Available online at www.sciencedirect.com
ScienceDirect
Journal of Applied Logic 5 (2007) 70–91
JOURNAL OF APPLIED LOGIC
www.elsevier.com/locate/jal

Defining answer classes using resolution refutation

Debra T. Burhans [a,*], Stuart C. Shapiro [b]

[a] *Department of Computer Science, Canisius College, Buffalo, NY, USA*
[b] *Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA*

Available online 18 January 2006

Abstract

Resolution theorem proving provides a useful paradigm for the exploration of question answering. A partition of the clauses generated during resolution refutation based on their syntactic structure is presented. The three classes comprising this partition correspond to semantically intuitive types of answers. This work encompasses and expands upon previous work on question answering in a theorem proving paradigm, which began with the association of answers with proofs. A complete, formal definition of what is meant by *answer* in the context of resolution theorem proving is presented. In this context, 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 clause form of a 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. The three answer classes identified are: *specific*, *generic*, and *hypothetical*. 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 results presented provide a foundation for further work by establishing a context-independent logical pragmatics of question answering.

Keywords: Resolution; Automated deduction

1. Introduction

1.1. Theorem proving as question answering

This article 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. Although we have worked within the context of resolution theorem proving, the results have implications for all question answering systems. 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 by broadening the notion of answer.

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 [7,23,26,29,37]. Resolution plus

* Corresponding author.
E-mail addresses: burhansd@canisius.edu (D.T. Burhans), shapiro@cse.buffalo.edu (S.C. Shapiro).

1570-8683/$ – see front matter
doi:10.1016/j.jal.2005.12.004

factoring is refutation complete for first order predicate calculus (FOPC) [7,26]: 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 [19,21,27]. 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 [4,19–21,27–29,32,33,36,39]. For our purposes we will use the term "proof" to refer to the termination of resolution and the corresponding production of the empty clause (or, in case an answer literal is employed, the production of a clause containing only answer literals). We also assume the use of a set of support strategy, meaning that resolution must involve clauses that descend from the original negated question clauses.

Our view of resolution refutation is one in which every reasoning step produces an answer: 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 a (possibly incomplete) set of answers that are the resolvents generated as resolution proceeded. A proof does not necessarily signal the production of a complete set of answers, rather, it indicates the discovery of a particular type of answer. The notion of failure in terms of question answering is an empty set of relevant clauses: this is only possible when no clause in a knowledge base may be resolved with those derived from a question.

1.2. Motivation

It is clear that the centrality of question answering to the issue of intelligence makes it a worthwhile topic for any AI researcher. Theorem-proving approaches to question answering remain pervasive in the field: consider the widespread use of Prolog as one example. 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 valuable. Moreover, all resolution theorem proving systems are already generating these "answers".

This introduction has served as an overview of the work presented. In Section 2 a complete exploration of answer generation in a resolution theorem prover is provided. The answer classes *specific*, *generic*, and *hypothetical* are introduced and formally characterized. Section 3 presents the conclusions.

2. Resolvents as answers

2.1. 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 [10], the resulting clauses are added to a *consistent* set of clauses, and a refutation proof is initiated [7,23,26,37]. 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 [19]. 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. Given a ground query, the interpretation of the *failure* to find a proof is an answer of "no" when a closed world assumption is made [34], 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* [19–21,24,27–29,32,33,36,39] and *intensional* [3,8,9,16,18,22,30–33]. 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 the predicates in 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 [19–21]. 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 [27] extended Green's work on extensional answers to include answers containing universally quantified variables while still requiring a proof: their answers do not conform to the traditional notion of intensional answers because they do not have the form of a rule. 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 [11,12,17,40,45,47]. 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 [20]. Sergot recognized the utility of conditional answers in terms of providing follow-up questions to a user [40]. 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?".

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*. These answers may be viewed as "factoid" or "fact finding" in the context of more recent corpus-based work on question answering [13,43].
- *Generic answers*: Generic answers include and expand upon the category of *intensional answers*. We include Luckham and Nilsson's non-ground answers that are associated with proof 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 but may contain quantified variables.

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

2.2. Preliminaries

This section introduces basic terminology used in the subsequent discussion and formalizes our underlying assumptions about the way in which a resolution theorem prover is employed for the question answering task. We assume the use of an *answer literal* [21] that serves to identify clauses descended from the clause form of the original question. However, we employ a variant of the expanded notion of answer literal proposed by Shapiro [41] that replaces a question of the form $\exists\bar{x}\ \overline{P}(\bar{x})$ with $\forall\bar{x}\ (\overline{P}(\bar{x}) \Rightarrow ANSWER(\overline{P}(\bar{x})))$. 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 and that there are no limitations imposed on the form of a question, which may include existential and universal quantification. Our variation on this technique removes the quantifiers from the question prior to translation into clause for according to the following steps [5]:

(1) Put the proposition in prenex normal form so that it has the form $\overline{Q}\bar{x}\,\overline{P}(\bar{x})$, where $\overline{Q}$ are all of the quantifiers from the original question, $\bar{x}$ 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 $\overline{P}(\bar{x})$ with $[\overline{P}(\bar{x}) \Rightarrow ANSWER(\overline{P}(\bar{x}))]$.
(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 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 $(Px_1, x_2, \ldots, x_n)$ 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*. A clause contains an answer literal if and only if it is a question clause or is descended from a question clause. Note that step 3 above is where the answer predicate is introduced. We make use of answer literals in defining relevance:

Definition 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:

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

Note that $Th(\mathcal{KB})$ means the theory of $\mathcal{KB}$, namely, the set of all propositions implied by $\mathcal{KB}$.

Table 1
Questions, augmented negations, and question clauses

Question proposition	Augmented negation	Question clause
(*dog fido*)	[(*dog fido*) ⇒ (*ANSWER* (*dog fido*))]	{(¬(*dog fido*)), (*ANSWER* (*dog fido*))}
∃*x* (*dog x*)	∀*x* [(*dog x*) ⇒ (*ANSWER* (*dog x*))]	{(¬(*dog x*)), (*ANSWER* (*dog x*))}
∀*x* (*dog x*)	∃*x* [(*dog x*) ⇒ (*ANSWER* (*dog x*))]	{(¬(*dog S*10)), (*ANSWER* (*dog S*10))} [a]
∀*x* ∃*y* (*Rxy*)	∃*x* ∀*y* [(*Rxy*) ⇒ (*ANSWER* (*Rxy*))]	{(¬(*RS*513*x*)), (*ANSWER* (*RS*513*x*))} [b]
∃*x* ∀*y* (*xy*)	∀*x*∃*y*[(*Rxy*) ⇒ (*ANSWER* (*Rxy*))]	{(¬(*Rx*(*S*521*x*))), (*ANSWER* (*Rx*(*S*521*x*)))} [c]

[a] *S*10 is a Skolem constant.
[b] *S*513 is a Skolem constant.
[c] *S*521 is a unary Skolem function.

The import of this definition is that question clauses and their descendents are relevant to a question since they all contain at least one answer literal. Finally, we provide a broad definition of answer.

Definition 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: the degree to which such answers are helpful clearly depends on context. While such consideration are not included in this definition it does 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 as answers by a question answering system. We now examine the three answer classes in greater detail.

2.3. 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 representing 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 [27], 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. Specific answers are associated with a proof: it is the derivation of the empty clause, or in our case, of a clause containing only answer literals, that signals a *possible* specific answer. In order to qualify as a specific answer, there must be neither variables nor Skolem functions in the answer literals. Table 2 shows a simple knowledge base from Rich and Knight [36, p. 192] that is used to demonstrate specific answering. With respect to this KB, Rich and Knight [36, 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 3, where the question clause is numbered as clause 7 following the numbering above. Given the knowledge base shown in Table 2 and the question clause shown in Table 3, a refutation proof is initiated with steps shown in Table 4. Note that variables in clauses have been renamed after each resolution step.

Table 2
Simple knowledge base

Logic form	English gloss	Numbered 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 3
Question

Logic form	English gloss	Numbered question clause
$\exists x$ $(likestoeat\ Puss\ x)$	what does Puss like to eat	7.{(¬(LIKESTOEAT PUSS ?46)), (ANSWER (LIKESTOEAT PUSS ?46))}

Table 4
Specific answers: complete refutation proof example showing two different proofs

Numbered 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/[a] {?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))}	R 11,4/ {CHARLIE/?523}
QED	
13.{(ANSWER (LIKESTOEAT PUSS HERB))}	R 11,5/ {HERB/?523}
QED	

[a] R stands for "resolve", and the numbers following R are the numbers of the parent clauses.

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

(*LIKESTOEAT PUSS CHARLIE*)

(*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 satisfying individual. In this example, we have two specific answers, both of which entail the original question:

$$(\textit{LIKESTOEAT PUSS CHARLIE}) \Rightarrow \exists x\ (\textit{LIKESTOEAT PUSS}\ x)$$

$$(\textit{LIKESTOEAT PUSS HERB}) \Rightarrow \exists x\ (\textit{LIKESTOEAT PUSS}\ x)$$

Because specific answers are associated with the existence of a 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 clearly incorrect based on the results obtained by Luckham and Nilsson [27], who also associate the presence of *non-ground* answer literals with a proof. The following definition clarifies the form of clauses that are related to specific answers:

Definition 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 Skolem constants present.

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

Definition 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 semantics, 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. A problem noted by Webber [46, p. 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 ¬**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 [2] as a "completeness-claim". This shortcoming of specific answers in some situations points to the need to consider other types of answers.

2.4. Generic answers

The foregoing definition of specific answers fails to account for all clause forms. 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 represented 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 a proof because they contain non-answer literals. The question is whether all such clauses represent generic answers, and, if not, what distinguishes those clauses that do. In addition, we show that there are some clauses associated with generic answers that *are* associated with a proof.

In his description of *intensional answering*, Imielinski [22, p. 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) [22,31,33]. These sets are generally small relative to the size of the $\mathcal{EDB}$ (extensional database), which may contain a huge number of facts [22,33].

Cholvy and Demolombe [8] define intensional answers as those that provide sufficient conditions for tuples to satisfy a query. Their work was done in the context of theorem proving and focused on clauses that can be written in rule form. The antecedent of a rule can be seen as specifying sufficient 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 build upon this work by showing that not all generic answers are rules, and not all rules are generic answers.

We defer a formal definition of generic answer until some examples have been examined. To illustrate generic answering, consider once again the example from Rich and Knight [36] from the previous section. Clauses generated during the proof, excepting those associated with specific answers, namely, 12 and 13, are shown in Table 5.

These clauses contain answer literals, so they are descendents of the question clauses and are therefore relevant to the question by our previous definition. 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. 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.

Clauses 10 and 11 are descendents of the question clauses and are not syntactically subsumed by other clauses in Table 5. Subsumption is often considered a taxonomic phenomenon, relying on semantic relations among objects [48], but in our discussions we use subsumption to refer strictly to considerations of syntax. Taxonomic subsumption is clearly, however, an important consideration in the overall question answering endeavor.

Table 5
Generic answers and resolution

Numbered 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}

Table 6
Clauses as rules

Numbered 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)

Table 7
Rules as answers

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

Table 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 6. It is generally accepted that such rules should be written as closed sentences, where all of the variables are universally quantified [8,9,22,30]. 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 7. 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 [8] definition of generic answers as specifying "sufficient" conditions for specific answer tuples.

2.4.1. 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 [32] 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 [32, pp. 184–185]. Table 8 shows the initial knowledge base containing information about parents and children: Nilsson [32, p. 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 9, where the question clause is numbered as clause 3 following the numbering above. A refutation proof that answers this question with respect to the "Parent and Child" knowledge base is shown in Table 10.

In the example shown, "S24" is a Skolem constant. Luckham and Nilsson [27] offer a proof that Skolem constants and functions in an answer statement (corresponding to the answer literal) can be replaced by universally quantified variables and extend this result to any clause related to the answer statement. The important point is that this replace-

Table 8
Parent and child

Logic form	English gloss	Numbered clause
$\forall x\ (C(x, p(x)))$	for all x, x is the child of p(x)	1.{(C ?4 (PARENT-OF ?4))}[a]
$\forall x\ \forall y\ ((C(x, y)) \Rightarrow (P(y, x)))$	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)}

[a] 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 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 10
Generic answers: refutation proof example

Numbered 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))}	R 4,1/ {(PARENT-OF S24) /?60, S24/?4}
QED	

ment does not affect the reasoning process. This is achieved by Luckham and Nilsson [27] 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: they do not affect the reasoning process. The answer clause from Table 10 can be expressed replacing the Skolem constants with universally quantified variables:

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

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 there are no non-answer literals in the clause, we term *unconstrained generic answers*. As previously discussed, the antecedent of a generic answer, when present, describes sufficient conditions for individuals to be considered as answers to a question. This non-rule for of a generic answer describes properties of all individuals: there are no constraints regarding their properties. Variables occurring *only* in answer literals may be instantiated by any individual in the domain.

2.4.2. 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 2. Note that

Table 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)

"Boots" is not mentioned in the knowledge base: we have no idea what "Boots" is other than some individual. If we add the new information to the knowledge base and proceed with resolution refutation a number of clauses are generated, two of which are shown in Table 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 particular category. 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 2.5, we will see that these rules are *hypothetical answers*.

2.4.3. 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 [14,15,42]. 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 [22] 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, rather, what we see is variable sharing as a means to connect literals to one another.

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 represent 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 [1, 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 non-answer literals that contain variables, not all of which overlap the variables in the answer literals. 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 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 12. This 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 criteria 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 5 *(Variable sharing set of a literal).* The variable-sharing set of a literal l with respect to a set of literals $\mathcal{C}$, is a set of literals $\mathcal{VS}(l)$ such that each literal in $\mathcal{VS}(l)$ is an element of $\mathcal{C}$ 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 6 *(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 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 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:

$$\begin{aligned}
&\mathcal{CVS}_0 = \mathcal{VS}(\mathcal{L})\\
&\mathcal{CVS}_1 = \mathcal{VS}(\mathcal{CVS}_0)\\
&\vdots\\
&\mathcal{CVS}_n = \mathcal{VS}(\mathcal{CVS}_{n-1})\\
&\mathcal{CVS}_{n+1} = \mathcal{CVS}_n
\end{aligned}$$

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 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 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 something that loves something?* written in FOPC as $\exists x \exists y (loves\ x\,y)$, what answer is found? The clause representations 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 which is $\forall x\ \forall y\ [((green\ x) \wedge (blue\ y)) \Rightarrow (loves\ x\,y)]$. In other words, if x is green and y is blue then x loves y. While two variables appear in the answer literal, ?54 and ?56, the non-answer literals (¬(*green* ?54)) and (¬(*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))}

The answer is simply a reiteration of the rule that anyone who loves everyone. In this case, the variables ?56 and ?58 appear in the answer literal, but only ?56 appears in the non-answer literal. The "everyone" corresponds to the unconstrained generic portion of the answer, namely, the variable ?58.

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:

$$\text{Question: } \exists x\ (\textit{valuable}\ x)$$

$$\text{Answer: } \forall(x\,y\,z\,k\,l)$$
$$[((\textit{item}\ x) \wedge (\textit{cabinet}\ y) \wedge (\textit{senior-manager}\ z)$$
$$\wedge (\textit{key}\ k) \wedge (\textit{lock}\ l) \wedge (\textit{locks}\ l\,y)$$
$$\wedge (\textit{key-to}\ k\,l) \wedge (\textit{held-by}\ k\,z) \wedge (\textit{in}\ x\,y))$$
$$\Longrightarrow (\textit{valuable}\ x)]$$

The literals in the antecedent give a set of sufficient conditions for an object to be considered valuable and are connected to the consequent by the closure of variable sharing property.

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

Definition 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 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 describing hypothetical answers.

Definition 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 the arguments of all 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*.

2.4.4. Discussion of generic answers

Generic answers *describe* satisfying individuals, 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 [44]) 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 [46].

In addition to potential gains in computational efficiency 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 [38]. 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. 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 knowledge representation 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 [8] 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. In combination with specific answers they comprise the set of 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. We previously 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*.

2.5. Hypothetical answers

We have included all answers associated with proof in the categories specific and generic. All that are left are clauses containing both answer and non-answer literals, so hypothetical answers will all have the form of rules. Specifically, 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*. We refer to the condition as the *hypothetical component* of the answer, and the rest of the answer we term 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* [11] in terms of logic databases. According to his 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 intensional database, or rules, and Extensional Conditional Answers are derived from the extensional database, or facts. 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 [12, 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.

To illustrate hypothetical answers, we use an example from the previous sections. First, the KB to be considered is that presented in Table 2 (from [36, 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 13 shows some of the clauses that result. English glosses for the rules in Table 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.

Table 13
Hypothetical answers

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

(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 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, however, this information is not contained in the knowledge base, thus, it would be detrimental to deny "answer" status to an answer with hypothetical component *Boots is a calico* simply because we know that 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. Recent work in question answering has focused on the forms of "wh" questions in order to divide them into different classes [25] in terms of the types of individuals that might satisfy the questions.

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.

2.5.1. 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 14
Knowledge base for hypothetical answer example

Logic form	English gloss	Numbered clause
$\forall x\ [(American\ x) \Rightarrow (consumer\ x)]$	All Americans are consumers	1.{(¬(AMERICAN ?11)), (CONSUMER ?11)}
$(\forall x\ (\forall y\ [((consumer\ x) \wedge (bargain\ y)) \Rightarrow (likes\ x\ y)]))$	All consumers like bargains	2.{(¬(CONSUMER ?5)), (¬(BARGAIN ?7)), (LIKES ?5 ?7)}
$(\forall x\ (\forall y\ [((consumer\ x) \wedge (junk\ y)) \Rightarrow ((\neg(likes\ x\ y))]))$	All consumers don't like junk	3.{(¬(CONSUMER ?12)), (¬(JUNK ?14)), (¬((LIKES ?12 ?14))}
$(American\ Muffy)$	Muffy is an American	4.{(AMERICAN MUFFY)}
$(bargain\ cowPitcher)$	The cow pitcher is a bargain	5.{(BARGAIN COWPITCHER)}
$(bargain\ cowPitcher) \vee (junk\ cowPitcher)$	The cow pitcher is a bargain or the cow pitcher is junk	6.{(BARGAIN COWPITCHER), (JUNK COWPITCHER)}

Table 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 some of the answers in rule form along with English glosses:

1. $(LIKES\ MUFFY\ COWCREAMER)$
   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 first of these answers (interpreting x as "someone") says that you can take anyone, if the person is a consumer and likes a particular cow pitcher then Muffy likes it too. The second says that if there is someone who is a consumer and who likes a particular cow pitcher then Muffy likes it too.

In terms of the form of these answers, 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.

We now define what it means for a clause to be considered a hypothetical answer clause.

Definition 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 13) is:

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

Definition 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 and wide scope over the existentials. 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 13) is:

(*CAT BOOTS*) ⇒ (*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*.

2.5.2. 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 [35, p. 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 [11], 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.

2.5.3. 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 new question for which the reasoner may have or be able to find an 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 $\mathcal{H}$. 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 $\mathcal{A}$. 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 set of properties sufficient to determine that some entity is the member of the class of objects described by the predicates in the question. 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 individual for a question nor a general description of a satisfying individual, but they do provide information about what needs to be known in order for an answer to be forthcoming.

There are two situations under which hypothetical answer arise. In the examples shown earlier in this section, it is seen that hypothetical answers arise 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 other 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 $\forall x\ (Px)$, then (Pa) can be inferred for any individual.

Hypothetical answers will be present when there is *disjunctive* information in a knowledge base. When neither disjunct is known to be true or known to be false, yet the disjunction is true, this 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 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 [6]. 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.

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 yet provide new questions that may ultimately lead a questioner to an answer.

2.6. A complete characterization of answers in resolution refutation

The following formula provides a complete characterization of answers in a resolution theorem prover in accordance with our previous discussion:

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

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 Section 2.5 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 the arguments of the 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 component, and a question component that contains one or more variables indicates a generic answer component. 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 [20,21], Luckham and Nilsson [27], Prolog, Otter [29], 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.

2.6.1. The hypothetical component

The existentially quantified antecedent of the formula above is the *hypothetical component*, and has the 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 [11,12] 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.

2.6.2. The generic component

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

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

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 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* [31].

2.6.3. The question component

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

- $\mathcal{Q}$ stands for the predicates in the question component.
- The z_i are variables contained only in the question component. When present they signal an unconstrained 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.

3. Conclusions

We have presented a characterization of resolvents generated in the course of a resolution refutation proof that is based purely on the syntactic form of clauses, yet reflects an intuitive semantics about different types of answers. When the idea of what counts as an answer is expanded to include all information relevant to a question, regardless of the reasoning system employed for question answering, it increases the chances that important and relevant information is provided to a questioner rather than being overlooked.

In our discussions of the meaning of specific, generic, and hypothetical answers we have clearly demonstrated their importance and applicability to question answering. We have provided a detailed characterization of each of the three answer classes, and in doing so have accounted for and moved beyond previous work using the paradigm of resolution for question answering. We have presented a canonical form for answers that unites the three answer types in a three-part implication.

The results presented in this article provide a basis for future work in the logic of questions and answers.

References

[1] A.R. Anderson, N.D. Belnap Jr., Entailment: The Logic of Relevance and Necessity, Princeton University Press, Princeton, NJ, 1975.
[2] N.D. Belnap, T.B. Steel, The Logic of Questions and Answers, Yale University Press, New Haven, CT, 1976.
[3] A. Borgida, D.L. McGuinness, Asking queries about frames, in: Proceedings of KR-96, Cambridge, MA, Morgan Kaufmann, 1996, pp. 340–349.
[4] R.J. Brachman, H.J. Levesque, Knowledge Representation and Reasoning, Morgan Kaufmann, San Francisco, CA, 2004.
[5] D.T. Burhans, A question answering interpretation of resolution refutation, PhD Dissertation, Technical Report 2002-03, Department of Computer Science, State University of New York at Buffalo, 2002.
[6] H. Chalupsky, T.A. Russ, WhyNot: debugging failed queries in large knowledge bases, in: Proceedings of the 18th National Conference on Artificial Intelligence and the 14th Conference on Innovative Applications of AI, AAAI Press/MIT Press, 2002, pp. 870–877.
[7] C.-L. Chang, R.G.-T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1973.
[8] L. Cholvy, R. Demolombe, Querying a rule base, in: Proceedings of the First International Workshop on Expert Database Systems, 1986, pp. 365–371.
[9] L. Cholvy, Answering queries addressed to a rule base, Revue d'Intelligence Artificielle 4 (1) (1990).
[10] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (3) (1960) 201–215.
[11] R. Demolombe, A strategy for the computation of conditional answers, in: B. Neumann (Ed.), Proceedings of the 10th European Conference on Artificial Intelligence, ECAI 92, Vienna, John Wiley and Sons, 1992, pp. 134–138.
[12] R. Demolombe, Uncertainty in intelligent databases, in: A. Motro, P. Smets (Eds.), Uncertainty Management in Information Systems, Kluwer Academic Publishers, Dordrecht, 1997, pp. 89–126.
[13] A.R. Diekema, O. Yilmazel, J. Chen, S. Harwell, L. He, E.D. Liddy, Finding answers to complex questions, in: M.T. Maybury (Ed.), New Directions in Question Answering, AAAI Press/MIT Press, Menlo Park, CA, 2004, pp. 141–152, Chapter 11.
[14] K. Fine, In defense of arbitrary objects, in: Proceedings of the Aristotelian Society, vol. suppl. LVII, Aristotelian Society, 1983, pp. 55–77.
[15] K. Fine, Natural deduction and arbitrary objects, J. Philos. Logic 14 (1985) 57–107.
[16] T. Gaasterland, P. Godfrey, J. Minker, An overview of cooperative answering, in: R. Demolombe, T. Imielinski (Eds.), Nonstandard Queries and Nonstandard Answers, in: Studies in Logic and Computation, Clarendon Press, Oxford, 1994, pp. 1–40, Chapter 1.
[17] G.N. Gilbert, Question and answer types, in: Research and Development in Expert Systems IV, in: British Computer Society Workshop Series, Cambridge University Press, Cambridge, 1988, pp. 162–172.
[18] P. Godfrey, J. Gryz, Intensional query optimization, Technical Report UMIACS TR 96-72, University of Maryland at College Park, College Park, MD, October 1996.

[19] C.C. Green, B. Raphael, The use of theorem-proving techniques in question-answering systems, in: Sr. Richard B. Blue (Ed.), Proceedings of 23rd National Conference, Princeton, NJ, Association for Computing Machinery, Brandon/Systems Press, Inc., 1968, pp. 169–181.
[20] C. Green, Applications of theorem proving to problem solving, in: D.E. Walker, L.M. Norton (Eds.), Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1969, pp. 219–239.
[21] C. Green, Theorem-proving by resolution as a basis for question-answering systems, in: D. Michie, B. Melzer (Eds.), Machine Intelligence 4, Edinburgh University Press, Edinburgh, 1969, pp. 183–205.
[22] T. Imielinski, Intelligent query answering in rule based systems, J. Logic Programming 4 (3) (1987) 229–257.
[23] R. Kowalski, Logic for Problem Solving, Artificial Intelligence Series, North-Holland, New York, 1979.
[24] 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.
[25] M. Light, A. Ittycheriah, A. Latto, N. McCracken, Reuse in question answering: a preliminary study, in: M.T. Maybury (Ed.), New Directions in Question Answering, AAAI Press/MIT Press, Menlo Park, CA, 2004, pp. 169–181, Chapter 13.
[26] D.W. Loveland, Automated Theorem Proving: A Logical Basis, Fundamental Studies in Computer Science, North-Holland, Amsterdam, 1978.
[27] D. Luckham, N.J. Nilsson, Extracting information from resolution proof trees, Artificial Intelligence 2 (1971) 27–54.
[28] G.F. Luger, Artificial Intelligence, fifth ed., Pearson Education Limited (Addison-Wesley), Harlow, Essex, England, 2005.
[29] W.W. McCune, Otter 3.0 Reference Manual and Guide, Argonne National Laboratories, 1994.
[30] A. Motro, Using integrity constraints to provide intensional answers to relational queries, in: Proceedings of VLDB89, the 15th International Conference on Very Large Databases, Amsterdam, The Netherlands, August 1989, pp. 237–246.
[31] A. Motro, Intensional answers to database queries, IEEE Trans. Knowledge Data Engrg. 6 (3) (1994) 444–454.
[32] N.J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing Company, Palo Alto, CA, 1980.
[33] R. Reiter, Deductive question-answering in relational data bases, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, New York, 1978, pp. 149–177.
[34] R. Reiter, On closed world data bases, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, New York, 1978, pp. 55–76.
[35] R. Reiter, J. de Kleer, Foundations of assumption-based truth maintenance systems, in: Proceedings of the 6th National Conference on Artificial Intelligence, AAAI-87, vol. 1, MIT Press, Cambridge, MA, 1987, pp. 183–188.
[36] E. Rich, K. Knight, Artificial Intelligence, second ed., McGraw-Hill, New York, 1991.
[37] J.A. Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1965) 23–41.
[38] E. Rosch, C.B. Mervis, Family resemblances: studies in the internal structure of categories, Cognitive Psychol. 4 (1975) 573–605.
[39] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, fifth ed., Prentice-Hall, Upper Saddle River, NJ, 2003.
[40] M. Sergot, A query-the-user facility for logic programming, in: P. Degano, E. Sandewall (Eds.), Integrated Interactive Computer Systems, North Holland, Amsterdam, 1983, pp. 27–41.
[41] S.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.
[42] S.C. Shapiro, A logic of arbitrary and indefinite objects, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), AAAI Press, 2004, pp. 565–575.
[43] S. Small, T. Strzalkowski, T. Liu, N. Shimizu, B. Yamrom, A data driven approach to interactive QA, in: M.T. Maybury (Ed.), New Directions in Question Answering, AAAI Press/MIT Press, Menlo Park, CA, 2004, pp. 129–140, Chapter 10.
[44] M.E. Stickel, Automated deduction by theory resolution, J. Automat. Reason. 1 (1985) 333–355.
[45] P. Vasey, Qualified answers and their application to transformation, in: G. Goos, J. Hartmanis (Eds.), Proceedings of the 3rd International Conference on Logic Programming, London, in: Lecture Notes in Comput. Sci., vol. 225, Springer-Verlag, Berlin, 1986, pp. 425–432.
[46] B.L. Webber, Questions, answers and responses: interacting with knowledge-base systems, in: M.L. Brodie, J. Mylopoulos (Eds.), On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies, Springer-Verlag, Berlin, 1986, pp. 365–402, Chapter 26.
[47] D. Wolstenholme, Saying 'I don't know' and conditional answers, in: Research and Development in Expert Systems IV, in: British Computer Society Workshop Series, Cambridge University Press, Cambridge, 1988, pp. 115–125.
[48] W.A. Woods, Understanding subsumption and taxonomy: a framework for progress, in: J. Sowa (Ed.), Principles of Semantic Networks, Morgan Kaufmann, 1991, pp. 45–94.