Question Answering: From Partitions to Prolog*

Balder ten Cate[1] and Chung-chieh Shan[2]

[1] Institute for Logic, Language and Computation, Universiteit van Amsterdam
Nieuwe Doelenstraat 15, 1012 CP, Amsterdam, The Netherlands
b.ten.cate@hum.uva.nl

[2] Division of Applied and Engineering Sciences, Harvard University
33 Oxford Street, Cambridge MA 02138, USA
ccshan@post.harvard.edu

Abstract. We implement Groenendijk and Stokhof's partition semantics of questions in a simple question answering algorithm. The algorithm is sound, complete, and based on tableau theorem proving. The algorithm relies on a syntactic characterization of answerhood: Any answer to a question is equivalent to some formula built up only from instances of the question. We prove this characterization by translating the logic of interrogation to classical predicate logic and applying Craig's interpolation theorem.

1 The Partition Theory of Questions

An elegant account of the semantics of natural language questions from a logical and mathematical perspective is the one provided by Groenendijk and Stokhof [8]. According to them, a question denotes a partition of a logical space of possibilities. In this section, we give a brief summary of this influential theory, using a notation slightly different from Groenendijk's presentation [7].

A question is essentially a first order formula, possibly with free variables. We will denote a question by $?\phi$, where ϕ is a first order formula. (We will also denote a set of questions by $?\Phi$, where Φ is a set of first order formulas.) An answer is also a first order formula, but one that stands in a certain *answerhood* relation with respect to the question, to be spelled out in Sect. 2. For example, the statement "Everyone is going to the party" ($\forall x Px$) will turn out to be an answer to the question "Who is going to the party?" ($?Px$).

We assume that equality is in the language, so one can ask questions such as "Who is John?" ($?x \approx j$). We also assume that, for every function symbol—including constants—it is indicated whether it is interpreted rigidly or not. Intuitively, for a function symbol to be rigid means that its denotation is known.

* This paper will also appear in the *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'02)*. We would like to thank Patrick Blackburn, Paul Dekker, Jeroen Groenendijk, Maarten Marx, Stuart Shieber, and the anonymous referees for their useful comments and discussions. The 13th European Summer School in Logic, Language and Information, the 13th Amsterdam Colloquium, and Stanford University's Center for the Study of Language and Information provided stimulating environments that led to this collaboration. The second author is supported by the United States National Science Foundation under Grant IRI-9712068.

For example, under the notion of answerhood that we will introduce in Sect. 2, it is only appropriate to answer "Who is going to the party?" ($?Px$) with "John is going to the party" (Pj) if it is known who "John" is—in other words, if j is rigid. Also, for "Who is John?" ($?x \approx j$) to be a non-trivial question, "John" must have a non-rigid interpretation.

Questions are interpreted relative to first order modal structures with constant domain. That is, a model is of the form (W, D, I), where W is a set of worlds, D is a domain of entities, and I is an interpretation function assigning extensions to the predicates and function symbols, relative to each world. Furthermore, we only consider models that give rigid function symbols the same extension in every world. Relative to such a model $M = (W, D, I)$, a question $?\phi$ expresses a partition of W, in other words an equivalence relation over W:

$$[?\phi]_M = \{\, (w, v) \in W^2 \mid \forall g\colon M, w, g \models \phi \Leftrightarrow M, v, g \models \phi \,\} \ . \tag{1}$$

Roughly speaking, two worlds are equivalent if one cannot tell them apart by asking the question $?\phi$. In general, any set of questions $?\Phi$ also expresses a partition of W, namely the intersection of the partitions expressed by its elements:

$$\begin{aligned} [?\Phi]_M &= \textstyle\bigcap_{\phi \in \Phi} [?\phi]_M \\ &= \{\, (w, v) \in W^2 \mid \forall \phi \in \Phi\colon \forall g\colon M, w, g \models \phi \Leftrightarrow M, v, g \models \phi \,\} \ . \end{aligned} \tag{2}$$

Entailment between questions is defined as a *refinement* relation among partitions (i.e., equivalence relations): An equivalence relation A is a subset of another equivalence relation B if every equivalence class of A is contained in a class of B.

$$?\Phi \models ?\psi \quad \text{iff} \quad \forall M\colon [?\Phi]_M \subseteq [?\psi]_M \ . \tag{3}$$

A more fine-grained notion of entailment is as follows [7]. Let χ be a first order formula with no free variables, and let $M \models \chi$ mean that $M, w \models \chi$ for all w.

$$?\Phi \models_\chi ?\psi \quad \text{iff} \quad \forall M\colon M \models \chi \Rightarrow [?\Phi]_M \subseteq [?\psi]_M \ . \tag{4}$$

Pronunciation: The questions $?\Phi$ entail the question $?\psi$ *in the context of* χ (or, *given* χ). The context χ is intended to capture assertions in the common ground: If it is commonly known that everyone who got invited to the party is going, and vice versa ($\forall x(Ix \leftrightarrow Px)$), then the questions "Who got invited?" ($?Ix$) and "Who is going?" ($?Px$) entail each other.

1.1 Translation to First Order Logic

Groenendijk and Stokhof [8] do not provide an inference system for the above entailment relation, but we can use the following entailment-preserving translation procedure to ordinary first order logic.

The intuition behind the translation is simple: One question entails another iff the former distinguishes between more worlds than the latter does. In other words, one question entails another iff every pair of worlds considered equivalent by the first question is also considered equivalent by the second. For instance, the question "Who is going to the party?" ($?Px$) entails the question "Is John going to the party?" ($?Pj$). After all, if exactly the same people are going to

two parties ($\forall x(Px \leftrightarrow P'x)$), then either John is going to both parties, or he is going to neither of them ($Pj \leftrightarrow P'j$).

For any first order formula ϕ, let ϕ^* be the result of priming all occurrences of non-rigid non-logical symbols. Formally, define

$$\begin{aligned} (Pt_1 \dots t_n)^* &= P't_1^* \dots t_n^* & f(t_1, \dots, t_n)^* &= \begin{cases} f(t_1^*, \dots, t_n^*) & \text{if } f \text{ is rigid} \\ f'(t_1^*, \dots, t_n^*) & \text{otherwise} \end{cases} \\ (s \approx t)^* &= (s^* \approx t^*) \\ (\phi \wedge \psi)^* &= \phi^* \wedge \psi^* & x^* &= x \\ (\neg\phi)^* &= \neg(\phi^*) & \top^* &= \top \\ (\exists x\phi)^* &= \exists x(\phi^*) & \bot^* &= \bot \ . \end{aligned} \tag{5}$$

Furthermore, for any question $?\phi$, let $?\phi^\#$ be the first order formula $\forall\vec{x}(\phi \leftrightarrow \phi^*)$, where $\vec{x}$ are the free variables of ϕ. For any set of questions $?\Phi$, let $?\Phi^\#$ be the set of first order formulas $\{\, ?\phi^\# \mid \phi \in \Phi \,\}$. Now, we can reduce entailment between questions to ordinary first order entailment, as follows.

Theorem 1. *The following entailments are equivalent.*

1. $?\Phi \models_\chi ?\psi$
2. $?\Phi^\#, \chi, \chi^* \models ?\psi^\#$

Proof. [$\Rightarrow$] By contraposition. Suppose $?\Phi^\#, \chi, \chi^* \not\models ?\psi^\#$. Then there is a first order model $M = (D, I)$ that verifies the formulas $?\Phi^\#, \chi, \chi^*$ but not $?\psi^\#$.

Now consider the first order modal structure $N = (\{w, v\}, D, I')$, where for all non-logical symbols α, we let $I'_w(\alpha) = I(\alpha)$ and $I'_v(\alpha) = I(\alpha')$. By construction, $(w, v) \in [?\Phi]_N$ and $(w, v) \notin [?\psi]_N$. Furthermore, $N \models \chi$. Therefore, $?\Phi \not\models_\chi ?\psi$.

[$\Leftarrow$] Again by contraposition. Suppose $?\Phi \not\models_\chi ?\psi$. Then there is a first order modal structure $M = (W, D, I)$ with $w, v \in W$ such that $(w, v) \in [?\Phi]_M$ yet $(w, v) \notin [?\psi]_M$. Furthermore, $M, w \models \chi$ and $M, v \models \chi$.

Now consider the first order model $N = (D, I')$, where for all non-logical symbols α, we let $I'(\alpha) = I_w(\alpha)$ and $I'(\alpha') = I_v(\alpha)$. By construction, N verifies the formulas $?\Phi^\#, \chi, \chi^*$ but not $?\psi^\#$. Therefore, $?\Phi^\#, \chi, \chi^* \not\models ?\psi^\#$. □

By this theorem, we can determine whether an entailment between questions holds, by translating the formulas involved to ordinary predicate logic. For example, suppose that a is a rigid constant, and consider the supposed entailment

$$?Px \models ?Pa \ . \tag{6}$$

In order to determine whether the entailment is valid, we need only translate the two formulas to ordinary first order logic, giving

$$\forall x(Px \leftrightarrow P'x) \models Pa \leftrightarrow P'a \ . \tag{7}$$

As can be easily seen, this is indeed valid. On the other hand, if a were non-rigid then the entailment would not be valid. As a second example, consider

$$?Px \models_{\forall x(Px \leftrightarrow Qx)} ?Qa \ . \tag{8}$$

Given that a is rigid, this is again valid, as we can see from the translation

$$\forall x(Px \leftrightarrow P'x), \forall x(Px \leftrightarrow Qx), \forall x(P'x \leftrightarrow Q'x) \models Qa \leftrightarrow Q'a \ . \tag{9}$$

1.2 Varying Domains

The partition theory as formulated in the previous section makes some natural predictions regarding question entailment. For example, the question "Who is going to the party?" ($?Px$) entails the question "Is everyone going to the party?" ($?\forall xPx$). Intuitively, whenever one knows (completely) who is going to the party, one also knows whether everyone is going to the party. As a matter of fact, much of the motivation for the partition theory of questions comes from its natural account of entailment relations between sentences embedding such questions.[3]

Unfortunately, the theory also makes some counterintuitive predictions. For example, the question $?j \approx b$ ("Is John the same person as Bill?", where j and b are interpreted rigidly) is entailed by every question, including the trivial question $?\top$. Likewise, $?\exists x\exists y\neg(x \approx y)$ ("Does there exist more than one entity?") is entailed by every question.[4] For this reason, an alternative semantics has been discussed, in which the questions are interpreted with respect to first order modal structures with *varying* domains [9]. That is, each world w is associated with its own domain of entities D_w. This semantics is more general, since it allows more models: Every constant domain model is also a varying domain model, so fewer entailments between questions are valid for varying domains. In particular, $?\exists x\exists y\neg(x \approx y)$ is no longer entailed by every question. Also, $?Px$ no longer entails $?\neg Px$, though the entailment $?Px, ?x \approx x \models ?\neg Px$ remains valid.

It is not entirely trivial to generalize the partition semantics of questions to varying domains, since it is unclear in (1) which assignments g should be quantified over. One generalization, which Groenendijk and Stokhof [9] seem to suggest but do not state, is to define

$$\begin{aligned}[?\phi]_M = \{\, (w,v) \in W^2 \mid \forall g\colon \\ (g \in D_w^\phi \ \&\ M,w,g \models \phi) \Leftrightarrow (g \in D_v^\phi \ \&\ M,v,g \models \phi)\,\}\ ,\end{aligned} \tag{10}$$

where D_w^ϕ denotes the set of assignment functions that map all free variables in ϕ to entities in D_w. Question entailment, as defined in (4), remains the same.

Various generalizations of the partition theory to varying domains, including that in (10), can be straightforwardly reduced to the constant domain version by introducing an *existence predicate* E. In the case of (10), one would *relativize* all quantifiers—that is, replace $\exists x\phi$ by $\exists x(Ex \wedge \phi)$ and $\forall x\phi$ by $\forall x(Ex \rightarrow \phi)$—and consider questions of the form $?(Ex_1 \wedge \ldots \wedge Ex_n \wedge \phi)$, where $x_1, \ldots, x_n$ are the free variables of ϕ. Theorem 1 can then be applied after the reduction.

In the rest of this paper, we will restrict ourselves to constant domain models.

2 A Syntactic Characterization of Answerhood

Groenendijk and Stokhof's entailment relation between questions, as discussed in the previous section, allows us to define a notion of answerhood.

[3] Cf. Nelken and Francez [13] for one of the few competing theories in this respect.

[4] In Sect. 2, we will see exactly which such counterintuitive predictions are made by the theory. For now, the reader can use Theorem 1 to check that the theory indeed makes these predictions.

Definition 1 (Answerhood). *Let $?\phi$ be a question and ψ a first order formula without free variables. We say that ψ is an* answer *to $?\phi$ if $?\phi \models ?\psi$.*

According to this notion (also termed *licensing* by Groenendijk [7]), "Everyone is going to the party" ($\forall x Px$) is an answer to "Who is going to the party?" ($?Px$), because $?Px \models ?\forall x Px$. (We will generalize to sets of questions in Sect. 2.2.)

Note that, under this definition, any contradiction or tautology counts as an answer to any question. Groenendijk and Stokhof [7, 8] define a stricter notion of *pertinence*, which excludes these two trivial cases by formalizing Grice's Maxims of Quality and Quantity, respectively. In this paper, however, we will stick to the simpler criterion of answerhood as defined above, which corresponds to Grice's Maxim of Relation.

We now have a semantic notion of answerhood, telling us what counts as an answer to a question. However, for practical purposes, it is useful to have also a *syntactic* characterization of this notion. Can one give a simple syntactic property that is a necessary and sufficient condition for answerhood? As we will see in a minute, one can.

First, let us look at a partial result discussed by Groenendijk and Stokhof [8] and Kager [11]. Define rigidity of terms and formula instances as follows.

Definition 2 (Rigidity). *A term is* rigid *if it is composed of variables and rigid function symbols. A formula ϕ is a* rigid instance *of another formula ψ if ϕ can be obtained from ψ by uniformly substituting rigid terms for variables. An identity statement $s \approx t$ is* rigid *if the terms s and t are rigid.*

For example, if c is a rigid constant and f is a rigid function symbol, then rigid instances of Px include Pc, Px, $Pf(c)$, and $Pf(x)$. The identity statement $c \approx x$ is also rigid. Notice that rigid instances are not necessarily rigid: If c is rigid, then Rcd is a rigid instance of Rxd even if the constant d is not rigid.

Groenendijk and Stokhof [8] and Kager [11] observed that rigid instances of a question constitute answers to that question. By a simple inductive argument, one can generalize this a bit.

Definition 3 (Development). *A formula ψ is a* development *of another formula ϕ (written "$\phi \leq \psi$") if ψ is built up from rigid instances of ϕ and rigid identity statements using boolean connectives and quantifiers, or if ψ is $\top$ or $\bot$.*

For example, if c and d are rigid constants, then $Pc \wedge Pd$ and $\exists x(Px \wedge \neg(x \approx c))$ are both developments of Px.

Theorem 2. *If $\phi \leq \psi$ then $?\phi \models ?\psi$.*

Proof. By induction on the size of ϕ. □

Using the translation procedure in Sect. 1, together with Craig's interpolation theorem for first order logic, we can prove the converse as well.

Theorem 3. *Let $\vec{y}$ be the free variables of some formula ψ. If $?\phi \models_\chi ?\psi$, then there exists some formula ϑ with no free variables beside $\vec{y}$ such that $\phi \leq \vartheta$ and $\chi \models \forall \vec{y}(\psi \leftrightarrow \vartheta)$.*

Proof. First, we will prove the special case where ϕ is an atomic formula, say $P\vec{x}$. Suppose that $?P\vec{x} \models_\chi ?\psi(\vec{y})$. Then, by Theorem 1,

$$\forall\vec{x}(P\vec{x} \leftrightarrow P'\vec{x}), \chi, \chi^* \models \forall\vec{y}(\psi(\vec{y}) \leftrightarrow \psi^*(\vec{y})) \ . \tag{11}$$

As a fact of first order logic, we can replace the universally quantified variables in the consequent by some freshly chosen constants $\vec{c}$. This results in

$$\forall\vec{x}(P\vec{x} \leftrightarrow P'\vec{x}), \chi, \chi^* \models \psi(\vec{c}) \leftrightarrow \psi^*(\vec{c}) \ , \tag{12}$$

and, from this,

$$\forall\vec{x}(P\vec{x} \leftrightarrow P'\vec{x}), \chi^*, \psi^*(\vec{c}) \models \chi \rightarrow \psi(\vec{c}) \ . \tag{13}$$

By Craig's interpolation theorem for first order logic, we can construct an interpolant $\vartheta(\vec{c})$ such that

$$\forall\vec{x}(P\vec{x} \leftrightarrow P'\vec{x}), \chi^*, \psi^*(\vec{c}) \models \vartheta(\vec{c}) \ , \tag{14}$$

$$\vartheta(\vec{c}) \models \chi \rightarrow \psi(\vec{c}) \ , \tag{15}$$

and the only non-logical symbols in $\vartheta(\vec{c})$ are those occurring on both sides of (13). From the way the translation procedure $(\cdot)^*$ is set up, it follows that the only non-logical symbols that χ^* and ψ^* on the one hand, and χ and ψ on the other hand, have in common, are rigid function symbols. Thus, $\vartheta(\vec{c})$ contains no non-logical symbols beside P, $\vec{c}$, and rigid function symbols.

Removing primes uniformly from all predicate and function symbols in (14), we get $\chi \models \psi(\vec{c}) \rightarrow \vartheta(\vec{c})$. From (15), we get the converse: $\chi \models \vartheta(\vec{c}) \rightarrow \psi(\vec{c})$. Together, this gives us

$$\chi \models \psi(\vec{c}) \leftrightarrow \vartheta(\vec{c}) \ . \tag{16}$$

Since the constants $\vec{c}$ do not occur in χ, we can replace them by universally quantified variables. This results in

$$\chi \models \forall\vec{y}(\psi(\vec{y}) \leftrightarrow \vartheta(\vec{y})) \ . \tag{17}$$

Furthermore, $\vartheta(\vec{y})$ contains no non-logical symbols beside P and rigid function symbols. From this, it follows that $\vartheta(\vec{y})$ is a development of $P\vec{x}$.

As for the general case, suppose $?\phi(\vec{x}) \models_\chi ?\psi$. Choose a fresh predicate symbol P with the same arity as the number of free variables of ϕ. Then it follows that $?P\vec{x} \models_{\chi \wedge \forall\vec{x}(P\vec{x} \leftrightarrow \phi(\vec{x}))} ?\psi$. Apply the above strategy to obtain a development ϑ of $?P\vec{x}$ such that $\chi \wedge \forall\vec{x}(P\vec{x} \leftrightarrow \phi(\vec{x})) \models \forall\vec{y}(\vartheta \leftrightarrow \psi)$. Let ϑ' be the result of replacing all subformulas in ϑ of the form $P\vec{z}$ by $\phi(\vec{z})$. Then ϑ' is a development of ϕ, and $\chi \models \forall\vec{y}(\psi \leftrightarrow \vartheta')$. □

Thus, the syntactic notion of development corresponds precisely to the semantic notion of entailment between questions.

Recall that the formula χ in Theorem 3 represents an assertion in the common ground. If no assumptions on the common ground are made (i.e., $\chi = \top$), then Theorem 3 reduces to the following syntactic characterization of answerhood.

Corollary 1. *ψ is an answer to $?\phi$ iff ψ is equivalent to a development of ϕ.*

This syntactic characterization is useful for several purposes. First of all, it makes possible a thorough investigation of the predictions made by Groenendijk

and Stokhof's theory of answerhood: It shows what their semantic theory really amounts to, syntactically speaking.

Second, this result opens the way to practical question answering algorithms. Now that we have a syntactic characterization of answerhood, we can address the question answering problem purely in terms of symbolic manipulation without having to refer to the semantics. This will be the topic of Sect. 3.

Another advantage is that it becomes possible to compare different theories of questions and answers. For instance, while at first sight Prolog and the partition theory of questions seem incomparable, we will see in Sect. 3.2 that they are in fact closely related.

2.1 Languages without Equality

What if we do not have equality in the language? This is an interesting question for at least two reasons. First, as we saw in Sect. 1.2, a number of counterintuitive predictions of the partition semantics involve equality. For example, sentences such as $\neg(j \approx b)$ ("John is not Bill", where j and b are interpreted rigidly) and $\exists x \exists y \neg(x \approx y)$ ("There exist at least two entities") are answers to every question. Removing equality would prevent these predictions, though then answers like "Only John is going to the party" would no longer be expressible.

The second reason why eliminating equality from the language is interesting is more practical. In question answering algorithms, it is convenient not to have to deal with equality, since equality reasoning is very expensive. From the theorem proving literature, one can conclude that dealing with equality is not feasible for many practical applications.

As it turns out, the syntactic characterization result is even simpler for first order languages without equality. The corresponding notion of *development of* ϕ is simply *a formula built from rigid instances of* ϕ *using boolean connectives and quantifiers, or* $\top$ *or* $\bot$. The same proof of Theorem 3 goes through, since Craig's interpolation theorem holds regardless of whether equality is present.

2.2 Multiple Questions

The notion of answerhood that we introduced earlier (Definition 1) generalizes trivially to multiple questions: If ψ is a first order formula without free variables, we say that ψ is an answer to $?\Phi$ if $?\Phi \models ?\psi$. The notion of development (Definition 3) also generalizes: A formula ψ is a development of Φ if ψ is built up from rigid instances of elements of Φ and rigid identity statements using boolean connectives and quantifiers, or if ψ is $\top$ or $\bot$. The proof of Theorem 3 then generalizes directly to multiple questions, as does Corollary 1: ψ is an answer to $?\Phi$ iff ψ is equivalent to a development of Φ.

Multiple questions arise naturally in two settings. First, a query like "Who got invited to the party, and who is going to the party?" corresponds to a set of questions like $?\Phi = \{?Ix, ?Px\}$. According to the above generalization of answerhood, one answer to this query is $\forall x(Ix \leftrightarrow Px)$: "Everyone invited is going, and vice versa."

Second, suppose that, as part of their common ground, the questioner and the answerer both know the complete true answers to a certain set of questions $?\Theta$. For example, if it is commonly known who got invited, then $?\Theta$ contains $?Ix$. Extend the notion of entailment in (4) to handle questions as contexts in the following way: If χ is a first order formula with no free variables, then

$$?\Phi \models_{\chi,?\Theta} ?\psi \quad \text{iff} \quad \forall M = (W, D, I): \quad M \models \chi,\ [?\Theta]_M = W^2 \Rightarrow [?\Phi]_M \subseteq [?\psi]_M \ . \tag{18}$$

Intuitively, given the common ground $?\Theta$, a formula ψ is an answer to a question $?\phi$ just in case the entailment $?\phi \models_{?\Theta} ?\psi$ holds. Indeed, the entailment

$$?Px \models_{?Ix} ?\forall x(Ix \leftrightarrow Px) \tag{19}$$

is valid: If it is commonly known who got invited, then in response to "Who is going to the party?" one can answer "Everyone invited is going, and vice versa." In fact, we have

$$?\Phi \models_{\chi,?\Theta} ?\psi \quad \text{iff} \quad ?\Theta, ?\Phi \models_{\chi} ?\psi \ . \tag{20}$$

In particular,

$$?\phi \models_{?\Theta} ?\psi \quad \text{iff} \quad ?\Theta, ?\phi \models ?\psi \ . \tag{21}$$

This result means that asking a question $?\phi$ under some common ground $?\Theta$ is exactly like asking the set of questions $?\Theta \cup \{?\phi\}\cup$ under no common ground, in terms of what counts as an answer. So multiple questions arise again [7].

We are about to present a question-answering algorithm. For simplicity, we will assume that there is only a single question to answer, and that the context is empty. The algorithm we will present generalizes trivially to handle multiple questions at once, and hence to handle questions in the context.[5]

3 Finding the Answer to a Question

The general task of question answering is the following.

> *Given a finite first order theory Σ and a question $?\phi$, find an answer to $?\phi$ that is entailed by Σ and that is as informative as possible.*

In other words, we wish to find an informative formula ψ, such that $\Sigma \models \psi$ and $?\phi \models ?\psi$. Here the theory Σ is intended to capture the answerer's private knowledge, and we measure how *informative* an answer is by its logical strength: One answer is more informative than another if the former entails the latter. For example, an answerer who knows that John is going to the party and that it rains ($\Sigma = \{Pj, R\}$) might reply to "Who is going to the party?" ($?\phi = ?Px$) with the statement "Someone is going to the party" ($\psi = \exists x Px$). If the constant

[5] In passing, we mention that the semantics of multiple questions can be reduced to that of single questions, provided that the domain contains at least two elements. More specifically, any two questions $?\phi_1$ and $?\phi_2$ are equivalent to the single question $?\big((x{\approx}y \wedge \phi_1) \vee (\neg(x{\approx}y) \wedge \phi_2)\big)$. This reduction is of practical importance to the reader who encounters a fairy that promises to answer a single eternal burning question.

symbol j is rigid, then the statement "John is going to the party" (Pj) would be an answer and preferred because it is more informative.

The first question that comes to mind when considering the task of question answering is: *Is there always an optimal answer?* In other words, given a finite theory Σ and a question $?\phi$, is there always (modulo logical equivalence) a unique most informative answer to $?\phi$ entailed by Σ? The answer is *no*.

Theorem 4. *There is a finite theory Σ and a question $?\phi$ such that, for every answer to $?\phi$ entailed by Σ, there is a strictly more informative answer to $?\phi$ entailed by Σ.*

Proof. Let Σ be the theory

$$\{\forall xyz(x < y \wedge y < z \rightarrow x < z),\ \forall x \neg(x < x),\ \forall x \exists y(x < y)\} \tag{22}$$

("$<$ is an unbounded strict order"), and ψ_n ($n \in \mathbb{N}$) be the formula

$$\exists x_1 \ldots \exists x_n \bigwedge_{i,j \leq n;\ i \neq y} \neg(x_i \approx x_j) \tag{23}$$

("there are at least n different objects"). Every ψ_n is entailed by Σ. Furthermore, every ψ_n is an answer to the question $?\top$. (This follows either via Theorem 2, or directly from how question entailment was defined in (4).) The optimal answer would have to entail each ψ_n and, furthermore, contain no non-logical symbols. It follows from the compactness theorem that there is no such formula. □

Equality is not essential to this counterexample. Indeed, the same argument goes through if we replace the equality sign $\approx$ by a non-logical binary relation I, use $?Ixy$ as the question, and extend the theory Σ with *replacement axioms* [5]:

$$\begin{aligned} \Sigma^+ = \Sigma \ \cup\ \{\ & \forall x Ixx,\ \forall xyzu(Ixz \wedge Iyu \wedge x < y \rightarrow z < u), \\ & \forall xyzu(Ixz \wedge Iyu \wedge Ixy \rightarrow Izu)\ \}\ . \end{aligned} \tag{24}$$

Moreover, in the absence of equality there is a second problem, concerning undecidability. It is undecidable whether the given formula is the most specific answer to a question entailed by a theory. This follows from a simple reduction argument: Suppose we do not have equality in the language, and suppose the theory Σ does not contain any rigid function symbols. Then Σ is satisfiable iff $\top$ is the most specific answer to $?\top$ entailed by Σ. But as we all know, first order satisfiability is undecidable.

Notwithstanding these negative results, it is possible to construct a sound and complete question answering algorithm. The output of the algorithm is a sequence of answers that is *cofinal* in the set of all answers entailed by the theory Σ. Cofinality means that, for any answer that is entailed by Σ, there is a more informative answer in the sequence generated by the algorithm. Formally:

Definition 4 (Cofinality). *A set of formulas Φ is* cofinal *in another set of formulas Ψ if each formula in Ψ is entailed by some formula in Φ.*

We call a question answering algorithm *sound* if it generates only formulas that are answers to the question and that are entailed by the theory. We call the

algorithm *complete* if the sequence it generates is always cofinal in the set of all answers to the question entailed by the theory. A question answering algorithm with these properties is depicted in Fig. 1.

$$\left.\begin{array}{r}\text{Theory } \Sigma \\ \text{Question } ?\phi\end{array}\right\} \xrightarrow{\text{INPUT}} \boxed{\text{QA Algorithm}} \xrightarrow{\text{OUTPUT}} \text{Sequence of answers } \psi_1, \psi_2, \ldots$$

$$\begin{array}{rl} \textit{Soundness:} & \forall i : \Sigma \models \psi_i \ \& \ ?\phi \models ?\psi_i \\ \textit{Completeness:} & \forall \psi : \Sigma \models \psi \ \& \ ?\phi \models ?\psi \implies \exists i : \psi_i \models \psi \end{array}$$

Fig. 1. A question answering algorithm

To show that there are indeed algorithms satisfying these constraints, we will now discuss a rather trivial algorithm. Recall that, to answer a question $?\phi$ given a theory Σ, it suffices to find developments of ϕ that are entailed by Σ. The search for answers can be conducted using any theorem proving technique, including tableaux. A first stab at a question answering algorithm, then, is to syntactically enumerate and check all potential answers ψ.

Algorithm 1. To answer the question $?\phi$ given the theory Σ, repeat in dovetail fashion for every development ψ of ϕ:

1. Initialize a tableau with a single branch, consisting of Σ and $\neg\psi$.
2. Keep applying tableau expansion and closure rules.
3. If the tableau becomes closed, report ψ as an answer. □

This algorithm is sound and complete, because tableau-based first order theorem proving is. However, it is also horribly inefficient, since it considers every development of ϕ as a potential answer, without taking the theory Σ into account.

The challenge, then, is to construct an question answering algorithm that is sound and complete (like Algorithm 1) yet as efficient as possible. In the next section, we will make a start by providing a more intelligent algorithm.

3.1 Tableau-Based Question Answering

We will now introduce a sound and complete question answering algorithm based on free variable tableaux [5]. Without loss of generality, we assume that the question $?\phi$ consists of a single atomic formula $P\vec{x}$, called the *answer literal* [6]. (If that is not so, simply add the formula $\forall\vec{x}(P\vec{x} \leftrightarrow \phi)$ to the theory, where P is a new predicate symbol and $\vec{x}$ are the free variables of ϕ.) We also assume that the theory Σ is Skolemized; that is, any existential quantifier in Σ has been eliminated with the help of additional, non-rigid function symbols. Finally, we will disregard equality for the moment, but the algorithm can be extended to deal with equality as well (we will return to this issue at the end of this section).

Table 1 summarizes the tableau calculus for first order logic without equality given by Fitting [5]. (We omit the existential rule since the theory is Skolemized.)

Table 1. Tableau expansion and closure rules for theorem proving

Conjunction	$\dfrac{\phi \wedge \psi}{\phi, \psi}$	$\dfrac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi}$
Quantification	$\dfrac{\neg\exists x\phi}{\neg\phi[x/y]}$	where y is fresh
Negation	$\dfrac{\neg\neg\phi}{\phi}$	
Closure	A branch is closed if it contain a formula and its negation. A tableau is closed if all its branches are closed. A substitution σ is *closing* if the tableau is closed after applying σ.	

In the algorithm that we will introduce shortly, besides the usual tableau expansion rules, one other operation is allowed. At any stage of tableau construction, one is allowed to create a copy of the question—renaming its free variables to new variables—and either add the copy to all branches or add its negation to all branches. We term this operation the *Add Instance* rule, shown in Table 2.

Table 2. The Add Instance rule for question answering

Add Instance	$\overline{P\vec{y}}$	$\overline{\neg P\vec{y}}$
	where $?P\vec{x}$ is the question, and $\vec{y}$ are fresh variables	

Note that, whereas the rules in Table 1 are only applied to a single branch at a time, the Add Instance rule is always applied to the entire tableau. This difference is not essential—completeness is not lost if we restrict the application of the Add Instance rule to single branches—but it increases efficiency and it renders the generated answers more concise.

Example 1. A partially developed tableau is given in Fig. 2, where we are given the theory $\Sigma = \{(Pa \wedge Pc) \vee (Pb \wedge Pc)\}$ and wish to answer the question $?Px$. Note the use of the Add Instance rule, indicated by "!". Also note that $y := c$ is a closing substitution: After applying it, the tableau is closed.

Every closing substitution generates an answer, in the following way. Suppose we face a partially developed tableau T, and it can be closed by a substitution σ. Let Φ be the (finite) set of all instances of the question that have been added so far using the Add Instance rule. Then $\neg \bigwedge \Phi^\sigma$ is an appropriate answer. In fact, we can even do a little better and give a slightly more informative answer, by only considering those added instances that participate in closure.

Definition 5 (Closure). *Let T be a tableau, σ be a substitution over T, and κ be a set of formula occurrences in T. The pair (σ, κ) is a* closure *if κ^σ contains, on each branch, some formula and its negation.*

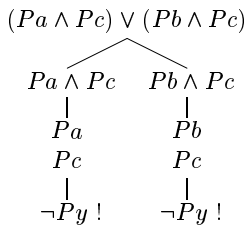

Fig. 2. A partially developed tableau

For every closing substitution σ, there is at least one closure (σ, κ). Let Φ_κ consist of exactly those formulas in Φ that appear in κ. With $\textsc{ans}(\sigma, \kappa)$ we will denote the result of universally quantifying all free variables in $\neg \bigwedge (\Phi_\kappa)^\sigma$.

Example 2. Consider again the tableau in Fig. 2. Let σ be the substitution $y := c$, and let κ contain the occurrences of Pc and $\neg Py$ on both branches. Then the pair (σ, κ) is a closure. It generates the answer $\textsc{ans}(\sigma, \kappa) = \neg\neg Pc$.

When generating answers from closures, we will only be interested in the *most general* closures. One closure (σ_1, κ_1) is more general than another (σ_2, κ_2) if σ_1 is more general than σ_2 and κ_1 is a subset of κ_2. Closures that are more general do as little as possible besides closing the tableau, and so generate answers that are more informative. When there is more than one most general closure, we compute all the corresponding answers and take their conjunction.

One remaining difficulty concerns rigid function symbols. According to our syntactic characterization of answerhood, non-rigid constants and function symbols are not allowed to occur in generated answers. For instance, suppose the theory is $\{Pc\}$, where c is a non-rigid constant. To the question $?Px$, we must answer $\exists x Px$ rather than Pc. Moreover, Skolem functions created during tableau expansion are also considered non-rigid and disallowed in answers. For instance, given the theory $\{\exists x Px\}$ and the question $?Px$, we must answer $\exists x Px$, rather than "Ps, where s is the Skolem constant such that Ps". Our algorithm achieves this by applying Chadha's *unskolemization* procedure [4].

The complete question answering algorithm is as follows.

Algorithm 2. To answer the question $?P\vec{x}$ given the theory Σ, start by initializing a tableau for Σ. Next, do one of the following, repeatedly, *ad infinitum*.

1. Apply a tableaux expansion rule (Table 1).
2. Apply the Add Instance rule (Table 2).
3. Take the conjunction $\psi_0 = \bigwedge \textsc{ans}(\sigma, \kappa)$ over all most general closures (σ, κ). Unskolemize ψ_0 to remove any non-rigid or Skolem function symbols, and report the result as the next answer. □

Theorem 5. *Algorithm 2 is sound and complete, provided the non-deterministic choices are made in a* fair *manner.*

Proof sketch. Call a formula ψ a *pre-answer* if ψ is built up from instances of the answer literal $P\vec{x}$—allowing non-rigid and Skolem function symbols—using boolean connectives and quantifiers, or if ψ is $\top$ or $\bot$. We first prove that the algorithm, minus unskolemization, generates pre-answers in a sound and complete manner. Soundness means that the algorithm only generates (yet-to-be-unskolemized) conjunctions ψ_0 that are pre-answers and that are entailed by the theory. Completeness means that the sequence of conjunctions generated is cofinal in the set of all pre-answers entailed by the theory.

Soundness: Because $\textsc{ans}(\sigma, \kappa)$ is always a pre-answer, so is any generated conjunction ψ_0. Moreover, any closure (σ, κ) for our question-answering tableau gives rise to to a closed theorem-proving tableau for $\Sigma \cup (\Phi_\kappa)^\sigma$. Since tableau theorem proving is sound, the theory Σ must entail $\textsc{ans}(\sigma, \kappa)$. Hence Σ entails ψ_0.

Completeness: Suppose that ψ is a pre-answer entailed by Σ. By completeness of tableau theorem proving, we can find a closed theorem-proving tableau for $\Sigma \cup \{\neg\psi\}$, and systematically transform it into a question-answering tableau for $(\Sigma, ?P\vec{x})$ that generates a pre-answer ψ_0 entailing ψ. Given this, it can be shown that in fact any fair question-answering tableau expansion procedure will eventually generate a pre-answer entailing ψ.

We now need to relate pre-answers to answers. The unskolemization procedure is sound; that is, unskolemizing ψ_0 always gives a formula entailed by ψ_0. Besides, unskolemizing any pre-answer gives an answer, so Algorithm 2 is sound. The unskolemization procedure is also complete; that is, whenever a pre-answer ψ_0 entails some answer ψ, the result of unskolemizing ψ_0 also entails ψ. Besides, every answer is a pre-answer, so Algorithm 2 is complete. □

One way to guarantee fairness in Algorithm 2 is to implement it with *depth first iterative deepening*, just as Beckert and Posegga [2] did in their $\mathsf{lean}T^AP$ prover. In fact, we have modified $\mathsf{lean}T^AP$ to become a lean question answerer.

The difference between Algorithms 1 and 2 is that the latter waits until closing the tableau before deciding which answer to prove to follow from the theory. In other words, Algorithm 2 does not commit to the rigid instances that constitute the development until they are determined by the closure. This ability to postpone commitment is exactly the strength of free variable tableau calculi as compared to ground tableaux [5].

This algorithm can be extended to deal with equality in two steps. First, add the tableau rules necessary for theorem proving with equality [1, 5]. Second, generalize the Add Instance rule, so that not only instances of the question but also (in)equalities can be added to the tableau.

3.2 Prolog as a Special Case

A precise connection can be established between Prolog and our algorithm. Prolog performs question answering in a sense more restrictive than considered here, because it makes extra assumptions about the theory Σ and the question $?\phi$:

1. The theory Σ is required to be in Skolemized Horn form and the question must consist of a single atom, in order to make computation feasible.

2. There is no equality predicate in the basic language of Prolog.
3. Prolog assumes that all function symbols and constants are rigid, making Pc a potential answer to $?Px$ even if the symbol c resulted from Skolemization.
4. Due to its depth-first search strategy, Prolog is complete only for some theories, for example theories without cycles among predicate symbols.

Subject to all these restrictions, Prolog is an optimal question answering algorithm: Given a theory and a question, it produces answers optimal in the sense of the partition theory of questions.[6]

As we can see, Prolog is a question answering algorithm that makes many extra assumptions. The present work makes it possible to distinguish and identify these assumptions, and eliminate them. A broad spectrum of generalizations of Prolog can then be considered.

4 Conclusion

This paper makes two contributions. First, we presented a syntactic characterization of answerhood for the partition semantics of questions. The applications of this result are mainly internal to the partition semantics: It explains the meaning of a question in terms of the form of its answers.

Second, our tableau-based question answering algorithm connects two important research traditions: question answering systems such as Prolog, and the formal semantics of natural language questions. We feel that the link between these two fields of research has been neglected in the past, and hope to bring them together.

We want to mention three directions of further research (we stress the third).

Logical Some theoretical issues are still to be addressed: How can answerhood be characterized syntactically when the semantics allows varying domains? Also, is it decidable whether a given answer is the optimal answer to a question entailed by a theory? Without equality, this problem is undecidable, as we proved in Sect. 3. We have yet to find a similar reduction argument for the case with equality.

Linguistic We want to bring our theoretical results to bear on the semantics of questions in natural language. In particular, our syntactic characterization result clarifies the linguistic predictions made by the partition semantics [14].

Computational We want to use this work as a unifying framework to compare different approaches to question answering. In particular, we want to investigate a variety of Prolog generalizations and determine which assumptions made by Prolog are feasible to drop.

Like us, Green [6] and Luckham and Nilsson [12] have applied theorem proving to question answering, but their criteria for answerhood are tied

[6] One apparent difference between Prolog and our algorithm is that Prolog is based on resolution, whereas we used tableaux. However, the two methods are closely related, and Prolog can be interpreted as a variant of so-called *connection tableaux* [10].

to the syntax of formulas in prenex normal form. This difference explains why their algorithms omit unskolemization, a step necessary for soundness under the notion of answerhood we adopt here. Also, Bos and Gabsdil [3] have devised a simplified version of the partition semantics for computational purposes. These efforts seem to fit nicely in our picture.

References

[1] Beckert, Bernhard. 1994. Adding equality to semantic tableaux. In *Proceedings of the 3rd workshop on theorem proving with analytic tableaux and related methods*, ed. Krysia Broda, Marcello D'Agostino, Rajeev Goré, Rob Johnson, and Steve Reeves, 29–41. Tech. Rep. TR-94/5, Department of Computing, Imperial College, London.

[2] Beckert, Bernhard, and Joachim Posegga. 1995. leanTAP: Lean, tableau-based deduction. *Journal of Automated Reasoning* 15(3):339–358.

[3] Bos, Johan, and Malte Gabsdil. 2000. First-order inference and the interpretation of questions and answers. In *Proceedings of Götalog 2000*, ed. Massimo Poesio and David Traum, 43–50. Gothenburg Papers in Computational Linguistics 00-5.

[4] Chadha, Ritu. 1991. Applications of unskolemization. Ph.D. thesis, Department of Computer Science, University of North Carolina.

[5] Fitting, Melvin C. 1996. *First order logic and automated theorem proving.* 2nd ed. Berlin: Springer-Verlag.

[6] Green, Claude Cordell. 1969. The application of theorem proving to question-answering systems. Ph.D. thesis, Department of Electrical Engineering, Stanford University. Reprinted by New York: Garland, 1980.

[7] Groenendijk, Jeroen. 1999. The logic of interrogation: Classical version. In *SALT IX: Semantics and linguistic theory*, ed. Tanya Matthews and Devon Strolovitch, 109–126. Ithaca: Cornell University Press.

[8] Groenendijk, Jeroen, and Martin Stokhof. 1984. Studies on the semantics of questions and the pragmatics of answers. Ph.D. thesis, Universiteit van Amsterdam.

[9] ———. 1996. Questions. In *Handbook of logic and language*, ed. Johan van Benthem and Alice ter Meulen, 1055–1124. Amsterdam: Elsevier Science.

[10] Hähnle, Reiner. 2001. Tableaux and related methods. In *The handbook of automated reasoning*, ed. Alan Robinson and Andrei Voronkov, vol. 1, chap. 3, 100–178. Amsterdam: Elsevier Science.

[11] Kager, Wouter. 2001. Questions and answers in query logic. Master's thesis, Universiteit van Amsterdam.

[12] Luckham, David, and Nils J. Nilsson. 1971. Extracting information from resolution proof trees. *Artificial Intelligence* 2:27–54.

[13] Nelken, Rani, and Nissim Francez. 2000. A calculus of interrogatives based on their algebraic semantics. In *Proceedings of TWLT16/AMILP2000*, ed. Dirk Heylen, Anton Nijholt, and Giuseppe Scollo, 143–160.

[14] Shan, Chung-chieh, and Balder ten Cate. 2002. The partition semantics of questions, syntactically. In *Proceedings of the ESSLLI-2002 student session*, ed. Malvina Nissim.