important in our work. First, our description of question answering as a process that happens over time leads to the idea of determining the most complete answer or most general answer *thus far*, rather than relying on a complete search for answers before any such categorization is made. Second, as previously discussed, the questioner may lack the ontological concepts necessary to understand a most general or most specific answer: the level of generality to the answering agent is not necessarily a measure that may be applied to the questioner. Defining most specific and most general is not difficult: a most specific answer is generally defined as the conjunction of the most specific concepts that comprise an answer, while a most general answer includes information at the highest level of generality possible.

Chapter 6

Implementation

The theories described in this dissertation have been developed and tested using a modified resolution refutation theorem prover. The original version of the theorem prover was written by Stuart C. Shapiro [1993] in Common Lisp, and incorporates standard resolution refutation with the set-of-support strategy, unit preference, factoring, and removal of tautologies and syntactically subsumed clauses from the knowledge base. Modifications to the theorem prover include changes to the control structure, additions to the data structure used to represent a clause, the addition of a data structure for the answer set, implementation of the new form of the answer literal described in chapter 4, and changes to way in which output is produced. The examples shown throughout this dissertation have been generated using this theorem prover, thus, we do not show any further examples in this section.

6.1 Control

There are two considerations related to the control of the theorem prover. The first is determining when the the resolution process should halt. Resolution refutation theorem provers are designed to halt when the empty clause is generated, which makes sense when the goal of resolution is to find a proof. Recall that, when an answer literal is employed, a proof is associated with a clause containing only answer literals. Because our interest is in finding all answers (when possible), stopping when the empty clause is generated is not appropriate. Instead, reasoning is halted when the set of support is empty. This occurs when no new resolvents can be generated. When the number of unique resolvents is infinite, some alternative mechanism must be available to halt the reasoning process because the set of support will never be empty. The problem of an infinite reasoning process is present even when the halting condition is the production of the empty clause (i.e., when there *is* no proof). Several of the modified versions of the theorem prover include alternative mechanisms for halting: prompting the user to see if he or she wishes to continue each time an answer is generated, which we term *Single Answer Mode*, halting after a specific number of reasoning steps, where the number of steps are specified by the user in advance, and halting after a certain amount of time specified by the user at the start of reasoning. Note that halting after each answer is found, and giving the user the option to continue, is not identical to the use of the semicolon in Prolog questioning. Prolog provides the semicolon option after the completion of a proof, and it gives the user the option of looking for an additional *proof*. Because answers in Prolog are all associated with proof, this is Prolog's version of *Single Answer Mode*.

The second control issue is the search strategy, namely, which clauses should be chosen at any point to resolve. Clauses are stored in two lists: those in the set of support, and those not in the set of support. The set of support is initialized to the question clauses, and the rest of the clauses comprise the starting knowledge base. In employing the set-of-support strategy, at least one clause to be resolved must be selected from the set of support.

The original version of the theorem prover incorporates unit preference, ordering the clauses in both of these sets from smallest to largest length. The length of a clause is defined to be the number of literals in the clause. Rather than relying on the built-in ordering of clauses, the user instead may elect to manually select the clauses to resolve at each step of resolution. A modified version of the theorem prover allows the user to specify a clause ordering so that the effect of different search strategies and preferences may be examined. The ordering criteria made available to the user are the following:

- Sort clauses in order of generation (oldest first)
- Sort clauses in reverse order of generation (most recent first)
- Sort clauses by length, shortest first
- Sort clauses by length, longest first

- Sort clauses by number negative literals from most to least
- Sort clauses by number negative literals from least to most
- Sort clauses by number positive literals from most to least
- Sort clauses by number positive literals from least to most
- Sort clauses placing hypothetical answer clauses first

These are not, for the most part, incompatible. For example, a selection of ordering by most recent clauses first plus hypothetical clauses first will result in hypothetical clauses at the front of the list in reverse order of generation, followed by the rest of the clauses in reverse order of generation.

6.2 Preferences

The user is given the option to specify a preference for either specific or generic answers. This preference affects both the search strategy and the way in which answers are returned to the user. Specific answers are discovered more quickly using depth-first search, while generic answers are associated with breadth-first search. Despite the indication of a preference, there is no guarantee that only answers of the desired type will be found and/or returned. There is, obviously, no "preference" for hypothetical answers as they are generally not to be preferred: whether or not hypothetical answers are returned depends upon the (lack of) information in the knowledge base, not on the user's choice. A preference for specific answers is taken as a preference for hypothetical specific answers over hypothetical generic answers, and vice versa.

6.3 Data Structures

The two primary data structures in the theorem prover are those for clauses and for the answer set. The set of support and the rest of the clauses in the knowledge base are lists of clauses, maintained in the order given by either the selected ordering criteria or according to the default order. A clause comprises the following components:

- number – the line number of the clause

- justification – either ”Assumption,” ”From Query”, or a pair of line numbers indicating the parent clauses
- hypo-clause – the hypothetical component of the clause
- gen-clause – the generic component of the clause
- ans-clause – the answer literals, comprising the question component of the clause

The only non-standard feature is the separation of the literals. For purposes of resolution, the entire clause comprising these three parts is utilized. The answer set comprises three lists of answer clauses: those corresponding to hypothetical answers, those corresponding to generic answers, and those corresponding to specific answers. These are ordered according to specified ordering preferences.

6.4 Input and Output

Questions and answers are given in standard FOPC. The theorem prover includes functions to convert from FOPC to clause form. Additional functions were added to convert answers from clause form back into an FOPC representation as specified by our canonical form for answers. The syntax of the FOPC matches that which is required for questions. The usefulness of this lies in the fact that a hypothetical answer component can immediately be posed as a new question. For self-questioning there is no need for translation, but of course, if this new question is posed by the answering agent to either the questioner or another agent it may need to be translated into yet another language.

6.5 Limitations

The theorem prover in its current state does not embody all of the features presented in our model of answering. In particular, the features required for disjunctive answering are not included. The modifications required for effective disjunctive answering are the conversion of clauses into canonical form before adding them to the answer set and the grouping of similar terms together in answers that are returned to the questioner. In addition, we have not to date considered creating collections of terms, which requires more than mere technical changes.

Answers and questions are given in standard FOPC. We have not attempted to perform any interaction using natural language, a clear limitation for a question-answering system if it is to be used to interact with people.

As stated early in the dissertation, we did not explore the answer selection or answer communication processes. While our hypothetical answers provide *questions* that may be utilized by the answering agent or the questioner, we did not allow the answering agent itself to ask questions directly, nor did we allow the questioner to receive information from the answering agent other than in the form of an answer. In addition, we employed only the minimal user model in our theorem prover: the set of selected answers.

Some of these limitations relate not to our particular focus but to the larger view of question-answering, namely, incorporating all features of our model and placing the system in a context where it must interact with a variety of questioners. Future work includes accomplishing precisely this task.

Chapter 7

Broader Applications of Our Approach to Answering

Our exploration of question answering has taken place in the context of resolution refutation. An important consideration is whether our ideas are more broadly applicable outside of this reasoning paradigm. Clearly, specific details of clause form, answer literals, and resolution are paradigm specific. In this chapter, we briefly explore some of the broader applications of our approach.

7.1 Reasoning Systems

Any reasoning system that performs back-chaining can potentially generate all of the types of answers described in this dissertation. For example, while Prolog only returns an answer when it finds a proof, it is generating intermediate results as it searches for a proof that can be captured using a meta-interpreter. These intermediate results correspond to generic and hypothetical answers. SNePS, a semantic network processing system discussed earlier in this dissertation [Shapiro, 1979, Shapiro and Rapaport, 1987, Shapiro and Rapaport, 1992], back-chains to try to satisfy the antecedents of a rule whose consequent has been matched. Our approach to answering suggests that a *partial* match of antecedents may be considered as an answer that is simply less informative than that which corresponds to a total match of the antecedents. The notion of partial matches of query concepts has been introduced in description logics. The presence of partial information is associated with hypothetical answers. As previously discussed, description logics are already capable of proving both generic and specific answers. Extending them to produce hypothetical answers may enable better explanation of partial query-concept matching.

7.2 Natural Language

The general applicability of specific, generic, and hypothetical answers to natural language may be examined with respect to a sample of Lehnert's question classes [Lehnert, 1978]. Table 7.2 shows the complete list of question classes proposed by Lehnert. While Lehnert's work is nearly twenty-five years old, her question classification scheme still forms the basis of question classification work in natural language understanding today. Graesser [1990] proposed a slightly different question classification based on his work on the psychological aspects of question answering, but it is Lehnert's categories that are most frequently cited in current work on question answering. Question classifications employed by groups participating in the TREC initiative are much shallower than Lehnert's categories, and are primarily syntax-based. For example, "how many", "who", and "what" are question types used in text analysis for question answering. Each of these syntactic question categories is associated with an answer template, enabling a more focused and precise answer search. It is not that semantic question categories and answer types are no longer necessary, rather that large-scale question answering is not sufficiently advanced to move to a categorization scheme as elaborate as Lehnert's, which was employed in a small, specifically focused system designed to investigate the semantics of question answering. As previously mentioned, the types of questions that require generic answers are linked to relatively poor performance in TREC, and hypothetical answering has not played a part in the competition. This chapter presents a short discussion of how the categories specific, generic, and hypothetical are related to natural language by examining a subset of Lehnert's question categories. A more detailed exploration of this topic is left for future work.

For each *question* class listed, an answer of each type (specific, generic, and hypothetical) is proposed. Note that repeated or habitual actions are associated with generic answers. This topic is extensively explored in work on generics [Krifka *et al.*, 1995]. Hypothetical answers are associated with possible explanations where there is a *lack of shared nominal referents* between a (possible) cause and effect. This is not surprising, given the connection between hypothetical answers and

Lehnert's Question Categories	
Question Type	Description
Causal Antecedent	asks about states or events that have caused the concept in question, e.g., *Why did you go home?*
Goal Orientation	special case of *why* question that asks about the mental goal of an action, e.g., *For what purpose did you borrow my key?*
Enablement	asks about the reason why something was able to happen, e.g., *How were you able to run so far?*
Causal Consequent	asks about states that are the consequence of the concept in question, e.g., *What happened when Joan left?*
Verification	asks whether it is true that some event occurred, e.g., *Did Mary leave?*
Disjunctive	the same as verification but with multiple question concepts, e.g., *Does Jean or Ray live here?*
Instrumental/Procedural	the question concept is a totally or partially unknown procedure for doing something or an instrument with which to accomplish something, e.g., *What did Fred use to eat?*, *What is the best way to get to UB from downtown?*
Concept Completion	typical who/what/when/where question with an unknown conceptual component in the question which should be "filled in" by the answer, e.g., *Where did you put my coat?*
Expectational	asks about a causal antecedent of an act that did *not* occur, e.g., *Why didn't you go to Rochester?*
Judgemental	solicits a judgement from the hearer, e.g., *What should I do to keep from failing my defense?*
Quantification	asks for an amount (or scalar adjective), e.g., *How many? How much? How badly does that hurt?*
Feature Specification	asks about some property of a given person or thing, e.g., *What color are Mary's eyes?, What kind of dog is Rover?*
Request	asks the hearer to perform an act, e.g., *Would you please pass the salt?*

abduction. In the following list, specific answers are indicated by S, generic by G, and hypothetical by H. Question classes and question examples are taken from [Lehnert, 1978, page 41].

1. Causal Antecedent: *Why did you go home?*

   S: I went home because it was 8:00.

   G: I always go home for dinner.

   H: If it was last Tuesday, I went home for my piano lesson.

   The specific answer is *specific* for the situation in question, the generic is associated with a habitual action, and the hypothetical indicates a *possible* explanation.

2. Verification: *Did Mary leave?*

   S: Mary left an hour ago.

   G: Mary always leaves at ten.

   H: If John left then Mary left.

   The nominal referent shared in the generic answer is *time*, which is not explicitly mentioned. For all times, if the time is ten then Mary leaves at *that time*. The antecedent and consequent of the hypothetical answer do *not* share a nominal referent.

3. Disjunctive: *Does Jean or Ray live here?*

   S: Jean or Ray lives here.

   G: All the students live here.

   H: If this is 345 Main Street then Jean lives here.

   Note that generic answers may require the questioner to perform inference. In this case, the inference would be that Jean and Ray are students. Answers that require inference on the part of the questioner are generally considered to be better answers than others [Lehnert, 1978].

4. Instrumental/Procedural: *What did Fred use to eat?*

   S: Fred used his fork.

   G: Fred always eats with his hands.

   H: If John didn't do the dishes this morning then Fred used a plastic spoon to eat.

   Note that some *if-then* sentences that *superficially* appear to be hypothetical answers are in fact generic. For example, consider the answer, "Whenever Fred eats a hotdog Fred uses his hands to eat." As mentioned in a previous example, what is shared here is a *event*: the event of Fred eating is *always* an event of Fred eating with his hands. The shared component is not reflected in the surface structure of the sentence.

5. Concept Completion: *Where did you put my coat?*

   S: I put your coat in the hall closet.

   G: If you put your coat on the chair then I put your coat in the closet.

   H: If that pink thing was your coat then I put it in the laundry pile.

   Here, the generic answer involves the shared referent "coat".

6. Expectational: *Why is the grass wet?*

   S: The grass is wet because it rained.

   G: The grass is always wet in the morning.

   H: If Bob ran the sprinkler then that is why the grass is wet.

   The generic answer involves a shared time referent.

7. Request: *Would you please pass the salt?*

   S: Yes, I will pass the salt.

   G: Yes, I always pass the salt when you ask for it.

   H: If you pass the potatoes, I'll pass the salt.

These examples demonstrate that the notions of specific, generic, and hypothetical answers are general and reach beyond the theorem proving context. Specific answers provide facts relevant to a particular situation, generic answers provide information that is relevant in *all* similar situations. The form of generic answers and the fact that they apply to situations over time associates them with habitual activity. Hypothetical answers describe *conditions that must hold* in order for an accompanying statement about a sitution to count as an answer.

The logical notion of the closure of variable sharing of a literal is not necessarily realized at the level of surface structure of a sentence. The "shared" variables in generic answers in natural language may be events or times to which a sentence *indirectly* refers. Thus, surface analysis of sentences is not sufficient in determining whether an answer is generic or hypothetical, rather, a logical form of the sentence that makes explicit all shared referents must be employed.

Hypothetical answers were logically characterized by a lack of sharing of variables between the antecedent and consequent. In natural language, similarly, a hypothetical answer is characterized by its form as an implication and the lack of shared referents, whether implicit or explicity, between the antecedent and consequent.

Chapter 8

Conclusions

When the idea of what counts as an answer is expanded to include all relevant information, question answering may be viewed as a process of searching for and returning information to a questioner that takes places in time. There is no preconceived, narrow notion of answer, which has its place but may lead to important and relevant information being overlooked. When there is not enough information to provide a specific or generic answer, which are typically those returned by a system, a hypothetical answer may be provided. Intensional or generic answers have been investigated primarily in the context of databases, where the intensional database component may be queried separately from the extensional component. When data mining techniques are employed, additional rules may be added to this component, enabling better and more up-to-date answers. Description logics provide a natural representation and reasoning mechanism for reasoning about and answering generic questions. Situating a query as a concept in a taxonomic hierarchy makes explicit the relationships among different answers, an important part of intelligent answering.

In our discussions of the meaning of the three answer types, we have clearly demonstrated their importance and applicability to question answering, including their relationship to natural language. In particular, we have focused on providing detailed characterizations of the class of hypothetical answers. Of the three answer classes, it is the least mentioned and analyzed, and few question answering systems have enabled the production of hypothetical answers.

Hypothetical answers *pose questions* that may be answered by an answering agent as reasoning proceeds, or returned to a questioner in the form of an antecedent to a hypothetical answer. Both approaches may be employed in case of time constraints, since it is possible that a questioner may find an answer more quickly. If the questioner knows the answer to *that* question, he or she has an immediate answer in hand. If not, it is explicitly clear to the questioner that the desired information is *not known* to the answering agent. Back-chaining reasoners produce potential generic and hypothetical answers "along the way" as the search for specific answers proceeds. These generics and hypotheticals are generally not considered as potential responses in a typical theorem prover due to the identification of answer with proof.

Our exploration of question answering has taken place using a resolution refutation theorem prover. In order to proceed with our work we first introduced an expanded notion of answer literal that may be used with any question expressible in FOPC. We have provided a complete classification of resolvents as answers within the theorem-proving paradigm. Our definition of the answer classes specific, generic, and hypothetical not only accounts for all resolvents, it provides a framework for understanding previous work on question answering and for comparing existing question answering systems. We have presented a canonical form for answers that unites the three answer types in a three-part implication. We have also explored the issue of disjunctive answers, and have suggested ways in which disjunctive answers may be presented in an informative manner.

While the dominant paradigm in theorem proving remains that of providing only specific answers, it has been demonstrated that a standard approach to theorem proving could provide additional, valuable information in the form of generic and hypothetical answers by simply paying attention to intermediate results that are currently not considered as interesting in their own right. Different search strategies can be devised to encode preferences for particular types of answers, though in general it can not be guaranteed that only answers of a desired type will be produced.

The foundational work presented in this dissertation will enable future work in the formalization of answers as well as provide a means for comparing and characterizing current question answering systems. Despite the apparent variety of such systems, the roots of question answering in theorem proving can be found in most, and the majority of their answers may be mapped onto our three categories.

We have proposed an *answer set* as a dynamic structure for keeping track of answers. Question answering as a process is viewed as a means of generating a sequence of answers sets that, in the limit, converge to the complete answer set for a given question and knowledge base. Formal properties of answer sets and their structure have been explored in detail.

Finally, we presented a reasoning-based model of answering that focuses on the answering side of question answering. The decoupling of answer search and selection enables a system to explicitly encode answer biases while allowing "undesirable" answers to be added to an answer set: over time, answers that are particularly sought may be found, but in case an answer is needed before then our model enables an answering agent to give *some* relevant information. The only case where an answering agent has an empty answer set is when there is no information relevant to a question in its knowledge base.

The chapter on question answering that started with Green's work in the late nineteen-sixties [1969b] is continued in this dissertation, which ties together all previous work on question answering in the context of resolution refutation and provides a complete, formal framework for the interpretation of resolvents as answers. We have clearly answered the question *what is an answer* within our chosen framework.

8.1 Future Directions for Research

There are a number of directions for future research based on the work presented in this dissertation. Some of these represent expanding and applying the theory to implemented systems, while others involve building bridges to work on question answering in linguistics and philosophy. Future work should validate the theory, show its scalability, and help to connect question-answering work in different disciplines. The following is a list of directions in which to take this research.

1. Thoroughly explore the problem of disjunctive answering, including the separation of disjuncts into groups and the representation of multiple terms as collections.

2. Examine the inner structure of the hypothetical component of answers.

3. Explore methods for pruning the search tree in the face of an intractable number of hypothetical answers.

4. Test the theories in a system that answers questions using a large knowledge base and is capable of returning all types of answers.

5. Explore the efficacy of these ideas in open-domain question answering that involves large text corpora.

6. Use this work to formally extend Belnap and Steel's philosophical foundations of question answering [Belnap and Steel, 1976].

7. Tie this work into existing linguistic theories of question answering.

8. Expand the types of questions considered and determine the relationship of new question classes to these answer classes.

9. Consider alternative representations of generic answers, particularly those involving locally-scoped existential quantifiers, and determine when such an alternative representation better accords with the semantics of a generic answer.

10. Examine text corpora for hypothetical information. The existence of a significant number of counterfactuals may argue for an approach to question answering that enables the retrieval and interpretation of these structures, provided the consequents of these counterfactuals correspond to interesting questions.

Appendix A

Glossary of Terms and Symbols

answer All relevant clauses except for the question clauses are answers.

answer clause A clause that contains at least one answer literal.

answer component The union of the generic component and the question component of an answer clause.

answer literal A literal of the form $(\mathcal{ANSWER}\ (P\overline{xc}))$, where $\overline{x}$ is a list of one or more variables and $\overline{c}$ is a list of one or more constants.

$\mathcal{ANSWER}$ **predicate** A special predicate denoted $\mathcal{ANSWER}$ that is not present in a knowledge base prior to reasoning about a question and is introduced with a question clause.

answer set An answer set $\mathcal{S}$ is a set of answer clauses c_i such that

$\forall\, c_i \in \mathcal{S}, \mathcal{INFM}(c_i, \mathcal{S})$.

answer set with respect to question $\mathcal{Q}$ An answer set with respect to question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$ is defined as follows:

- $\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},0]} = \emptyset$
- $\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]} = \text{Update}(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i-1]},\, a_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}), \forall\, i > 0.$

augmented negation An intermediate form of a question as it is converted to clause form that has the form:

$[\overline{P}(\overline{x}) \Rightarrow ANSWER(\overline{P}(\overline{x}))]$

$\overline{P}(\overline{x})$ represents the original question in prenex normal form that has been Skolemized and contains no universal quantifiers.

canonical form (of an answer clause) An answer clause $\mathcal{C}$ is in canonical form if all of its literals that unify with the literals in the original question clauses are arguments of ANSWER predicates.

$\mathcal{CLEAN}(\mathcal{S})$ For a set $\mathcal{S}$ of answer clauses, $\mathcal{CLEAN}(\mathcal{S}) = \mathcal{S}'$ where $\mathcal{S}'$ is the set of the informative members of $\mathcal{S}$.

closure of variable sharing of answer literals The closure of variable sharing of the set of answer literals in a clause.

closure of variable sharing of a set of literals The closure of variable sharing of a set of literals $\mathcal{L}$, denoted $\mathcal{CVS}(\mathcal{L})$, is the fixed point of the $\mathcal{VS}$ operator.

complete answer set The complete answer set for a question $\mathcal{Q}$ and a knowledge base $\mathcal{KB}$ is the fixed point of the $\mathcal{GENANS}$ function, where $\mathcal{GENANS}$ is associated with any appropriate control structure for reasoning. We denote the complete answer set $\mathcal{A}_{[\mathcal{Q},\mathcal{KB}]}$.

create operation Create($\mathcal{A}$) creates a new answer set $\mathcal{A}$ and intializes it to the empty set.

$\mathcal{CVS}(\mathcal{L})$ Closure of variable sharing of a set of literals $\mathcal{L}$.

disjunctive answer An answer associated with a clause that contains more than one answer literal.

find answers algorithm An algorithm for finding answers that involves repeated application of the $\mathcal{GENANS}$ function.

$\mathcal{GENANS}$ **function** $\mathcal{GENANS}$ is defined with respect to question $\mathcal{Q}$, knowledge base $\mathcal{KB}$, and control structure $\mathcal{CS}$ as follows:

$\mathcal{GENANS}(\mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i]}) = \mathcal{A}_{[\mathcal{Q},\mathcal{KB},\mathcal{CS},i+1]}$

generic answer A generic answer is a universally quantified formula associated with a generic answer clause. The (possibly empty) antecedent is the conjunction of the negations of all of the non-answer literals in the clause. The consequent is the disjunction of all of the arguments of the answer literals in the clause. Skolem constants and functions are replaced by variables.

Variables local to the consequent are universally quantified with narrow scope. All other variables are universally quantified with wide scope over the entire formula.

generic answer clause A generic answer clause is a clause with the following properties:

1. There is at least one answer literal that contains a variable, a Skolem constant, or a Skolem function.
2. All non-answer literals must be in the closure of variable sharing of the answer literals.

generic answer literal A non-answer literal that is in the closure of variable sharing of the answer literals in a clause.

generic component The set of generic answer literals in a clause.

hypothetical answer A hypothetical answer is a rule associated with a hypothetical answer clause. The antecedent of the rule is the conjunction of the negations of the non-answer literals in the clause that are not in the closure of variable sharing of the answer literals. Any variables in the antecedent are existentially quantified with narrow scope over the antecedent. Any Skolem functions in the antecedent are replaced by universally quantified variables with narrow scope over the antecedent. The remaining literals in the clause correspond either to a specific or a generic answer that comprises the consequent of the rule.

hypothetical answer clause A hypothetical answer clause is a clause with the following properties:

1. There is at least one answer literal.
2. There is at least one non-answer literal that is not in the closure of variable sharing of the answer literals.

hypothetical answer literal A non-answer literals that is not in the closure of variable sharing of the answer literals in a clause.

hypothetical component The set of hypothetical answer literals in a clause.

hypothetical generic answer clause A hypothetical answer clause whose answer component is a generic answer.

hypothetical specific answer clause A hypothetical answer clause whose answer component is a specific answer.

$\mathcal{INF}(c,\mathcal{S})$ Informative answer clause.

$\mathcal{INFM}(c_i,\ \mathcal{S})$ Informative member.

informative answer clause An answer clause c is an informative answer clause with respect to a set of answer clauses $\mathcal{S}$, written $\mathcal{INF}(c,\mathcal{S})$, if c is either an informative specific answer clause, an informative generic answer clause, or an informative hypothetical answer clause with respect to $\mathcal{S}$.

informative answer clause An answer clause c is an informative answer clause with respect to a set of answer clauses $\mathcal{S}$, written $\mathcal{INF}(c,\mathcal{S})$, if c is either an informative specific answer clause, an informative generic answer clause, or an informative hypothetical answer clause with respect to $\mathcal{S}$.

informative member An answer clause c is an informative member of a set of answer clauses $\mathcal{S}$, written $\mathcal{INFM}(c,\ \mathcal{S})$, if $\mathcal{INF}(c,(\mathcal{S}\ \setminus\ \{c\}))$.

informative specific answer clause A specific answer clause c_s is informative with respect to a set of answer clauses $\mathcal{S}$ if $\neg(\mathcal{SUBS}(\mathcal{S}_s,\ c_s))$.

informative generic answer clause A generic answer clause c_g is informative with respect to a set of answer clauses $\mathcal{S}$ if $\neg(\mathcal{SUBS}(\mathcal{S}_g,\ c_g))$.

informative hypothetical answer clause A hypothetical answer clause c_h is informative with respect to a set of answer clauses $\mathcal{S}$ if it is either an informative hypothetical specific answer clause with respect to $\mathcal{S}$ or an informative hypothetical generic answer clause with respect to $\mathcal{S}$.

informativeness relation An answer set $\mathcal{S}$ is more informative than an answer set $\mathcal{T}$ if the following hold: for all clauses $c_s\ \in\ (\mathcal{S}\ \setminus\ \mathcal{T})\}$, it is the case that $\mathcal{INF}(c_s,\ \mathcal{T})$, and, for all clauses $c_t\ \in\ (\mathcal{T}\ \setminus\ \mathcal{S})$, it is that case that $\neg\mathcal{INF}(c_t,\ \mathcal{S})$.

non-answer literal A literal whose predicate is anything other than $\mathcal{ANSWER}$.

question clause A question clause is a clause that is generated when the augmented negation is converted to clause form. That is, a clause associated with the original question posed to an answering agent.

question component The set of answer literals in a clause.

relevant Given a knowledge base $\mathcal{KB}$, a question $\mathcal{Q}$, and question clauses $\mathcal{QC}$, a clause $\mathcal{C}$ is relevant to $\mathcal{Q}$ if and only if the following is true:

$\mathcal{C} \in \text{Th}(\mathcal{KB} \cup \mathcal{QC}) \setminus \text{Th}(\mathcal{KB})$.

$\mathcal{S}_s$ The specific answer clauses in a set of clauses $\mathcal{S}$.

$\mathcal{S}_g$ The generic answer clauses in a set of clause $\mathcal{S}$.

$\mathcal{S}_h$ The hypothetical answer clauses in a set of clause $\mathcal{S}$.

$\mathcal{S}_{h_s}$ The hypothetical specific answer clauses in a set of clause $\mathcal{S}$.

$\mathcal{S}_{h_g}$ The hypothetical generic answer clauses in a set of clause $\mathcal{S}$.

specific answer A specific answer is the disjunction of the arguments of all of the answer literals in a specific answer clause.

specific answer clause A specific answer clause is a clause with the following properties:

1. It contains at least one answer literal.
2. It contains no non-answer literals.
3. All literals are ground.
4. There are no Skolem functions or constants present in literals.

syntactic subsumption A set of clauses $\mathcal{S}$ syntactically subsumes a clause c, written $\mathcal{SUBS}(\mathcal{S}, c)$, if, for some clause $c_i \in \mathcal{S}$, $\mathcal{SUBS}(c_i, c)$.

$\mathcal{SUBS}(\mathcal{S}, c)$ Syntactic subsumption.

unconstrained generic answer An answer where there are no restrictions on the variables, derived from a clause that contains only answer literals where the answer literals contain at least one variable.

unconstrained variable A variable the occurs in an answer literal that does not occur in any non-answer literal.

update function The update function is defined on an answer set $\mathcal{A}$ with respect to a clause c as follows: Update($\mathcal{A}$, c): $\mathcal{A} \times c \longrightarrow \mathcal{A}'$, where c is an answer clause, $\mathcal{A}$ and $\mathcal{A}'$ are answer sets, and $\mathcal{A}' = \mathcal{CLEAN}(\mathcal{A} \cup \{c\})$

Note that if c is not informative with respect to $\mathcal{A}$ then Update($\mathcal{A}$, c) = $\mathcal{A}$

variable sharing set of a literal The variable sharing set of a literal l with respect to a set of literals C, is a set of literals $\mathcal{VS}$(l) such that each literal in $\mathcal{VS}$(l) is an element of C and shares at least one variable with l.

variable sharing set of a set of literals The variable sharing set of a finite set of literals $\mathcal{S} = \{l_1, l_2, \ldots, l_n\}$ is the union of the variable sharing sets for each of the literals: $\mathcal{VS}(\mathcal{S}) = \bigcup_{l_i \in \mathcal{S}} \mathcal{VS}(l_i)$.

$\mathcal{VS}$ This symbol denotes the variable sharing operation.

Bibliography

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

[Ali, 1994] Syed S. Ali. *A Natural Logic for Natural Language Processing and Knowledge Representation*. Phd dissertation technical report 94-01, Department of Computer Science, State University of New York at Buffalo, 1994.

[Allen, 1995] James Allen. *Natural Language Understanding*. Benjamin Cummings, Redwood City, CA, second edition, 1995.

[Anderson and Nuel D. Belnap, 1975] Alan Ross Anderson and Jr. Nuel D. Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, Princeton, NJ, 1975.

[Barwise and Perry, 1983] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, Cambridge, MA, 1983.

[Belnap and Steel, 1976] Nuel D. Belnap and Thomas B. Steel. *The Logic of Questions and Answers*. Yale University Press, New Haven, CT, 1976.

[Berwick, 1983] Robert C. Berwick. Introduction: Computational Aspects of Discourse. In *Computational Models of Discourse*, pages 27–105. MIT Press, Cambridge, MA, 1983.

[Bonnet and Tomasic, 1998] Philippe Bonnet and Anthony Tomasic. Partial answers for unavailable data sources. In Troels Andreasen, Henning Christiansen, and Henrik L. Larsen, editors, *Flexible Query Answering Systems*, Lecture Notes in Artificial Intelligence, pages 43–54. Springer–Verlag, 1998.

[Borgida and McGuinness, 1996] Alexander Borgida and Deborah L. McGuinness. Asking Queries about Frames. In *Proceedings of KR-96*, pages 340–349, Cambridge MA, 1996. Morgan Kaufmann.

[Bromberger, 1992] Sylvain Bromberger. *On What We Know We Don't Know*. The University of Chicago Press; Center for the Study of Language and Information, Chicago; Stanford, 1992.

[Burhans and Shapiro, 1999] Debra T. Burhans and Stuart C. Shapiro. Finding Answers with a Resolution Theorem Prover. In Vinay Chaudhri and Richard Fikes, editors, *Question Answering Systems: Papers from the 1999 AAAI Fall Symposium, Technical Report FS-99-02*. AAAI, 1999.

[Burhans and Shapiro, 2001] Debra T. Burhans and Stuart C. Shapiro. Abduction and Question Answering. In *Proceedings of the IJCAI-01 Workshop on Abductive Reasoning*, Seattle, WA, 2001. AAAI.

[Chalupsky and Russ, 2002] Hans Chalupsky and Thomas A. Russ. WhyNot: Debugging Failed Queries in Large Knowledge Bases. *submitted to IAAI*, 2002.

[Chang and Lee, 1973] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, NY, 1973.

[Charniak, 1988] Eugene Charniak. Motivation analysis, abductive unification, and nonmonotonic equality. *Artificial Intelligence*, 34:275–295, 1988.

[Choi, 1993] Joongmin Choi. *Experience-Based Learning in Deductive Reasoning Systems*. Phd dissertation technical report 93-20, Department of Computer Science, State University of New York at Buffalo, 1993.

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

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

[Chu *et al.*, 1994] Wesley W. Chu, Q. Chen, and M. Merzbacher. CoBase: a cooperative database system. In R. Demolombe and T. Imielinski, editors, *Nonstandard Queries and Nonstandard*

Answers, Studies in Logic and Computation, chapter 2, pages 41–72. Clarendon Press, Oxford, 1994.

[Chu *et al.*, 1996] Welsey W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. Cobase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6, 1996.

[Clark *et al.*, 1999] Peter Clark, John Thompson, and Bruce Porter. A knowledge-based approach to question answering. In Vinay Chaudhri and Richard Fikes, editors, *Question Answering Systems: Papers from the 1999 AAAI Fall Symposium, Technical Report FS-99-02*, pages 43–51. AAAI, 1999.

[Clark, 1978] K. L. Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York NY, 1978.

[Cox and Pietrzykowski, 1984] P. T. Cox and T. Pietrzykowski. A complete, nonredundant algorithm for reversed skolemization. *Theoretical Computer Science*, 28:239–261, 1984.

[Cox and Pietrzykowski, 1986] P. T. Cox and T. Pietrzykowski. Causes for events: Their computation and applications. In *Proceedings of the 8th International Conference on Automated Deduction*, Lecture Notes in Computer Science, pages 608–621. Springer–Verlag, 1986.

[Cox and Pietrzykowski, 1987] P.T. Cox and T. Pietrzykowski. General diagnosis by abductive inference. In *Proceedings of the IEEE Symposium on Logic Programming*, pages 183–189. IEEE Press, 1987.

[Dahl, 1983] Veronica Dahl. Logic programming as a representation of knowledge. *IEEE Computer*, 16(10), October 1983.

[Dalal, 1996] Mukesh Dalal. Semantics of an anytime family of reasoners. In *Proceedings of the European Conference on Artificial Intelligence*, pages 360–364, Budapest, 1996.

[Dale and Reiter, 1995] R. Dale and E. Reiter. Computational Interpretations of the Gricean Maxis in the Generation of Referring Expressions. *Cognitive Science*, 19:233–263, May-June 1995.

[Dale and Reiter, 1996] Robert Dale and Ehud Reiter. The Role of the Gricean Maxims in the Generation of Referring Expressions. In *Proceedings of the AAAI 1996 Spring Symposium on Computational Models of Conversational Implicature*, Stanford University, CA, 1996.

[Davis and Putnam, 1960] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7(3):201–215, 1960.

[Demolombe, 1992] Robert Demolombe. A strategy for the computation of conditional answers. In Bernd Neumann, editor, *Proceedings of the 10th European Conference on Artificial Intelligence, ECAI 92*, pages 134–138, Vienna, 1992. John Wiley and Sons.

[Demolombe, 1997] Robert Demolombe. Uncertainty in intelligent databases. In Amihai Motro and Philippe Smets, editors, *Uncertainty Management in Information Systems*, pages 89–126. Kluwer Academic Publishers, 1997.

[Dimopoulos and Kakas, 1996] Yannis Dimopoulos and Antonis Kakas. Abduction and Induction: an AI Perspective. In *Proceedings of the ECAI '96 Workshop on Abductive and Inductive Reasoning*, pages 68–70, Budapest, 1996.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, and Werner Nutt. Adding Epistemic Operators to Concept Languages. In *Proceedings of KR-92*, pages 342–353, Cambridge MA, 1992. Morgan Kaufmann.

[Fine, 1983] Kit Fine. In Defense of Arbitrary Objects. In *Proceedings of the Aristotelian Society*, volume Volume Suppl. LVII, pages 55–77. Aristotelian Society, 1983.

[Fine, 1985] Kit Fine. Natural Deduction and Arbitrary Objects. *Journal of Philosophical Logic*, 14:57–107, 1985.

[Forgy, 1982] Charles Forgy. Rete: A fast algorithm for the many patterns/many objects match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[Gaasterland *et al.*, 1994] Terry Gaasterland, Parke Godfrey, and Jack Minker. An Overview of Cooperative Answering. In R. Demolombe and T. Imielinski, editors, *Nonstandard Queries and Nonstandard Answers*, Studies in Logic and Computation, chapter 1, pages 1–40. Clarendon Press, Oxford, 1994.

[Gallier, 1986] Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*, chapter 10, pages 448–476. Harper & Row, New York, 1986.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Logic Programming: Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080, 1988.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Gilbert, 1988] G. Nigel Gilbert. Question and answer types. In *Research and Development in Expert Systems IV*, British Computer Society workshop series, pages 162–172. Cambridge University Press, Cambridge, 1988.

[Ginsberg, 1993] Matt Ginsberg. *Essentials of Artificial Intelligence*. Morgan Kaufmann Publishers, 1993.

[Ginzburg, 1996] Jonathan Ginzburg. Interrogatives: Questions, Facts and Dialogue. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell Publishers, 1996.

[Godfrey and Gryz, 1996] Parke Godfrey and Jarek Gryz. Intensional Query Optimization. Technical Report UMIACS TR 96-72, University of Maryland at College Park, College Park, MD, October 1996.

[Godfrey *et al.*, 1994] Parke Godfrey, Jack Minker, and Lev Novik. An Architecture for a Cooperative Database System. In *Proceedings of the 1994 International Conference on Applications of Databases*, volume 819 of *Lecture Notes in Computer Science*, Vadstena Sweden, June 1994. Springer Verlag.

[Golding *et al.*, 1990] Jonathan M. Golding, Arthur C. Graesser, and Keith K. Millis. What makes a good answer to a question? testing a psychological model of question answering in the context of narrative text. *Discourse Processing*, 13:305–325, 1990.

[Graesser and Franklin, 1990] Arthur C. Graesser and Stanley P. Franklin. Quest: A cognitive model of question answering. *Discourse Processing*, 13:279–303, 1990.

[Graesser *et al.*, 1992] Arthur C. Graesser, Sallie E. Gordon, and Lawrence E. Brainerd. QUEST: A Model of Question Answering. In F. Lehmann, editor, *Semantic Networks in Artificial Intelligence*. Pergamon Press, 1992.

[Green and Raphael, 1968] C. Cordell Green and Bertram Raphael. The user of theorem-proving techniques in question-answering systems. In Sr. Richard B. Blue, editor, *Proceedings of 23rd National Conference*, pages 169–181, Princeton, NJ, 1968. Association for Computing Machinery, Brandon/Systems Press, Inc.

[Green, 1969a] Cordell Green. Applications of theorem proving to problem solving. In Donald E. Walker and Lewis M. Norton, editors, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 219–239. IJCAI, May 1969.

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

[Grice, 1975] H. P. Grice. Logic and Conversation. In P. Cole and J. L. Morgan, editors, *Speech Acts*, volume 3 of *Syntax and Semantics*. Seminar Press, 1975.

[Groenendijk and Stokhof, 1993] Jeroen Groenendijk and Martin Stokhof. Interrogatives and adverbs of quantification. In K. Bimbo and A. Mate, editors, *Proceedings of the 4th Symposium on Logic and Language*, 1993.

[Groenendijk and Stokhof, 1997] Jeroen Groenendijk and Martin Stokhof. Questions. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 19, pages 1055–1124. MIT Press, Cambridge, MA, 1997.

[Groenendijk, 1999] Jeroen Groenendijk. The logic of interrogation classical version. In T. Matthews and D. L. Stolovitch, editors, *The Proceedings of the Ninth Conference on Semantics and Linguistic Theory*, Santa Cruz, February 1999. CLC Publications.

[Harabagiu *et al.*, 2001] Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. FALCON: Boosting Knowledge for Answer Engines. In E. M. Voorhees and D. K. Harman, editors, *NIST Special*

Publication XXX-XXX: The Ninth Text REtrieval Conference (TREC-9). National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2001. Pre-publication version of the proceedings located at (http://trec.nist.gov/pubs/trec9/t9_proceedings.html).

[Higginbotham, 1996] James Higginbotham. The Semantics of Questions. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 362–383. Blackwell Publishers, 1996.

[Hintikka, 1985] Jaakko Hintikka. True and false logics of scientific discovery. In Jaakko Hintikka and Fernand Vandamme, editors, *Logic of Discovery and Logic of Discourse*. Plenum Press, 1985.

[Hirschberg, 1991] Julia Hirschberg. *A Theory of Scalar Implicature*. Garland Publishing, Inc., New York NY, 1991.

[Hobbs *et al.*, 1993] Jerry R. Hobbs, Mark Stickel, Douglas Apelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63(1–2):69–142, 1993.

[Imielinski and Mannila, 1996] Tomasz Imielinski and Heikki Mannila. A Database Perspective on Knowledge Discovery. *Communications of the ACM*, 39(11):58–64, November 1996.

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

[Joshi and Kaplan, 1977] Aravind K. Joshi and S. Jerrold Kaplan. Approximate responses from a data base query system: An application of inferencing in natural language. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 211–212. IJCAI, 1977.

[Joshi *et al.*, 1981] Aravind K. Joshi, Bonnie J. Weber, and Ivan A. Sag. *Elements of Discourse Understanding*. Cambridge University Press, Cambridge, 1981.

[Jurafsky and Martin, 2000] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice Hall, 2000.

[Kakas *et al.*, 1998] A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in AI and Logic Programming 5*, pages 235–324. Oxford University Press, 1998.

[Kaplan, 1981] S. Jerrold Kaplan. Appropriate responses to inappropriate questions. In Aravind K. Joshi, Bonnie J. Weber, and Ivan A. Sag, editors, *Elements of Discourse Understanding*, pages 127–144. Cambridge University Press, Cambridge, 1981.

[Kaplan, 1984] Jerrold Kaplan. Designing a portable natural language database query system. *ACM Transactions on Database Systems*, 9(1):1–19, March 1984.

[Karttunen, 1977] L. Karttunen. Syntax and Semantics of Questions. *Linguistics and Philosophy*, 1:3–44, 1977.

[Kempson, 1975] Ruth M. Kempson. *Presupposition and the Delimitation of Semantics*. Cambridge University Press, 1975.

[Klahr *et al.*, 1980] Philip Klahr, Larry Travis, and Charles Kellogg. A deductive system for natural-language question answering. In Leonard Bolc, editor, *Natural Language Question Answering Systems*, Natural Communication with Computers, pages 73–136. Carl Hanser Verlag, 1980.

[Kowalski *et al.*, 1998] R. A. Kowalski, F. Sadri, and F. Toni. An agent architecture that combines backward and forward reasoning. In B. Gramlich and F. Pfenning, editors, *Proceedings of CADE-15 Workshop on Strategies in Automated Deduction*, pages 49–56, 1998.

[Kowalski, 1979] Robert Kowalski. *Logic for Problem Solving*. Artificial Intelligence Series. North Holland, New York, NY, 1979.

[Krifka *et al.*, 1995] Manfred Krifka, Francis Jeffry Pelletier, Gregory N. Carlson, Alice ter Meulen, Gennaro Chierchia, and Godehard Link. Genericity: An Introduction. In Gregory N. Carlson and Francis Jeffry Pelletier, editors, *The Generic Book*, chapter 1, pages 1–124. University of Chicago Press, Chicago, IL, 1995.

[Krifka, 1999] Manfred Krifka. For a structure account of questions and answers. In C. Smith, editor, *Proceedings of the Workshop on Spoken and Written Text*, Austin, TX, 1999. University of Texas at Austin.

[Kuhns, 1967] J. L. Kuhns. Answering questions by computer: A logical study. Technical Report Rand Memorandum RM-5428-PR, The Rand Corporation, Santa Monica, CA, December 1967.

[Lehnert, 1978] Wendy G. Lehnert. *The Process of Question Answering*. Lawrence Erlbaum, 1978.

[Lehnert, 1981] Wendy G. Lehnert. A computational theory of human question answering. In Aravind K. Joshi, Bonnie J. Weber, and Ivan A. Sag, editors, *Elements of Discourse Understanding*, pages 145–176. Cambridge University Press, Cambridge, 1981.

[Lenat, 1995] Douglas B. Lenat. CYC: A Large Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38, November 1995.

[Lenzerini and Schaerf, 1991] Maurizio Lenzerini and Andrea Schaerf. Concept Language as Query Languages. In *Proceedings of AAAI-91*, pages 471–476, Anaheim CA, 1991. AAAI Press/MIT Press.

[Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Programming*. Symbolic Computation – Artificial Intelligence. Springer-Verlag, second, expanded edition, 1987.

[Loveland, 1978] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis*. Fundamental Studies in Computer Science. North-Holland, 1978.

[Luckham and Nilsson, 1971] David Luckham and Nils J. Nilsson. Extracting information from resolution proof trees. *Artificial Intelligence*, 2:27–54, 1971.

[Luger and Stubblefield, 1997] George F. Luger and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley, third edition, 1997.

[MacGregor and Bates, 1987] R. MacGregor and R. Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, USC Information Sciences Institute, 1987.

[Magnani, 1998] Lorenzo Magnani. Abduction and hypothesis withdrawal in science. In *Twentieth World Congress of Philosophy*, Boston, MA, August 1998. The Paideia Project On-Line. (http://www.bu.edu/wcp/Papers/Scie/ScieMagn.htm).

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

[Michalski, 1993] R. S. Michalski. Inferential theory of learning as a conceptual basis for multistrategy learning. *Machine Learning*, 11:111–151, 1993.

[Minock and Chu, 1996] Michael J. Minock and Wesley W. Chu. Explanation for cooperative information systems. In *Proceedings of the Ninth International Symposium on Methogologies for Intelligent Systems*, June 1996.

[Moldovan *et al.*, 2000] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. LASSO: A Tool for Surfing the Answer Net. In E. M. Voorhees and D. K. Harman, editors, *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2000. (http://trec.nist.gov/pubs/trec8/t8_proceedings.html).

[Motro and Rakov, 1997] Amihai Motro and Igor Rakov. Not all answers are equally good: Estimating the quality of database answers. In Troels Andreasen, Henning Christiansen, and Henrik Legind Larsen, editors, *Flexible Query Answering Systems*, pages 1–21. Kluwer, 1997.

[Motro, 1989] Amihai Motro. Using integrity constraints to provide intensional answers to relational queries. In *Proceedings of VLDB89, the 15th International Conference on Very Large Databases*, pages 237–246, Amsterdam The Netherlands, August 1989. VLDB89.

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

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

[Nilsson, 1980] Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, 1980.

[Nilsson, 1998] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1998.

[Peirce, 1932] Charles Sanders Peirce. *Collected Paper of Charles Sanders Peirce: Elements of Logic*, volume II. Harvard University Press, Cambridge, MA, 1st edition, 1932.

[Pirotte *et al.*, 1991] Alain Pirotte, Dominique Roelants, and Esteban Zimanyi. Controlled generation of intensional answers. *IEEE Transactions on Knowledge and Data Engineering*, 3(2):221–236, June 1991.

[Plaisted, 1993] David A. Plaisted. Equational reasoning and term rewriting systems. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in AI and Logic Programming 1*, pages 276–277. Oxford University Press, 1993.

[Poole *et al.*, 1987] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: A Logical Reasoning System for Defaults and Diagnosis. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier, Essays in the Representation of Knowledge*, pages 331–352. Springer-Verlag, New York, 1987.

[Poole, 1989] David Poole. Explanation and prediction: An architecture for default and abductive reasoning. *Computational Intelligence*, 5(2):97–110, 1989.

[Poole, 1990] David Poole. A methodology for using a default and abductive reasoning system. *International Journal of Intelligent Systems*, 5(5):521–548, 1990.

[Prior and Prior, 1955] A. Prior and M. Prior. Erotetic logic. *Philosophical Review*, 64:43–59, 1955.

[Ram, 1991] Ashwin Ram. A theory of questions and question asking. *The Journal of the Learning Sciences*, 1(3&4):273–318, 1991.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems. In *Proceedings of the 6th National Conference on Artificial Intelligence, AAAI-87*, volume 1, pages 183–188. MIT Press, 1987.

[Reiter, 1978a] Raymond Reiter. Deductive question-answering in relational data bases. In Herve Gallaire and Jack Minker, editors, *Logic and Databases*, pages 149–177. Plenum Press, New York NY, 1978.

[Reiter, 1978b] Raymond Reiter. On closed world data bases. In Herve Gallaire and Jack Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, New York NY, 1978.

[Reiter, 1986] Raymond Reiter. A sound and sometimes complete query evaluation for relational databases with null values. *Journal of the ACM*, 33(2):349–370, 1986.

[Rich and Knight, 1991] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw Hill, 2 edition, 1991.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[Rosch and Mervis, 1975] E. Rosch and C. B. Mervis. Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology*, 4:573–605, 1975.

[Russell and Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[Satoh, 2000] Ken Satoh. Consistency management of normal logic program by top-down abductive proof procedure. In *Proceedings of the 8th International Workshop on Nonmonotonic Reasoning (NMR2000)*, April 2000.

[Sergot, 1983] M. Sergot. A query-the-user facility for logic programming. In P. Degano and E. Sandewall, editors, *Integrated Interactive Computer Systems*, pages 27–41. North Holland, 1983.

[Shapiro and Rapaport, 1987] Stuart C. Shapiro and William J. Rapaport. SNePS Considered as a Fully Intensional Propositional Semantic Network. In Nick Cercone and Gordon McCalla, editors, *The Knowledge Frontier, Essays in the Representation of Knowledge*, pages 262–315. Springer-Verlag, New York, 1987.

[Shapiro and Rapaport, 1992] Stuart C. Shapiro and William J. Rapaport. The SNePS Family. In F. Lehmann, editor, *Semantic Networks in Artificial Intelligence*. Pergamon Press, 1992.

[Shapiro and Woodmansee, 1969] Stuart C. Shapiro and George H. Woodmansee. A net structure based relational question answerer: Description and examples. In Donald E. Walker and Lewis M. Norton, editors, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 325–345. IJCAI, May 1969.

[Shapiro *et al.*, 1981] Stuart C. Shapiro, Donald P. McKay, João Martins, and Ernesto Morgado. SNePSLOG: A Higher Order Logic Programming Language. SNeRG Technical Note 8, State University of New York at Buffalo, August 1981.

[Shapiro, 1979] Stuart C. Shapiro. The SNePS Semantic Network Processing System. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*. Academic Press, New York, 1979.

[Shapiro, 1993] Stuart C. Shapiro. Resolution refutation theorem prover. LISP code for theorem prover at (http://www.cse.buffalo.edu/faculty/shapiro/Reasoning/resolution.html), April 1993.

[Shapiro, 1995] Stuart C. Shapiro. Foundations of logic and inference. Slides of talk available at (http://www.cse.buffalo.edu/faculty/shapiro/Reasoning), August 1995. Tutorial SA1, 14th International Conference on Artificial Intelligence, Montreal, CA.

[Shum and Muntz, 1988a] Chung-Dak Shum and Richard Muntz. Implicit representation for extensional answers. In Larry Kerschberg, editor, *Proceedings from the Second International Conference on Expert Database Systems*, pages 497–522. Benjamin Cummings Publishing Company, 1988.

[Shum and Muntz, 1988b] Chung-Dak Shum and Richard Muntz. An information-theoretic study on aggregate responses. In *Proceedings of the 14th VLDB Conference*, pages 479–490, Los Angeles, CA, 1988.

[Simmons, 1965] Robert F. Simmons. Answering english questions by computer: a survey. *Communications of the ACM*, 8(1):53–70, January 1965.

[Simmons, 1970] Robert F. Simmons. Natural language question-answering systems: 1969. *Communications of the ACM*, 13(1):15–30, January 1970.

[Slagle, 1965] James R. Slagle. Experiments with a deductive question-answering program. *Communications of the ACM*, 8(12):792–798, December 1965.

[Srihari and Li, 2000] Rohini Srihari and Wei Li. Information extraction supported question answering. In E. M. Voorhees and D. K. Harman, editors, *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2000. (http://trec.nist.gov/pubs/trec8/t8_proceedings.html).

[Starr *et al.*, 1998] Barbara Starr, Vinay K. Chaudhri, Adam Farquhar, and Richard Waldinger. Knowledge-intensive query processing. In A. Borgida, V. Chaudhri, and Ms. Staudt, editors, *Proceedings of the 5th KRDB Workshop*, Seattle, WA, May 1998. (http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-10).

[Sterling and Shapiro, 1999] Leon Sterling and Ehud Shapiro. *The Art of Prolog*. MIT Press, third edition, 1999.

[Stickel, 1985] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Stickel, 1990] Mark E. Stickel. Rationale and methods for abductive reasoning in natural-language interpretation. In R. Studer, editor, *Natural Language and Logic*, volume 459 of *Lecture Notes in Artificial Ingelligence*, pages 233–252. Springer-Verlag, 1990.

[Vasey, 1986] Phil Vasey. Qualified answers and their application to transformation. In G. Goos and J. Hartmanis, editors, *Proceedings of the 3rd International Conference on Logic Programming*, number 225 in LNCS, pages 425–432, London, 1986. Springer-Verlag.

[Verberne *et al.*, 2000] A. Verberne, F. van Harmelen, and A. ten Teije. Anytime diagnostic reasoning using approximate boolean constraint propagation. In *Proceedings of the Seventh Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, Boulder, Colorado, April 2000.

[Voorhees and Harman, 2000] E. M. Voorhees and D. K. Harman, editors. *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC-8)*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2000. (http://trec.nist.gov/pubs/trec8/t8_proceedings.html).

[Voorhees and Harman, 2001] E. M. Voorhees and D. K. Harman, editors. *NIST Special Publication XXX-XXX: The Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2001. Pre-publication version of the proceedings located at (http://trec.nist.gov/pubs/trec9/t9_proceedings.html).

[Webber, 1986] Bonnie Lynn Webber. Questions, Answers and Responses: Interacting with Knowledge-Base Systems. In Michael L. Brodie and John Mylopoulos, editors, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, chapter 26, pages 365–402. Springer-Verlag, 1986.

[Weiner, 1980] J. L. Weiner. Blah, a system which explains its reasoning. *Artificial Intelligence*, 15:19–48, 1980.

[Wisniewski, 1999] Andrzej Wisniewski. Erotetic logic and explanation by abnormic hypotheses. *Synthese*, 120:295–309, 1999.

[Wolstenholme, 1988] Dave Wolstenholme. Saying ”i don’t know” and conditional answers. In *Research and Development in Expert Systems IV*, British Computer Society workshop series, pages 115–125. Cambridge University Press, Cambridge, 1988.

[Woods *et al.*, 2000] W. A. Woods, Stephen Green, and Paul Martin. Halfway to question answering. In E. M. Voorhees and D. K. Harman, editors, *NIST Special Publication XXX-XXX: The Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards and Technology (NIST), U.S. Department of Commerce, NIST, 2000. Pre-publication version of the proceedings located at (http://trec.nist.gov/pubs/trec9/t9_proceedings.html).

[Woods, 1991] W. A. Woods. Understanding Subsumption and Taxonomy: A Framework for Progress. In John Sowa, editor, *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann, 1991.

[Wos *et al.*, 1965] Lawrence Wos, George A. Robinson, and Daniel F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *Journal of the ACM*, 12(4):536–541, October 1965.