Proceedings of HLT-NAACL 2003
Main Papers , pp. 87-93
Edmonton, May-June 2003

COGEX: A Logic Prover for Question Answering

Dan Moldovan, Christine Clark, Sanda Harabagiu
Language Computer Corporation
Richardson, Texas 75080
moldovan@languagecomputer.com

Steve Maiorano
ATP
Washington, DC 20505

Abstract

Recent TREC results have demonstrated the need for deeper text understanding methods. This paper introduces the idea of automated reasoning applied to question answering and shows the feasibility of integrating a logic prover into a Question Answering system. The approach is to transform questions and answer passages into logic representations. World knowledge axioms as well as linguistic axioms are supplied to the prover which renders a deep understanding of the relationship between question text and answer text. Moreover, the trace of the proofs provide answer justifications. The results show that the prover boosts the performance of the QA system on TREC questions by 30%.

1 Introduction

Motivation

In spite of significant advances made recently in the Question Answering technology, there still remain many problems to be solved. Some of these are: bridging the gap between question and answer words, pinpointing exact answers, taking into consideration syntactic and semantic roles of words, better answer ranking, answer justification, and others. The recent TREC results (Voorhees 2002) have demonstrated that many performing systems reached a plateau; the systems ranked from 4th to 14th answered correctly between 38.4% to 24.8% of the total number of questions. It is clear that new ideas based on a deeper language understanding are necessary to push further the QA technology.

In this paper we introduce one such novel idea, the use of automated reasoning in QA, and show that it is feasible, effective, and scalable. We have implemented a Logic Prover, called COGEX (from the permutation of the first two syllables of the verb *excogitate*) which uniformly codifies the question and answer text, as well as world knowledge resources, in order to use its inference engine to verify and extract any lexical relationships between the question and its candidate answers.

Usefulness of a Logic Prover in QA

COGEX captures the syntax-based relationships such as the syntactic objects, syntactic subjects, prepositional attachments, complex nominals, and adverbial/adjectival adjuncts provided by the logic representation of text. In addition to the logic representations of questions and candidate answers, the QA Logic Prover needs world knowledge axioms to link question concepts to answer concepts. These axioms are provided by the WordNet glosses represented in logic forms. Additionally, the prover needs rewriting procedures for semantically equivalent lexical patterns. With this deep and intelligent representation, COGEX effectively and efficiently re-ranks candidate answers by their correctness, extracts the exact answer, and ultimately eliminates incorrect answers. In this way, the Logic Prover is a powerful tool in boosting the accuracy of the QA system. Moreover, the trace of a proof constitutes a justification for that answer.

Technical challenges

The challenges one faces when using automated reasoning in the context of NLP include: logic representation of open text, need of world knowledge axioms, logic representation of semantically equivalent linguistic patterns, and others. Logic proofs are accurate but costly, both in terms of high failure rate due to insufficient input axioms, as well as long processing time. Our solution is to integrate the prover into the QA system and rely on reasoning methods only to augment other previously implemented answer extraction techniques.

2 Integration of Logic Prover into a QA System

The QA system includes traditional modules such as question processing, document retrieval, answer extraction, built in ontologies, as well as many tools such as syntactic parser, name entity recognizer, word sense disambiguation (Moldovan and Noviscki 2002), logic representation of text (Moldovan and Rus 2001) and others. The Logic Prover is integrated in this rich NLP environment and augments the QA system operation.

As shown in Figure 1, the inputs to COGEX consist of logic representations of questions, potential answer paragraphs, world knowledge and lexical information. The term Answer Logic Form (ALF) refers to the candidate answers in logic form. Candidate answers returned by the Answer Extraction module are classified as open text due to the unpredictable nature of their grammatical structure. The term Question Logic Form (QLF) refers to the questions posed to the Question Answering system represented in logic form.

The prover also needs world knowledge axioms supplied by the WordNet glosses transformed into logic representations. Additionally there are many other axioms representing equivalence classes of linguistic patterns, called NLP axioms. All these are described below.

The Axiom Builder converts the Logic Forms for the question, the glosses, and its candidate answers into axioms. Based on the parse tree patterns in the question and answers, other NLP axioms are built to supplement the existing general NLP axioms. Once the axioms are complete and loaded, justification of the answer begins. If a proof fails, the relaxation module is invoked. The purpose of this module is twofold: (1) to compensate for errors in the text parsing and Logic Form transformation phase, such as prepositional attachments and subject/object detection in verbs, (2) to detect correct answers when the NLP and XWN (Extended WordNet) axioms fail to provide all the necessary inferences. During the relaxation, arguments to predicates in the question are incrementally uncoupled, the proof score is reduced, and the justification is re-attempted. The loop between the Justification and the Relaxation modules continues until the proof succeeds, or the proof score is below a predefined threshold. When all the candidate answers are processed, the candidate answers are ranked based on their proof scores, with the output from COGEX being the ranked answers and the answer justifications.

3 Logic Representation of Text

A text logic form (LF) is an intermediary step between syntactic parse and the deep semantic form. The LF codification acknowledges syntax-based relationships such as: (1) syntactic subjects, (2) syntactic objects, (3) prepositional attachments, (4) complex nominals, and (5) adjectival/adverbial adjuncts. Our approach is to derive the LF directly from the output of the syntactic parser which already resolves structural and syntactic ambiguities.

Essentially there is a one to one mapping of the words of the text into the predicates in the logic form. The predicate names consist of the base form of the word concatenated with the part of speech of the word. Each noun has an argument that is used to represent it in other predicates. One of the most important features of the Logic Form representation is the fixed-slot allocation mechanism of the verb predicates (Hobbs 1993). This allows for the Logic Prover to see the difference between the role of the subjects and objects in a sentence that is not answerable in a keyword based situation.

Logic Forms are derived from the grammar rules found in the parse tree of a sentence. There are far too many grammar rules in the English language to efficiently and realistically implement them all. We have observed that the top ten most frequently used grammar rules cover 90% of the cases for WordNet glosses. This is referred to as the 10-90 rule (Moldovan and Rus 2001). Below we provide a sample sentence and its corresponding LF representation.

Example:
Heavy selling of Standard & Poor's 500-stock index futures in Chicago relentlessly beat stocks downward.

LF:
heavy_JJ(x1) & selling_NN(x1) & of_IN(x1,x6) & Standard_NN(x2) & &_CC(x13,x2,x3) & Poor_NN(x3) &'s_POS(x6,x13) & 500-stock_JJ(x6) & index_NN(x4) & future_NN(x5) & nn_NNC(x6,x4,x5) & in_IN(x1,x8) & Chicago_NN(x8) & relentlessly_RB(e12) & beat_VB(e12,x1,x9) & stocks_NN(x9) & downward_RB(e12)

4 World Knowledge Axioms

Logic representation of WordNet glosses

A major problem in QA is that often an answer is expressed in words different from the question keywords. World knowledge is necessary to conceptually link questions and answers. WordNet glosses contain a source of world knowledge. To be useful in automated reasoning, the glosses need to be transformed into logic forms. Taking the same approach as for open text, we have parsed and represented in logic forms more than 50,000 WordNet glosses. For example, the gloss definition of concept `sport_NN#1` is *an active diversion requiring physical exertion and competition*, which yields the logic representation:
active_JJ(x1) & diversion_NN(x1) & require_VB(e1,x1,x2) & or_CC(x2,x3,x4) & physical_JJ(x3) & exertion_NN(x3) & competition_NN(x4)

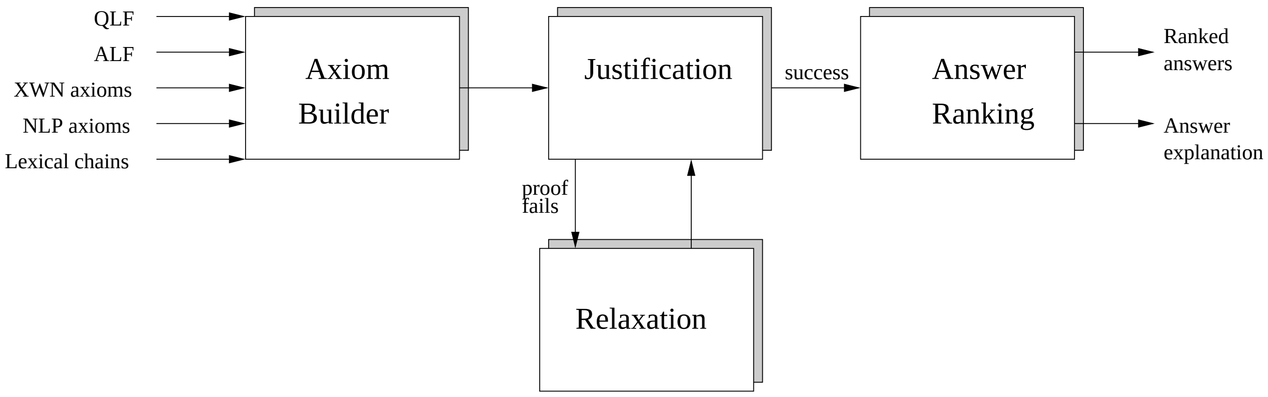

Figure 1: COGEX Architecture

Lexical Chains
A much improved source of world knowledge is obtained when the gloss words are semantically disambiguated (Moldovan and Noviscki 2002). By doing this, the connectivity between synsets is dramatically increased. Lexical chains can be established between synsets in different hierarchies. These are sequences of semantically related words that link two concepts.

Lexical chains improve the performance of question answering systems in two ways: (1) increase the document retrieval recall and (2) improve the answer extraction by providing the much needed world knowledge axioms that link question keywords with answers concepts.

We developed software that automatically provides connecting paths between any two WordNet synsets S_i and S_j up to a certain distance (Moldovan and Noviscki 2002). The meaning of these paths is that the concepts along a path are topically related. The path may contain any of the WordNet relations augmented with a GLOSS relation which indicates that a certain concept is present in a synset gloss.

Examples
Below we provide some relevant lexical chains that link a few selected TREC 2002 questions with their answers.

Q1394: What country did the game of croquet originate in ?
Answer: Croquet is a 15th-century French sport that has largely been dominated by older, wealthier people who play at exclusive clubs.
Lexical chains:
(1) game:n#3 → HYPERNYM → recreation:n#1 → HYPONYM → sport:n#1
(2) originate_in:v#1 → HYPONYM → stem:v#1→ GLOSS → origin:n#1 → GLOSS → be:v#1

Q1403: When was the internal combustion engine invented ?
Answer: The first internal - combustion engine was built in 1867.
Lexical chains:
(1) invent:v#1 → HYPERNYM → create_by_mental_act:v#1 → HYPERNYM → create:v#1 → HYPONYM → build:v#1

Q: 1518 What year did Marco Polo travel to Asia ?
Answer: Marco Polo divulged the truth after returning in 1292 from his travels, which included several months on Sumatra.
Lexical chains:
(1) travel_to:v#1 → GLOSS → travel:v#1 → RGLOSS → travel:n#1
(2) travel_to#1 → GLOSS → travel:v#1 → HYPONYM → return:v#1
(3) Sumatra:n#1 → ISPART → Indonesia:n#1 → ISPART → Southeast_Asia:n#1 → ISPART → Asia:n#1

Q: 1540 What is the deepest lake in America ?
Answer: Rangers at Crater Lake National Park in Oregon have closed the hiking trail to the shore of the nation 's deepest lake
Lexical chains:
(1) America:n#1 → HYPERNYM → North_American_country:n#1 → HYPERNYM → country:n#1 → GLOSS → nation:n#1

Q: 1570 What is the legal age to vote in Argentina ?
Answer: Voting is mandatory for all Argentines aged over 18
Lexical chains:
(1) legal:a#1 → GLOSS → rule:n#1 → RGLOSS → mandatory:a#1
(2) age:n#1 →RGLOSS → aged:a#3
(3) Argentine:a#1 → GLOSS → Argentina:n#1

5 NLP Axioms

In additions to world knowledge axioms, a QA Logic Prover needs linguistic knowledge. This is what distinguishes an NLP prover from a traditional mathematical prover. General axioms that reflect equivalence classes of linguistic patterns need to be created and instantiated when invoked. We call these NLP axioms and present below some examples together with questions that call them.

Complex nominals and coordinated conjunctions
A question may refer to a subject/object by its full proper name, and the answer will refer to the subject/object in an abbreviated form. For example in the correct candidate answer for the question, *"Which company created the Internet browser Mosaic?"*, Internet browser Mosaic is referred to as Mosaic.

Using abduction, an axiom is built such that the head noun of the complex nominal in the question implies the remaining nouns in the complex nominal:
all x1 (mosaic_nn(x1) → internet_nn(x1) & browser_nn(x1))
An additional axiom is built such that all the nouns in the complex nominal imply a complex nominal:
all x1 (internet_nn(x1) & browser_nn(x1) & mosaic_nn(x1) → nn_nnc(x1,x1,x1,x1))
So as not to restrict the ordering of the nouns in the noun phrase from which the complex nominal is built, the same argument is used for each of the noun predicates in the complex nominal. Similar to the above issue, a question may refer to the subject/object in an abbreviated form, while the answer will refer to the subject/object in its full, proper form. For example in the correct candidate answer for the question, *"When was Microsoft established?"*, Microsoft is referred to as Microsoft Corp.

An axiom is built such that each noun of the complex nominal takes on the identifying argument of the complex nominal:
all x1 x2 x3 (microsoft_nn(x1) & corp_nn(x2) & nn_nnc(x3,x1,x2) → microsoft_nn(x3) & corp_nn(x3))
Similar axioms are used for coordinated conjunctions detected in the answer and the question. These are considered weak axioms, and any proof that uses them will be penalized by being given a lower score than those that do not.

Appositions
A candidate answer for a question may use an apposition to describe the subject/object of the answer. The question may refer to the subject/object by this apposition. For example in the question, *"Name the designer of the shoe that spawned millions of plastic imitations , known as jellies"*, the candidate answer, *"..Italian Andrea Pfister , designer of the 1979 " bird cage " shoe that spawned millions of plastic imitations, known as " jellies ..."* uses an apposition to describe the designer.

An axiom is built to link the head of the noun phrases in the apposition such that they share the same argument:
all x12 x13 x14 x15 x17 x18 x19 (italian_nn(x12) & andrea_nn(x13) & pfister_NN(x14) & nn_nnc(x15,x12,x13,x14) → designer_nn(x15) & of_in(x15,x17) & 1979_nn(x17) & bird_nn(x18) & cage_nn(x19))

Possesives
A question/answer substitutes the use of a possesive by using an *of* or *by* preposition. For example, in the question, *"What was the length of the Wright brothers' first flight?"*, the candidate answer, *"Flying machines , which got off the ground with a 120 - foot flight by the Wright brothers in 1903..."* implies ownership using the preposition *by* to connect the Wright brothers to flight.

An axiom is built to connect *by* to the possessive:
all x1 x2 (by_in(x1,x2) → _pos(x1,x2))

Equivalence classes for prepositions
Prepositions can be grouped into equivalence classes depending on the context of the question, which is determined by the expected answer type. In location seeking questions the prepositions *at* and *in* are often interchangeable. Similarly for *in* and *into,* and *from* and *of*. In date seeking questions *in* and *of* have interchangeable meanings as do *at* and *in*. For example, in the question, *"What body of water does the Colorado River flow into?"*, the candidate answer, *"...the Colorado River flowed in the Salton trough about 130 miles east of San Diego"*, the preposition *in* and *into* in the answer take in the same meaning.

An axiom is built to link *in* to *into*:
all x1 x2 (in_in(x1,x2) → into_in(x1,x2))

Part of relations in location questions
A location seeking question may have a candidate answer that identifies a location by referring to a part of the location. For example, in the question, *"Where is Devil 's Tower?"*, the answer, *"American Indians won another court battle over their right to worship without interference at Devils Tower National Monument in the northeast corner of Wyoming"*, identifies Wyoming as the location of Devil 's Tower by referring to the part of Wyoming in which it lies. An axiom is built to connect Wyoming to its part:
all x1 x2 x3 (corner_nn(x1) & of_in(x1,x2) & wyoming_nn(x2) → wyoming_nn(x1))

Attribute of relations in quantity seeking questions
A question seeking a quantity may have a candidate answer that implies quantity of subject by prefixing the quantity to the subject. For example in the question *"What is the height of the tallest redwood?"* the answer is *"329 feet Mother of Forest's Big Basin tallest redwood.."*
An axiom is built to connect the quantity to its subject, redwood:
all x1 x2 (_quantity(x1) & redwood_NN(x2) → of_in(x1,x2))

This is a weak axiom since the proximity of redwood to quantity in the answer text is not guaranteed. As mentioned for the complex nominal and coordinated conjunction axioms, any proof that uses these axioms should be penalized and ranked lower than those that do not. Note that for this axiom to be effective, an axiom linking the heads of the apposition is built:

all x8 x9 x10 x11 x12 (mother_nn(x8) & of_in(x8,x9) & forest_nn(x9) → big_nn(x10) & basin_nn(x11) & nn_nnc(x12,x10,x11) & _s_pos(x8,x12) & tall_jj(x8) & redwood_nn(x8))

6 Control Strategy

Axiom partitioning mechanism

The search strategy used is the Set of Support Strategy, which partitions the axioms used during the course of a proof into those that have support and those that are considered auxiliary (Wos 1988). The axioms with support are placed in the Set of Support (SOS) list and are intended to guide the proof. The auxiliary axioms are placed in the Usable list and are used to help the SOS infer new clauses. This strategy restricts the search such that a new clause is inferred if and only if one of its parent clauses come from the Set of Support. The axioms that are placed in the SOS are the candidate answers, the question negated (to invoke the proof by contradiction), and axioms related to linking named entities to answer types.

Axioms placed in the Usable list are: (1) Extended WordNet axioms, (2) NLP axioms, and (3) axioms based on outside world knowledge, such as people and organizations.

Inference rules

The inference rule sets are based on hyperresolution and paramodulation. Hyperresolution is an inference rule that does multiple binary resolution steps in one, where binary resolution is an inference mechanism that looks for a positive literal in one clause and negative form of that same literal in another clause such that the two literals can be canceled, resulting in a newly inferred clause. Paramodulation introduces the notion of equality substitution so that axioms representing equality in the proof do not need to be explicitly included in the axiom lists. Additionally, similar to hyperresolution, paramodulation combines multiple substitution steps into one.

All modern theorem provers use hyperresolution and paramodulation inference rules since they allow for a more compact and efficient proof by condensing multiple steps into one.

COGEX will continue trying to find a proof until the Set of Support becomes empty, a refutation is found, or the proof score drops below a predefined threshold.

Two techniques have been implemented in COGEX to deal with incomplete proofs:

1. Count the number of unifications/resolutions with terms in the question along the longest search path in the proof attempts, and
2. Relax the question logic form by incrementally uncoupling arguments in the predicates, and/or removing prepositions or modifiers that are not crucial to the meaning of the text.

For example in question, *"How far is Yaroslavl from Moscow?"* a candidate answer is *".. Yaroslavl, a city 250 miles north of Moscow."* By dropping the *from* predicate in the question makes the proof succeed for the candidate answer.

7 An example

The following example illustrates how all these pieces are put together to generate answer proofs.

Question 108?:
Which company created the Internet Browser Mosaic ?

QLF:
_organization_AT(x2)) & company_NN(x2) & create_VB(e1,x2,x6) & Internet_NN(x3) & browser_NN(x4) & Mosaic_NN(x5) & nn_NNC(x6,x3,x4,x5)

Question Axiom:
-(exists e1 x2 x3 x5 x6 (_organization_at(x2) & company_nn(x2) & create_vb(e1,x2,x6) & internet_nn(x3) & browser_nn(x4) & mosaic_nn(x5) & nn_nnc(x6,x3,x4,x5))).

Answer:
In particular, a program called Mosaic , developed by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana - Champaign , is gaining popularity as an easy to use point and click interface for searching portions of the Internet.

ALF:
In_IN(x1,x28) & particular_JJ(x29) & program_NN(x1) & call_VB(e1,x27,x30) & Mosaic_NN(x2) & develop_VB(e2,x2,x31) & by_IN(e2,x8) & National_NN(x3) & Center_NN(x4) & for_NN(x5) & Supercomputing_NN(x6) & application_NN(x7) & nn_NNC(x8,x3,x4,x5,x6,x7) & NCSA_NN(x9) & at_IN(e2,x15) & University_NN(x10) & of_NN(x11) & Illinois_NN(x12) & at_NN(x13) & Urbana_NN(x14) & nn_NNC(x15,x10,x11,x12,x13,x14) & Champaign_NN(x16) & gain_VB(e3,x1,x17) & popularity_NN(x17) & as_IN(e3,x32) & easy_JJ(x33) & use_VB(e4,x34,x26) & point_NN(x18) & and_CC(x26,x18,x21) & click_NN(x19) & interface_NN(x20) & nn_NNC(x21,x19,x20) & for_IN(x26,e5) & search_VB(e5,x26x22) & portion_NN(x22) & of_IN(x22,x23) & Internet_NN(x23)

Answer Axiom:
exists e1 e2 e3 e4 e5 x1 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x2 x20 x21 x22 x23 x26 x27 x28 x3 x32 x33 x4 x5 x6 x7 x8 x9 (in_in(e2,x28) & particular_jj(x28) & program_nn(x1) & call_vb(e1,x27,x1) & mosaic_nn(x2) & develop_vb(e2,x8,x2) & by_in(e2,x8) & national_nn(x3) & center_nn(x4) & for_nn(x5) & supercomputing_nn(x6) & application_nn(x7) & nn_nnc(x8,x3,x4,x5,x6,x7) & ncsa_nn(x9) & at_in(x8,x15) & university_nn(x10) & of_nn(x11) & illinois_nn(x12) & at_nn(x13) & urbana_nn(x14) & nn_nnc(x15,x10,x11,x12,x13,x14) & champaign_nn(x16) & gain_vb(e3,x2,x17) & popularity_nn(x17) & as_in(e3,x32) & easy_jj(x33) & use_vb(e4,x9,x2) & point_nn(x18) & and_cc(x26,x18,x21) & click_nn(x19) & interface_nn(x20) & nn_nnc(x21,x19,x20) & for_in(x26,e5) & search_vb(e5,x2,x22) & portion_nn(x22) & of_in(x22,x23) & internet_nn(x23)).

Named entity axioms:
(1) all x3 x4 x5 x6 x7 x8 (national_nn(x3) & center_nn(x4) & for_nn(x5) & supercomputing_nn(x6) & application_nn(x7) & nn_nnc(x8,x3,x4,x5,x6,x7) $\rightarrow$ _organization_at(x8)).
(2) all x10 x11 x12 x13 x14 x15 (university_nn(x10) & of_nn(x11) & illinois_nn(x12) & at_nn(x13) & urbana_nn(x14) & nn_nnc(x15,x10,x11,x12,x13,x14) $\rightarrow$ _university_at(x15)).
(3) all x16 (champaign_nn(x16) $\rightarrow$ _town_at(x16)).

Wordnet Gloss axioms:
The question contained the verb *create* while the answer contains the verb *develop*. In order to prove that this answer is in fact correct, we need to detect and use a lexical chain between *develop* and *create*. WordNet supplies us with that chain such that
develop $\leftrightarrow$ *make* and *make* $\leftrightarrow$ *create*
Using WordNet glosses, this chain is transformed into two axioms:
(4) exists x2 x3 x4 all e2 x1 x7 (develop_vb(e2,x7,x1) $\leftrightarrow$ make_vb(e2,x7,x1) & something_nn(x1) & new_jj(x1) & such_jj(x1) & product_nn(x2) & or_cc(x4,x1,x3) & mental_jj(x3) & artistic_jj(x3) & creation_nn(x3)).
(5) all e1 x1 x2 (make_vb(e1,x1,x2) $\leftrightarrow$ create_vb(e1,x1,x2) & manufacture_vb(e1,x1,x2) & man-made_jj(x2) & product_nn(x2)).

Furthermore, the question asks about the *Internet browser Mosiac*, while the candidate answer refers to *Mosaic*. To provide the knowledge that the *Internet browser Mosaic* refers to the same thing as *Mosaic*, the head of the complex nominal, *Internet browser Mosaic*, implies its remaining components.
(6) all x1 (mosaic_nn(x1) $\rightarrow$ internet_nn(x1) & browser_nn(x1)).
(7) all x1 x2 x3 x4 (mosaic_nn(x1) & internet_nn(x1) & browser_nn(x1) $\rightarrow$ nn_nnc(x1,x1,x1,x1)).

The next step is to build the Set of Support Axiom(s) for the Question. The question is negated to invoke the proof by contradiction
-(exists e1 x2 x3 x5 x6 (_organization_at(x2) & company_nn(x2) & create_vb(e1,x2,x6) & internet_nn(x3) & browser(x4) & mosaic_nn(x5) & nn_nnc(x6,x3,x4,x5))).
Next, link the answer type term, its modifiers, and any prepositional attachments to the answer type as a substitute for more refined named entity recognition.
all x1 (_organization_at(x1)$\rightarrow$company_nn(x1)).
It remains to create axioms for the ALF of the candidate answer and to start the proof.

The Proof
34 [] -develop_vb(x22,x7,x1)|make_vb(x22,x7,x1).
44 [] -make_vb(x23,x1,x2)|create_vb(x23,x1,x2).
74 [] -_organization_at(x1)|company_nn(x1).
121 [] -mosaic_nn(x1)|internet_nn(x1).
122 [] -mosaic_nn(x1)|browser_nn(x1).
123 [] -mosaic_nn(x1) | -internet_nn(x1) | -browser_nn(x1) | nn_nnc(x1,x1,x1,x1).
294 [] -_organization_at(x2) | -company_nn(x2) | -create_vb(x34,x2,x6) | -internet_nn(x3) | -browser_nn(x4) | -mosaic_nn(x5) | -nn_nnc(x6,x3,x4,x5).
299 [] mosaic_nn($c111).
300 [] develop_vb($c126,$c96,$c111).
302 [] national_nn($c103).
303 [] center_nn($c100).
304 [] for_nn($c99).
305 [] supercomputing_nn($c98).
306 [] application_nn($c97).
307 [] nn_nnc($c96,$c103,$c100,$c99,$c98,$c97).
332 [] -national_nn(x3) | -center_nn(x4) | -for_nn(x5) | -supercomputing_nn(x6) | -application_nn(x7) | -nn_nnc(x8,x3,x4,x5,x6,x7) | _organization_at(x8).
335 [hyper,299,122] browser_nn($c111).
336 [hyper,299,121] internet_nn($c111).
337 [hyper,336,123,299,335] nn_nnc($c111,$c111,$c111,$c111).
347 [hyper,300,34] make_vb($c126,$c96,$c111).
356 [hyper,347,44] create_vb($c126,$c96,$c111).
372 [hyper,332,302,303,304,305,306,307] _organization_at($c96).
373 [hyper,372,74] company_nn($c96).
374 [hyper,373,294,372,356,336,335,299,337] $F.
The numbers on the left hand side of the proof summary indicate the step number in the search, not the step number in the proof. Through step 332 we see that COGEX has selected all the axioms it needs to prove that the candidate answer is correct for the question posed to the QA system. Steps 335 through 374 show hyperresolutions that result in all the terms of the

question being derived in their positive form so the proof by contradiction succeeds, which is indicated by the $F in the final step and the hyperresolution of all the derived terms with the negated question from step 1 of the proof. The success of this proof boosts the candidate answer to the first position.

When the proof fails, we devised a way to incrementally relax some of the conditions that hinder the completion of the proof. This relaxation process puts weights on the proof such that proofs weaker than a predefined threshold are not accepted.

8 Results

COGEX was implemented and integrated into a state-of-the-art Question Answering system that participated in TREC 2002. All questions are attempted by the prover, but if the proof fails the QA system resorts to other answer extraction methods that were part of the system before the prover. Thus, some questions are answered by the QA system without the prover, some only by the prover and some by both the non-prover system and the prover. The complete system answered 415 questions out of 500 TREC 2002 questions. Of these, 206 were answered by COGEX but some of these answers were also provided by QA system without COGEX. A careful analysis indicates that the QA system without logic prover answered 317 questions and the prover can answer only 98 additional questions for which the system without prover failed. Table 1 summarizes these results.

Questions answered by the complete system	415
Questions answered by COGEX	206
Questions answered only by COGEX	98
Questions answered without COGEX	317

Table 1: Performance over 500 TREC 2002 questions

The added value of automated reasoning to the QA system is 30.9% (98/317). This represents a significant improvement in the performance of the logic prover for QA over the one reported in (Moldovan 2002). The failures of the prover are due primarily to the lack of linguistic axioms.

9 Discussion

A logic prover brings several advantages to question answering, but at a high cost. Some advantages are: the capability of pinpointing exact answers that otherwise will be missed, answer justification, and a quantifiable measure of how close a system is to providing an answer. However, the implementation of a QA logic prover is expensive as it requires logic representation of text, world knowledge axioms and a large number of linguistic axioms, that all take time to develop.

Acknowledgment
This work was supported in part by the ARDA AQUAINT program.

We wish to thank Mihai Surdeanu and Marius Pasca from LCC for their contribution to this work.

References

J. Hobbs, M. Stickel, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63 (1993), pp 69-142.

D. Moldovan, M. Pasca, S. Harabagiu and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of ACL 2002*.

Dan Moldovan and Adrian Noviscki. Lexical Chains for Questions. In *Proceedings of Coling 2002*.

Dan Moldovan and Vasile Rus. Logic Form Transformation of WordNet and its Applicability to Question Answering. In *Proceedings of ACL 2001*.

Ellen Voorhees. Overview of the TREC 2002 Question Answering Track. In *TREC 2002* http://trec.nist.gov

Larry Wos. Automated Reasoning, 33 Basic Research Problems. *Prentice Hall, 1988.*