AN APPROACH TO NATURAL LANGUAGE UNDERSTANDING

by

MICHAEL SCOTT MARLEN

B.S., Kansas State University, 2005
B.S., Kansas State University, 2005
M.S., Kansas State University, 2007

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Abstract

Natural Language understanding over a set of sentences or a document is a challenging problem. We approach this problem using semantic extraction and an ontology for answering questions based on the data. There is more information in a sentence than that found by extracting out the visible terms and their obvious relations between one another. It is the hidden information that is not seen that gives this solution the advantage over alternatives. This methodology was tested against the FraCas Test Suite with near perfect results (correct answers) for the sections that are the focus of this paper (Generalized Quantifiers, Plurals, Adjectives, Comparatives, Verbs, and Attitudes). The results indicate that extracting the visible semantics as well as the unseen semantics and their interrelations using an ontology to reason over it provides reliable and provable answers to questions validating this technology.

AN APPROACH TO NATURAL LANGUAGE UNDERSTANDING

by

MICHAEL SCOTT MARLEN

B.S., Kansas State University, 2005
B.S., Kansas State University, 2005
M.S., Kansas State University, 2007

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Dr. David Gustafson

Copyright

MICHAEL MARLEN

2014

Abstract

Natural Language understanding over a set of sentences or a document is a challenging problem. We approach this problem using semantic extraction and an ontology for answering questions based on the data. There is more information in a sentence than that found by extracting out the visible terms and their obvious relations between one another. It is the hidden information that is not seen that gives this solution the advantage over alternatives. This methodology was tested against the FraCas Test Suite with near perfect results (correct answers) for the sections that are the focus of this paper (Generalized Quantifiers, Plurals, Adjectives, Comparatives, Verbs, and Attitudes). The results indicate that extracting the visible semantics as well as the unseen semantics and their interrelations using an ontology to reason over it provides reliable and provable answers to questions validating this technology.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my friends for helping to delay me in finishing this paper in a timely manner.

Chapter 1 - Introduction

There has yet to be a system that is fully capable of understanding English. Natural Language understanding over a set of sentences or a document is a challenging problem. We approach this problem using semantic extraction and building an ontology for answering questions based on the data. There is more information in a sentence than found by extracting out the visible terms and their obvious relations between one another. Keeping track of inferences, quantities, inheritance, properties, and set related information is what gives this approach the advantage over alternatives. Our methodology was tested against the FraCas Test Suite with near perfect results for the sections: Generalized Quantifiers, Plurals, Adjectives, Comparatives, Verbs, and Attitudes. The results indicate that extracting the visible semantics as well as the unseen semantics and their interrelations using an ontology to logically provide reliable and provable answers to questions validating our methodology.

Brief Background

The system in this study is based on rules and pattern matching. Groups of words that matched a particular pattern have a set of rules associated with it as to how the word(s) can modify the knowledge base. The rules vary greatly, as some provide the capacity to infer, while others to provide reference information, relationships between nouns, or even when something occurred in time relative to other events that have occurred.

This is demonstrated in this paper using the FraCas Test Suite [Co96] problems that are presented in English. The FraCas Test Suite is widely regarded as the gold standard for Natural Language Understanding systems [Su03]. The current study provides research that takes it one step closer to solving the problems presented in the FraCas Test Suite, thus allowing multiple premises to be presented by using an open world framework.

Appropriate domain knowledge is essential with Natural Language Understanding, so background knowledge (additional premises) for select problems is provided as natural language since information necessary to answer the question is not always provided in the premises. This

is only done on problems for which the reader could utilize life experience to understand particular relationships, where a computer would not inherently know them. For this work, the assumption for this work is that there is sufficient domain knowledge to interpret the semantics of the propositions. This must occur for any test set in which the set of premises does not describe relationships generally understood to be known by a reader.

Hypothesis

The hypothesis for this study is that the extraction of semantic information, such as tracking inferences, quantities, inheritance, properties, and set related information from both premises and questions allows us to form an understanding.

Terms and Limitations

In this study, understanding is defined as providing the system with information (statements), asking questions about the information and expecting it to answers correctly. This behavior of answering a question correctly, we claim demonstrates understanding.

The system cannot understand anything more than the words it has definitions (rules) for. A word definition in terms of the systems' understanding tells the system what information it can extract from analyzing the word in a given input. In addition, it can demonstrate the behavior of a word in the system. (e.g. the word *is* can indicate a parent relationship between two nodes in the ontology).

This work considers only the subset of natural language (English) with which a parser can produce a valid grammar tree. This subset allows testing of what is possible for the current strategy while not having to deal with it failing due to a poor or incomplete parse from one or both of the parsers we use. While it is not the focus of this paper, we do have ways of ruling out particularly bad parses, such as when the StanfordNLP parser produces an incomplete parse tree, it is trivial to tell that this is a poor parse and can be ruled out.

Additionally, if the system presented in this paper could not understand a particular statement, the statement could be rephrased in such a way that the system can interpret it. Presently, we are rejecting the statements that the system's parsers do not interpret correctly.

The scope of this research is limited to the FraCaS Test Suite and the language contained within it. This system currently focuses on the language contained within the FraCaS Test Suite. The reason for this is to identify if it is possible to be able to generate sufficient knowledge to reason over to be able to answer the questions contained within the test suite and if so, it could be extended further to be tested against other test suites or even more real world scenarios.

Additional domain knowledge and general knowledge will be obtained from reading internet sources, such as Wikipedia. Currently, background domain information is provided to the system as part of the problem statements, such as those contained within the FraCaS Test Suite. The system presented in this paper is designed for areas in which domain information could be gathered in order to provide sufficient information to the system for satisfactory results.

Theoretical Framework

Because humans do not utilize a closed system to understand information, this system also does not utilize a closed system. Words with meaning have 'definitions as defined below' applied to them before run-time of the system.

What is the problem?

There has yet to be a system that is fully capable of understanding English. In this study, understanding is defined by being able to reason with a combination of first order logic like system and mapping an ontology onto a preexisting ontology built from premises or background knowledge.

Everyone is always looking for more intelligent results faster. A system able to analyze textual documents and answer 'high' level questions concerning the certainly saves time compared to current internet searches that must find terms that match in a document. A Google-

style search requires a user to read one or more documents to find the actual answer. This research introduces a semantic understanding search into the documents.

This study attempts to answer the question: Is it possible to represent the language found within the FraCaS Test suite so that the system would have sufficient capacity to provide a useful level of understanding given sufficient background information?

Why is this needed?

Machines must understand natural language. The primary objective of this study is to provide a new direction for Natural Language Understanding to head. In addition to just providing a new direction this direction must demonstrate validity and benefits over other techniques.

Chapter 2 - FraCaS Test Suite

The FraCaS Test Suite [Co96] contains 346 NLI problems, divided into nine sections, each section focused on a specific category of semantic phenomena [Ma14] and also appearing in [Ma08]. The nine different semantic phenomena include: Generalized Quantifiers, Plurals, Anaphora, Ellipsis, Adjectives, Comparatives, Temporal Reference, Verbs, and Attitudes. The following sub sections originate from the FraCaS Test Suite itself.

A majority of the categories below have elements from other categories. For example, quantifiers are present in every single category. These categories emphasize particular phenomena as opposed to only that phenomenon.

The FraCaS Test Suite does not cover all aspects of language. Take analogies as seen in Hofstadter, 2010. The FraCaS Test Suite also does not cover relationships between words and a physical sense of the word. As an example, what does it mean for something to be a chair? In addition, the FraCaS Test Suite claims that it does in fact contain imperfect grammatical sentences, with even four problems missing the question all together.

Relevance

Bunt uses Bill MacCartney's work on natural logic [Ma14], to obtain an accuracy of 70% and a precision of 89% on the FraCaS data set [Bu14].

Chatzikyriakidis conducted experiments based on Modern Type Theories using the FraCaS Test Suite that look promising [Ch13a]. Chatzikyriakidis showed the basics for Modern Type Theory and how you could convert a few small examples from the FraCaS Test Suite into formal semantics of modern type theory [Ch13b].

Cabrio mentioned that the FraCaS Test Suite is worthwhile because it provides "textbook examples of semantic phenomena" problems despite it being a small data set relative to the Recognizing Textual Entailment (RTE) data which is more geared towards entailment checking

[Ca13]. This dataset is certainly a very popular dataset and continues to grow each year encompassing more and more concepts.

The RTE data set contains pairs of sentences, no questions about the information. Given the two sentences, the job with this data set is to determine if the first sentence entails the second sentence. The second sentence could be inferred given the first if it did in fact entail the second sentence.

In 2012, Ljunglöf conducted research on parsing of multiple parallel context free grammars using the FraCaS Test Suite [Lj12]. This work argued how four published parsing algorithms for parallel multiple context-free grammar were similar. They went on to make improvements by adding in their own modifications even using the Swedish translation of the FraCaS Test Suite done by Ljunglöf and Siverbo, [Lj11].

Koller provides a state-of-the-art overview of everything related to computational semantics. He also discussed the FraCaS Test Suite, stating "The FraCaS testsuite was hand-crafted to cover challenging semantic phenomena (such as quantifiers, plural, anaphora, temporal reference, and attitudes), while minimizing the impact of problems like syntactic complexity and word-sense ambiguity [Ko12]. He criticized the data suite because real-world language contains more word-sense ambiguity, and effectively limiting the FraCaS Test Suite [Ko12]. Koller also says that systems utilizing the FraCaS Test Suite have the capacity to compare a systems results with the gold standard [Ko12].

Generalized Quantifiers

Five subsections are present in this particular category. The subsections are: Conservativity, Monotonicity (upwards on second argument), Monotonicity (downwards on second argument), Monotonicity (upwards on first argument), and Monotonicity (downwards on first argument).

Barwise (1981) showed that first order language is not sufficient to represent all the types of quantifiers that exist. LA Zadeh (1983) utilized a fuzzy logic approach to representing and understanding quantifiers. Terms such as *few, some, approximately ten, etc*. are not exact quantities and are well suited to being considered fuzzy.

Barwise and Cooper and van Bentham showed various properties, and forms of generalized quantifiers and Partee builds upon this and adds in a type driven resolution to noun phrases with quantifiers [Pa87].

All noun phrases contain an implicit, or explicit quantifier in the noun phrase, so noun phrases can be given type information as shown in (Montague, 1974). Partee builds upon this work by allowing noun phrases to belong to a family tree of types rather than a single type [Pa87].

Thijsee (1983) extended work on universal quantifiers by providing examples of valid and invalid universal quantifiers.

The Conservativity section provides information that all As are Bs and the question asks if all As who are Bs. Figure 2-1 shows an example problem from this section.

fracas-009	answer: yes
P1	Several great tenors are British.
Q	Are there great tenors who are British?

Figure 2-1 FraCaS Test Suite Problem 9

Montonicity (upwards on second argument) are of the form, As are Bs and all Bs are Cs and the question asks As are Cs. Figure 2-2 presents an example problem show-casing the structure of upwards on the second argument. In this problem, you are given information about a specific instance of a Nobel prize is provided and asked about a generic version of it.

fracas-017	answer: yes
P1	An Irishman won the Nobel prize for literature.
Q	Did an Irishman win a Nobel prize?

Figure 2-2 FraCaS Test Suite Problem 17

Montonicity (downwards on second argument) are of the form, As are Bs and all Bs are Cs and the question asks As are Cs. Figure 2-3 presents an example problem show casing the downwards structure on the second argument. This problem offers a more specific instance of a generic Nobel prize.

fracas-033	answer: unknown
P1	An Irishman won a Nobel prize.
Q	Did an Irishman win the Nobel prize for literature?

Figure 2-3 FraCaS Test Suite Problem 33

Monotonicity (upwards on first argument) are of the form, As are Bs and all As are Cs where the question is asks are Cs Bs? Figure 2-4 showcases a problem that is of this particular form. The premise contains a specific type of delegate and "some" number of them while asking about any delegate of any type.

fracas-055	answer: yes
P1	Some Irish delegates finished the survey on time.
Q	Did any delegates finish the survey on time?

Figure 2-4 FraCas Test Suite Problem 55

Monotonicity (downwards on first argument) are of the form, As are Bs and all As are Cs where the question asks are Cs Bs? Figure 2-5 also showcases a problem that is of this particular form, with the exception that the premise provides a generic delegate while the question asks about a Scandinavian delegate.

fracas-070	answer: no
P1	No delegate finished the report on time.
Q	Did any Scandinavian delegate finish the report on time?

Figure 2-5 FraCas Test Suite Problem 70

Plurals

Plurals contains six sections: Conjoined Noun Phrases, Definite Plurals, Bare Plurals, Dependent Plurals, Negated Plurals, Collective and Distributive Plurals. This section effectively follows the previous section except that noun phrases reference multiples or several nouns are referenced.

Conjoined Noun Phrases contain multiple nouns joined by conjuctive words, such as *and* or *or*. One example problem shown in this section of the FraCaS Test Suite is shown in Figure 2-6.

fracas-082	answer: yes
P1	Smith, Jones and several lawyers signed the contract.
Q	Did Jones sign the contract?
H	Jones signed the contract.

Figure 2-6 FraCaS Test Suit Problem 82

"Definite plurals can often be non-anaphoric and behave like universally quantified noun phrases (90). However, as with (generic) bare plurals, the force of the quantification can also be less than universal (91). Whether this lessening of quantificational force is due to the noun phrase or to the predicate of which the NP is an argument is unclear (92, 93)." from [Co96].

fracas-095	answer: unknown
P1	The Ancient Greeks were noted philosophers.
Q	Was every Ancient Greek a noted philosopher?
H	Every Ancient Greek was a noted philosopher.

Figure 2-7 FraCaS Test Suit Problem 95

"Bare plurals can exhibit existential, (quasi) universal, generic or dependent plural behaviour." [Co96]. A problem showcasing one of these is seen in Figure 2-8.

fracas-098	answer: unknown
P1	Software faults were blamed for the system failure.
P2	Bug # 32-985 is a known software fault.
Q	Was Bug # 32-985 blamed for the system failure?
H	Bug # 32-985 was blamed for the system failure.
Why	*Existential interpretation: not every software fault contributed to the failure.*

Figure 2-8 FraCaS Test Suite Problem 98

The next section is Dependent Plurals, which describes an entire set/group and then define a particular element within the group as shown in Figure 2-9.

fracas-103	answer: yes
P1	All APCOM managers have company cars.
P2	Jones is an APCOM manager.
Q	Does Jones have a company car?
H	Jones has a company car.

Figure 2-9 FraCaS Test Suite Problem 103

Negated Plurals is the next section, which the FraCaS Test Suite describes as an accountant attending a meeting. The question part varies asking about if 0 or more accountants attended the meeting. An example is shown in Figure 2-10.

fracas-106	answer: no
P1	Just one accountant attended the meeting.
Q	Did no accountant attend the meeting?
H	No accountant attended the meeting.

Figure 2-10 FraCaS Test Suite Problem 106

The final section in the Plurals section is Collective and Distrubitive Plurals. This section requires one to read of one of two ways to illicit an answer, as shown in Figure 2-11.

fracas-111	answer: yes **
P1	Smith signed one contract.
P2	Jones signed another contract.
Q	Did Smith and Jones sign two contracts?
H	Smith and Jones signed two contracts.
A	Yes, on a collective/cumulative reading of the conclusion.

Figure 2-11 FraCaS Test Suite Problem 111

Several methods are available to handle plurals in natural language. Kamp (1993) demonstrated some methods for handling each of the various subsections above, in theory.

Lapata, and Keller (2004) use both singular and plural word forms of the word when querying their information base to extract results. They converted a word or phrase to a singular version (stemming) of it and use that to search.

Anaphora

Anaphora is the "repetition of a word or words at the beginning of two or more successive verses, clauses, or sentences" from Dictionary.com Unabridged. In this section, six subsections are present: Intra-Sentential, Inter-Sentential, Plural Anaphora, E-type and Donkey Pronouns, Functional and Subsectional, and Simple Reflexives. Each subsection showcases a particular phenomenon with anaphora.

The first section, Intra-Sentenial, forces you to make inferences through the use of pronouns within the sentence. If a deduction can be made regarding to whom the pronoun is referring, then verbs with multiple meanings must be able to be extracted, as demonstrated in Figure 2-12.

fracas-116	answer: yes
P1	Mary used her workstation.
Q	Is Mary female?
H	Mary is female.

Figure 2-12 FraCaS Test Suite Problem 116

The second section, Inter-Sentential, forces you to make inferences through pronouns using all premises. If a deduction can be made regarding to whom the pronoun is referring, then all relationships must still be extracted from verbs that have multiple meetings, as shown in Figure 2-13.

fracas-120	answer: yes
P1	Smith attended a meeting.
P2	She chaired it.
Q	Did Smith chair a meeting?
H	Smith chaired a meeting.

Figure 2-13 FraCaS Test Suite Problem 120

For the Plural Anaphora section, handles making inferences through the use of plural pronouns as shown in Figure 2-14.

fracas-124	answer: yes
P1	Two out of ten machines are missing.
P2	They have been removed.
Q	Have two machines been removed?
H	Two machines have been removed.

Figure 2-14 FraCaS Test Suite Problem 124

E-Type and Donkey pronouns are bound by semantics but not the syntax [Ga01]. Heim (1990) showed how to handle these 'donkey sentences,', and an example problem is presented in Figure 2-15.

fracas-134	answer: yes
P1	Every customer who owns a computer has a service contract for it.
P2	MFI is a customer that owns exactly one computer.
Q	Does MFI have a service contract for all its computers?
H	MFI has a service contract for all its computers.
Why	*Donkey sentence*

Figure 2-15 FraCaS Test Suite Problem 134

Functional and Subsectional contains only one problem since these types of problems typically require substantial background knowledge. The only problem within this section is demonstrated in Figure 2-16.

fracas-138		answer: yes
P1	Every report has a cover page.	
P2	R-95-103 is a report.	
P3	Smith signed the cover page.	
Q	Did Smith sign the cover page of R-95-103?	
H	Smith signed the cover page of R-95-103.	

Figure 2-16 FraCaS Test Suite Problem 138

The final section in Anaphora is Simple Reflexives. These problems use pronouns to describe a property or action of the subject or object of the sentence as showcased in Figure 2-17.

fracas-140	answer: yes
P1	John said Bill had hurt himself.
Q	Did John say Bill had been hurt?
H	John said Bill had been hurt.

Figure 2-17 FraCaS Test Suite Problem 140

Anaphora resolution requires the integrated application of syntactic, semantic, and pragmatic knowledge [Ca88]. Carbonell demonstrated a multi-strategy approach to resolving anaphora by utilizing eight different strategies that are only capable of handling only a small subset of anaphora using: local anaphor constraints, case-role semantic constraints, pre/post condition constraints, case-role persistence preference, semantic alignment preference, syntactic parallelism preference, syntactic topicalization preference, and intersentiantial recency preference [Ca88].

Another approach to anaphora resolution is a statistical approach by [Ge98], in which he calculates "the distance between the pronoun and the proposed antecedent, gender/number/animaticity of the proposed antecedent, governing head information and noun phrase repetition." [Ge98].

Hardt (1993) presented a series of required steps in order to understand some types of ellipsis. He strategically deconstructed a sentence down into a parse tree and extract as many sentence properties as possible. In his example, "Arthur worked.", is converted as shown in Figure 2-18.

```
[issue = [predicate = [wordstem = work,
                       cat = verb],
          arg1 = [wordstem = arthur,
                  cat = propernoun,
                  role = subject]],
 type = declarative,
 polarity = positive,
 tense = past,
 tenseaspect = simple,
 progressive = no]
```

Figure 2-18 Hardt' Initial Representation

Hardt also extracted various entities and bound them to variables in order to look up references, similar to in the methodology presented in this paper [Ha93].

Additional approaches to anaphora resolution to resolve from using expectation maximization by [Ch09], through the use of pattern matching in hand crafted lexical resources, [Ma03] achieve state of the art, results of non-pronimal anaphora.

Ellipsis

This section in the FraCaS Test Suite contains nine sections: VP Ellipsis, Gapping, One Anaphora, Sluicing, Phrasal Ellipsis, Antecedent Contained Deletion, Configurational Effects, Ellipsis and Anaphora, Ellipsis and Quantification.

VP Ellipsis uses prior context to apply a previous situation to the current premise, as demonstrated in Figure 2-19. Hardt, showed that these expressions must be determined at some stage during processing as opposed to pre-determined [Ha93].

fracas-142	answer: yes
P1	John spoke to Mary.
P2	So did Bill.
Q	Did Bill speak to Mary?
H	Bill spoke to Mary.
Why	*Basic example.*

Figure 2-19 FraCaS Test Suite Problem 142

The Gapping section of ellipsis lack the verb from the second clause and assumes the verb is implied given prior clause. The challenge here is to keep track of the verb part of speech to maintain the 'when'. This is just another example of the interplay between multiple categories within the FraCaS Test Suite. Johnson 2009 states "That Gapping involves, or is related to, across-the-board movement has several precedents. See, e.g., Goodall (1987), Zoerner (1995) and Steedman (1990, 1996).". An example of a gapping ellipsis as demonstrated in Figure 2-20.

fracas-150	answer: yes
P1	John went to Paris by car, and Bill by train.
Q	Did Bill go to Paris by train?
H	Bill went to Paris by train.
Why	*Basic example*

Figure 2-20 FraCaS Test Suite Problem 150

One Anaphora sub-section applies a previous situation to a new subject in which an inference must be made regarding what the anaphora is referencing. Techniques for doing this are discussed in [Mi96][Mi98][Mi07]. This section is similar to Anaphora, as shown in Figure 2-21.

Fracas-155	answer: yes
P1	John owns a car.
P2	Bill owns one too.
Q	Does Bill own a car?
H	Bill owns a car.
Why	*Basic example*

Figure 2-21 FraCaS Test Suite Problem 155

The sluicing sub-section contains only one problem, shown in Figure 2-22. Sluicing has been investigated by [Ro69][Le82][Ri82][Ch87][Ch95][Lo95][Ro98][La01][Me01][Me08]. Despite all of the various ways of trying to handle the various types of sluicing, there is still only one problem that contains sluicing within this entire data set.

fracas-163	answer: no
P1	John had his paper accepted.
P2	Bill doesn't know why.
Q	Does Bill know why John had his paper accepted?
H	Bill knows why John had his paper accepted.

Figure 2-22 FraCaS Test Suite Problem 163

Phrasal Ellipsis are often continuations of a prior statement as shown in Figure 2-23.

fracas-164	answer: yes
P1	John spoke to Mary.
P2	And to Sue.
Q	Did John speak to Sue?
H	John spoke to Sue.
Why	*PP ellipsis (subcategorized)*

Figure 2-23 FraCaS Test Suite Problem 164

"Antecedent contained deletion is a notorious problem for copying approaches to ellipsis, since the antecedent clause contains the ellipsis and some way must be found of removing it from the copy. [Co96].". An example of antecedent contained deletions is seen in Figure 2-24.

fracas-173	answer: yes
P1	Bill spoke to everyone that John did.
P2	John spoke to Mary.
Q	Did Bill speak to Mary?
H	Bill spoke to Mary.

Figure 2-24 FraCaS Test Suite Problem 173

Configurational Effects: "There are a number of syntactic and other configurational constraints on what can constitute the antecedent to an ellipsis. These constraints varying

depending on the type of ellipsis (VP, phrasal, gapping, etc).” [Co96]. An example of one configuration is seen in Figure 2-25.

fracas-175	answer: yes **
P1	John said Mary wrote a report, and Bill did too.
Q	Did Bill say Mary wrote a report?
H	Bill said Mary wrote a report.
A	Yes, on one possible reading/parse

Figure 2-25 FraCaS Test Suite Problem 175

Ellipsis and Anaphora: “The following inferences illustrate differences between strict and sloppy interpretations of anaphors in elliptical clauses.” [Co96]. An example in this sub-section is seen in Figure 2-26.

fracas-182	answer: yes **
P1	Smith represents his company and so does Jones.
Q	Does Jones represent Jones' company?
H	Jones represents Jones' company.
A	Yes, on one reading
Why	*Sloppy identity*

Figure 2-26 FraCaS Test Suite Problem 182

Ellipsis and Quantification is often tricky to illustrate [Co96]. Reading the first premise from Figure 2-27, and understanding how the methodology that is presented later works, clearly reveal one auditor and one lawyer.

fracas-196	answer: yes
P1	A lawyer signed every report, and so did an auditor.
P2	That is, there was one lawyer who signed all the reports.
Q	Was there one auditor who signed all the reports?
H	There was one auditor who signed all the reports.

Figure 2-27 FraCaS Test Suite Problem 196

Dalrymple (1991) provided a methodology for interpreting possibilities generated by an ellipsis covering half the sub sections within the Ellipsis category by using the parallel phrase structure to determine what is missing.

Adjectives

Six sub-sections exist for Adjectives: Affirmative and Non-Affirmative, No Comparison Class, Opposites, Extensional Comparison Classes, Extensional and Intensional Comparison Classes, and Default Comparison Classes.

"Affirmative adjectives map the denotation of the predicate they modify onto a subset of the denotation. So for example, an old man is a man." [Co96]. Although a majority of adjectives can be classified as the affirmative category, not all adjectives fit this description. Not all adjectives apply to the parent class. A list of adjectives, with a flag that describes if an adjective is affirmative or not is sufficient when combined with the methodology in this paper. An example of affirmative adjective is shown in Figure 2-28.

fracas-197	answer: yes
P1	John has a genuine diamond.
Q	Does John have a diamond?
H	John has a diamond.
Why	*Affirmative adjectives: Adj N entails N*

Figure 2-28 FraCas Test Suite Problem 197

No Comparison Class answers the question that asks if A is a type of B, does the adjective that applies to A, also apply to B, as demonstrated in Figure 2-29.

fracas-202	answer: yes
P1	Every mammal is an animal.
Q	Is every four-legged mammal a four-legged animal?
H	Every four-legged mammal is a four-legged animal.
Why	*[N1 entails N2] entails [Adj(N1) entails Adj(N2)]*

Figure 2-29 FraCas Test Suite Problem 202

Opposites adjectives showcase adjectives that have opposites when applied to the same comparison class (Cooper et al., 96) as shown in Figure 2-30.

fracas-204	answer: no
P1	Mickey is a small animal.
Q	Is Mickey a large animal?
H	Mickey is a large animal.
Why	*Small(N) entails !Large(N)*

Figure 2-30 FraCas Test Suite Problem 204

Extensional Comparison Classes subsection describes an instance of some base class and assigns an adjective that describes a particular instance of that base class such as demonstrated in Figure 2-31.

fracas-210	answer: no
P1	All mice are small animals.
P2	Mickey is a large mouse.
Q	Is Mickey a large animal?
H	Mickey is a large animal.

Figure 2-31 FraCas Test Suite Problem 210

Some adjectives require an "intensional" comparison class meaning that various inferences may follow when two distinct but co-extensive predicates provide the comparison class [Co96] An example of this is shown in Figure 2-32.

fracas-214	answer: yes
P1	All legal authorities are law lecturers.
P2	All law lecturers are legal authorities.
Q	Are all fat legal authorities fat law lecturers?
H	All fat legal authorities are fat law lecturers.
Why	*Extensional comparison class*

Figure 2-32 FraCas Test Suite Problem 214

Default Comparison Classes depend on the adjective and parent class. Some adjective and parent class types can apply to an instance of a parent class such as is shown in Figure 2-33.

fracas-218	answer: yes
P1	Kim is a clever person.
Q	Is Kim clever?
H	Kim is clever.

Figure 2-33 FraCas Test Suite Problem 218

Cimiano (2007) captured adjectives and placed them into an OWL-based ontology. Observation of the adjective and the rules, understanding and ability to assign adjectives as properties, or understand that the adjective *long* and associate it with having a height/measurement, similar to an integer.

Comparatives

Six sub-sections are also present within Comparatives: Phrasal Comparatives, Clausal Complements, Measure Phrases, Differential Comparatives, Attributive Comparatives, and Comparatives and Quantifiers. Each section looks at words or phrases in order to create a relative ordering.

No research is readily in regards to comparatives in the computational sense. Therefore, for this section, an example of each various sub-section is provided.

An example of Phrasal Comparatives is shown in Figure 2-34.

fracas-220	answer: yes
P1	The PC-6082 is faster than the ITEL-XZ.
P2	The ITEL-XZ is fast.
Q	Is the PC-6082 fast?
H	The PC-6082 is fast.

Figure 2-34 FraCas Test Suite Problem 220

An example of Clausal Complements is shown in Figure 2-35.

fracas-240	answer: unknown
P1	ITEL won more orders than APCOM lost.
Q	Did APCOM lose some orders?
H	APCOM lost some orders.

Figure 2-35 FraCas Test Suite Problem 240

An example of Measure Phrases is shown in Figure 2-36. In this example, it is all about keeping track of the relative speed.

fracas-242	answer: yes
P1	The PC-6082 is faster than 500 MIPS.
P2	The ITEL-ZX is slower than 500 MIPS.
Q	Is the PC-6082 faster than the ITEL-ZX?
H	The PC-6082 is faster than the ITEL-ZX.

Figure 2-36 FraCas Test Suite Problem 242

An example of Differential Comparatives is shown in Figure 2-37.

fracas-243	answer: yes
P1	ITEL sold 3000 more computers than APCOM.
P2	APCOM sold exactly 2500 computers.
Q	Did ITEL sell 5500 computers?
H	ITEL sold 5500 computers.

Figure 2-37 FraCas Test Suite Problem 243

An example of Attributive Comparatives is shown in Figure 2-38

fracas-244	answer: yes **
P1	APCOM has a more important customer than ITEL.
Q	Does APCOM have a more important customer than ITEL is?
H	APCOM has a more important customer than ITEL is.
A	Yes, on one reading of the premise

Figure 2-38 FraCas Test Suite Problem 244

An example of Comparatives and Quantifiers is shown in Figure 2-39.

fracas-246	answer: yes
P1	The PC-6082 is faster than every ITEL computer.
P2	The ITEL-ZX is an ITEL computer.
Q	Is the PC-6082 faster than the ITEL-ZX?
H	The PC-6082 is faster than the ITEL-ZX.

Figure 2-39 FraCas Test Suite Problem 246

Temporal Reference

The temporal section contains five sub-sections: Standard Use of Tenses, Temporal Adverbials, Anaphoric Dimensions, Adverbs of Quantification, and Some more Complex Examples. "Inference patterns involving temporal reference are complicated by the interplay between tense, aspectual information, lexical semantics, defeasible interpretation principles such as narrative progression, rhetorical relations, a theory of action and causation, world knowledge, interaction between plurality, genericity and temporal/aspectual phenomena etc. Some of the inferences are very basic, some are more involved. The more complex examples give ample illustration of the fact that temporal phenomena are usually discourse phenomena." [Co96].

Standard Use of Tenses is a straightforward section in regards to verb tenses and conversion between past, present, and future. An example of this is shown in Figure 2-40.

racas-251	answer: yes
P1	ITEL has a factory in Birmingham.
Q	Does ITEL currently have a factory in Birmingham?
H	ITEL currently has a factory in Birmingham.

Figure 2-40 FraCas Test Suite Problem 251

Temporal Adverbials require understanding of the concept of time, including relative points in time, such as *today* and *yesterday*. In Figure 2-41, dates must be understood in order to answer the question.

fracas-259	answer: yes
P1	The conference started on July 4th, 1994.
P2	It lasted 2 days.
Q	Was the conference over on July 8th, 1994?
H	The conference was over on July 8th, 1994.

Figure 2-41 FraCas Test Suite Problem 259

The Anaphoric Dimensions sub-section utilizes anaphoric inferences to describe events that occur in a particular order in time as shown in Figure 2-42.

fracas-311	answer: yes
P1	Smith had left the house at a quarter past five.
P2	Then she took a taxi to the station.
Q	Did Smith leave the house before she took a taxi to the station?
H	Smith left the house before she took a taxi to the station.

Figure 2-42 FraCas Test Suite Problem 311

The Adverbs of Quantification sub-section attaches an adverb to a situation and through the context of this adverb, the question is understood as shown Figure 2-43. In this section, the incorrect verb tense is used in the question and notated in the note section on both of the problems in this sub-section.

fracas-312	answer: yes
P1	ITEL always delivers reports late.
P2	In 1993 ITEL delivered reports.
Q	Did ITEL delivered reports late in 1993?
H	ITEL delivered reports late in 1993.
Note	*Sic: the original did have "delivered" in the question.*

Figure 2-43 FraCas Test Suite Problem 312

Some more Complex Examples is the inclusive sub-section that accommodates problems that do not fit other temporal reference sub-sections. An example of a complex temporal reference problem as shown in Figure 2-44.

fracas-314	answer: yes

P1	Smith arrived in Paris on the 5th of May, 1995.
P2	Today is the 15th of May, 1995.
P3	She is still in Paris.
Q	Was Smith in Paris on the 7th of May, 1995?
H	Smith was in Paris on the 7th of May, 1995.

Figure 2-44 FraCas Test Suite Problem 240

Verbs

Two sections are present within the Verbs section: Aspectual Classes and Distributive and Collective Predication.

Aspectual Classes involve reading the verb and fully understanding the meaning within the context. For example, in Figure 2-45, the verb *finished* also means *built*.

fracas-326	answer: yes
P1	ITEL built MTALK in 1993.
Q	Did ITEL finish MTALK in 1993?
H	ITEL finished MTALK in 1993.

Figure 2-45 FraCas Test Suite Problem 326

Distributive and Collective Predication deals with how a verb applies to various parts of a complex noun phrase, as shown in Figure 2-46.

fracas-331	answer: yes
P1	Smith and Jones left the meeting.
Q	Did Smith leave the meeting?
H	Smith left the meeting.

Figure 2-46 FraCas Test Suite Problem 331

Attitudes

This section contains two sub-sections. The first section Epistemic, Intentional and Reportive Attitudes deals with directly inferable information. The second sub-section, Preceptive Attitudes: "See" with Bare Infinitive Complements, contains many sub-sections

which are described later in this paper. Problems contained within this section deal with the verb '*see*' and what is observable by the individual on a situation.

Epistemic, Intentional and Reportive Attitudes provide information typically known or believed by an individual about a particular fact. Given that information, must decide how that effects a particular statement. Figure 2-47 provides an example of this is shown where Smith states something he knows to be true. In this example, the assumption must be made that Smith is not misleading or lying.

fracas-334	answer: yes
P1	Smith knew that ITEL had won the contract in 1992.
Q	Did ITEL win the contract in 1992?
H	ITEL won the contract in 1992.

Figure 2-47 FraCas Test Suite Problem 334

The Preceptive Attitudes: "See" with Bare Infinitive Complements sub-section contains six problems within six sub-categories of attitudes each involving the verb "*see*". Each problem contains a premise that can be deconstructed to contain a sub-sentence in its own right, seen by any individual. One example of a preeptive attitude is shown in Figure 2-48.

fracas-340	answer: unknown
P1	Smith saw Jones sign the contract.
P2	If Jones signed the contract, his heart was beating.
Q	Did Smith see Jones' heart beat?
H	Smith saw Jones' heart beat.

Figure 2-48 FraCas Test Suite Problem 340

Wilson, and Wiebe (2005) demonstrated several ways to handle various attitudes. First, the type of attitude must be identified, of which they show that there are seven types. The seven types of attitudes including: Positive Attitudes, Negative Attitudes, Positive Arguing, Negative Arguing, Positive Intensions, and Negative Intensions, and Other [Wi05]. Depending on the type of attitude, a private state/frame for the phrase must be developed. Additional attributes,

such as the target link, intensity level, and properties must also be extracted, as defined in Figure 2-49 [Wi05].

Inferred	True, if attitude is inferred
Sarcastic	True, if attitude is realized through sarcasm
Repetition	True, if the attitude is realized through the repetition of words, phrases, or syntax
Contrast	True, if the attitude is realized through contrast with another attitude

Figure 2-49 Attitude Properties

Chapter 3 - Background

Input statements can be analyzed and information extracted in a variety of ways. The ability to segment information allows the extracted information to be provided to a NLP understanding system. In this chapter, various ways to look at input statements and turning them into different chunks, parses, or structures are considered. The FraCaS Test Suite is also discussed and show how various groups have tried tackling the sub problems within the test suite.

After information is extracted from the input statements, the information must be represented in a useful way. Many ways to represent information or knowledge exist, and each method has distinct advantages and disadvantages. Finally, brief comments on stop words conclude this chapter.

Part of the rationale behind pursuing a rule based method is that, if you can find a set of operators in language, which would let you understand or connect information together, to effectively learn new words that are some combination of those operators. It would be much simpler to make additions to the system and understand how they affect the system. With the argument that, say a neural net, modifying or adding neurons is a non-trivial process to understanding what is actually happening.

Information Extraction Techniques

The seven techniques information extraction techniques discussed in this chapter are 1) simple part of speech tagging, 2) shallow parsing, 3) deep parsing, 4) semantic role labeling, 5) co-reference resolution, 6) relations, and 7) named-entity extraction. Each technique extracts or adds information in different ways.

Early efforts focused on Part of speech tagging (POS tagging) simply tags each word in the sentence with the part of speech it thinks the word is. This is a challenging problem because, some words can be used in different parts of speech. There is limited valued in this as I claim more value is derived from identifying what words group together [Sc94].

Shallow parsing identifies word groupings to a particular part of speech such as noun phrases, verb phrases, but it does not provide internal representation of how words connect/relate within that structure itself or how it relates to other structures in the sentence. Shallow parsing techniques have shown to be quite successful using various methodologies to break apart a sentence into its various chunks or phrases [Sh03][Mo02]. Deep Parsing simply provides much more detail in terms of syntactic and semantic dependencies [Ka04]. This type of parse is used when more meaning is trying to be extracted.

"Coreference resolution is the process of determining whether two expressions in natural language refer to the same entity in the world" [So01]. I absolutely agree with [So01] when he states "The ability to link coreferring noun phrases both within and across sentences is critical to discourse analysis and language understanding in general."

Relationship extraction often unites two entities, in addition to associating modifiers or conditions with these relationships [Ng07]. Relationship extraction builds on other types of parses, such as shallow or deep parses as shown in [Ze03].

Named entity extraction attempts to identify various entities and assign them a category. Correct identification and understanding of what an entity belongs to is essential, as shown in [Et05][Co02]. Understanding the entity and corresponding category provides increased understanding of the entity.

Many systems employ a pattern matching approach as part of the strategy to extract information, as demonstrated in [Gr97].

Significant recent work in sentiment analysis techniques has been conducted [Pa04][Wi05][Es06][Zh06][Pa08]. Sentiment analysis aims to understand subjective information, such as user's emotions, perhaps towards a product or a company. The use of sentiment analysis techniques increase understanding of a positive or negative review of a product, movie, company, or news [Go07][Pa08]. Most analysis techniques use a sentiment

lexicon which has words that are rated either by positive, negative, or neutral rating. Other categories are possible, such as the indication of anger, sadness, or happiness. Regardless of the content category, information is lost, but the sentiment polarity or mood is generally understood. However, limits to this, as when the result is compared to humans and their sentiment understanding of the material, humans tend to agree 79% of the time about the sentiment polarity of a particular article [Og10].

Knowledge Representation and Understanding

There is work in many areas [No04][Re05], from statistical analysis of language [Ma99], to converting the text to predicate logic [Sc82], to Natural Logic [La73][Ma07]. Each type of system has benefits and drawbacks. No solution to the grand challenge problem of understanding language currently exists. The first type of system, the statistical analysis doesn't understand why it knows what it knows only that it is statistically likely. Complications also with prepositional logic type systems because these systems only work in the realm of true and false and do not allow for non-Boolean related queries. Natural Logic requires both premises and a working hypothesis and tries to find an answer through entailment checking the validity of the statement.

In 2011, a NLP system called Watson that aired on the Jeopardy! show to compete against the upper echelon of players that previously competed on the show [Fe10]. In order to compete at this level, Watson had to understand language. Using understanding capabilities it to "read" a large corpus of information from every category that Jeopardy! could throw at it. It won! (or something) With its win in Jeopardy!, the claim can be made that this NLP system understands language very well. To understand language, Watson uses a large set of rules and pattern matching to come to understand the material and the question.

Watson system utilizes shallow parses, deep parses [Mc90], logical forms, semantic role labels, coreference, relations, and named entities, as well as specific kinds of analysis for question answering [Fe10].

Knowledge can be represented in many ways, each with their distinct advantages and disadvantages. In this research, five core types of representation are discusses, including Formal Logic, Statistical Based, Natural Logic, Ontology Based, and semantic modeling methodologies. It is important to understand what it is, what extensions and systems use it, and specific methodological benefits to using a methodology and drawbacks.

Formal Logic

Many types of formal logic exist, including first order logic, temporal logics, and modal logics. Each type of formal logic offers distinct features. For example, compared to first order logic, temporal logic includes the addition of a time component [Me96]. Each formal logic results in true or false answers. If all questions could be asked in a language able to be converted into a formal logic with yes or no type questions these types of system would prove to be quite valuable. Unfortunately English's phrases, sayings, or entire statements are not able to be converted. However, several papers demonstrated how to convert English to a formal logic such as [Sc82]. In addition to the challenge or impossibility of modeling English into this formal logic, it does not contain sufficient amount of operators necessary to model basic arithmetic.

Automated theorem provers are utilized for the logics such as first order logic, including the conversion of higher order logics to first order logic [Hu03]. Unfortunately, theorem provers can get stuck with some inputs, or take unreasonable amounts of time to prove particular fact, thus causing concern. However, relatively small inputs do not have this issue.

To use a formal logic of any type, it needs to be expressive enough to handle the natural language that would be presented to the system. Using formal logic, you would need to a system that can handle all types of quantifiers, including the ones that cannot be handled by a first order logic. Additionally, it needs to be able to handle all of the exceptions that the language contains along with all other concepts that may be derived or presented in the language.

Statistical Based

Statistical methods have been applied to derive the parts of speech of text, build parse trees, identify a word sense, and understand meaning of a word as demonstrated in [Ch93][Br97][Ma99][Ju00].

Understanding a words meaning via statistical methods may require counting the number of times a word is seen and surrounding words in order to calculate a distance between other situations to see what it most closely relates to and then cluster those into a particular concept as is done in [Ch93].

Significant research has been conducted on information retrieval in the NLP domain of clustering concepts [Bi00][Ra05][Di06].

The use of statistical methods make a system flexible in terms of letting you know what a word may mean in context of what it has seen so far. However, that is only a function, perhaps based of word counts and distance to situations it has seen prior, it does not necessarily translate into a higher level of understanding.

Methodology presented in this paper utilizes statistical based parsers, OpenNLP and StanfordNLP. Minor statistics may come into play, if you consider keeping track of counts in order to maintain a most used to assist in coreference resolution more speedily.

Natural Logic

Natural Logic, a model for inferring and interpreting language [Be87]. This has been developed over the years by [La73][Br90][Fy03][Ma07][Ma08][Ma14]. "'Natural Logic' is a somewhat loose, but popular and suggestive term for recurrent attempts over the last decades at describing basic patterns of human reasoning directly in natural language without the intermediate of some formal system" [va08]. This type of system works on patterns 'surface' syntax [va08]. Natural Logic attempts to encompass reasoning capacities and understand everything that can be stated in a language.

MacCartney (2007) implemented the first computational model of Natural Logic working towards textual inference. In the NatLog system defined in [Ma07][Ma08][Ma14], MacCartney defined relations between words, phrases, and sentences in order to define entailment to answer questions with a yes/no/unknown answer.

The NatLog system has three phases: "(1) linguistic pre-preprocessing(2) alignment, and (3) entailment classification." [Ma07]. In the first phase, they obtain the parse tree, and build the monotonicity markings. These monotonicity markings are applied to each token within the parse tree. Monotonicity is determined by a set of rules according to the particular structure of the parse tree for a given token. The alignment phase involves a series of atomic edits: insertions, substitutions, deletions, and advance. These atomic edits attempt to transform the premise into a given hypothesis. From the FraCaS Test Suite, this is provided in every problem. The previous step, "decomposes the global entailment problem into a sequence of atomic entailment problems, one for each atomic edit" [Ma07]. A model is trained on each atomic edit to classify it into one of the five entailment relations depicted in Figure 3-1. A combination of these relationships through the tree produces the final result.

relation	symbol	in terms of $\sqsubseteq$	FraCaS	RTE
equivalent	$p = h$	$p \sqsubseteq h, h \sqsubseteq p$	yes	yes
forward	$p \sqsubset h$	$p \sqsubseteq h, h \not\sqsubseteq p$	yes	yes
reverse	$p \sqsupset h$	$h \sqsubseteq p, p \not\sqsubseteq h$	unk	no
independent	$p \# h$	$p \not\sqsubseteq h, h \not\sqsubseteq p$	unk	no
exclusive	$p \mid h$	$p \sqsubseteq \neg h$	no	no

Table 1: The five elementary entailment relations. The last two columns indicate correspondences to FraCaS and RTE answers; see sections 4 and 5.

Figure 3-1 "Natural logic for textual inference."

Source: MacCartney, Bill, and Christopher D. Manning. Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. Association for Computational Linguistics, 2007.

Towards a more real life scenario: It is almost never the case, that a hypothesis is presented along with a premise in a normal document to be able apply their alignment step.

MacCartney (2007) modified the relations output for the RTE test suite. Modification was made because the RTE is a challenge set to determine if one premise entails the premise. This is interesting, as opposed to converting all unknowns to no. I feel it is important to know, when you don't know the answer. After all, if you converted everything to yes, or always output yes, you have the most common class algorithm and still have reasonable odds at producing a

correct answer regardless of what is actually occurs behind the scenes (specifically referring to odds of answering a FraCaS Test Suite problem).

Natural Logic systems can be integrated/bootstrapped with other types of systems to increase overall functionality. MacCartney (2014) showed that hybridizing his NatLog system with the Stanford RTE system as described in [de06] produced better results than either system could produce independently.

NatLog can flexibly handle broad ranges of inferences and new scenarios while producing favorable results. However, NatLog has distinct disadvantages as well.

Braine (1990) claimed that universal natural logic for which all language participants could agree is implausible. Many types of inference are not addressed by Natural Logic such as temporal reasoning and; causal reasoning; in addition, Natural Logic systems will also struggle with inferences that require model building or proof based searches [Ma07]. MacCartney (2014) demonstrated that Natural Logic cannot accommodate paraphrasing, verb alterations, relationship extraction, and common sense reasoning. Natural Logic also cannot handle any type of inference that involves the use of De Morgan's law such as seen in the example "Not all birds fly = Some birds don't fly." [Ma14].

Given these limitations, it would not be practical to base a system that understands off of a natural logic based system.

Ontology Based

There is a growing body of researchers that contend that, "In the field of natural language processing (NLP) there is now a consensus that all NLP systems that seek to represent and manipulate meanings of text need an ontology (e.g., Bateman, 1993; Nirenburg, Raskin, and Onyshkevych, 1995)" [Ma96][Ma95].

A system developed by Vargas-Vera, Motta, and Domingue, called AQUA, is a ontology based question answering system (2003). They handcrafted the ontology to begin within that

contains all information that is known. Similar to the methodology presented in this paper, they permitted AQUA to handle only natural language questions.

AQUA, utilized the following steps shown in Figure 3-2.

1. To provide the question Q.
2. To parse the question in its grammatical components.
3. To translate the English question into a logic formulae.
4. To execute the logic formulae against the knowledge
base. if succeed then provide an answer and go to step 5.
if not then
To classify question in one of the following types:
what - specification of objects, activity definition
who - person specification
when - date
which - specification of objects, attributes
why - justification of reasons
where - geographical location
To transform the query Q into a new query Q'.
To launch a search engine with the new question Q'
To analyze retrieved documents which satisfy the query
Q'.
To perform answer extraction
To perform answer selection
5. Stop

Figure 3-2 AQUA's Methodology

AQUA accounts for many question types that the current system does not handle because it is not required for the FraCaS Test Suite.

The only problem with AQUA at one level is that the domain knowledge in terms of the ontology is provided before the system is ever run. This means that the ontology must be

specially crafted for a particular domain and cannot easily handle new information coming in real time since it cannot convert a given premise into new ontology facts.

Another system, CELT, developed by Pease and Murray (2003), accepts multiple premises, converts it to a logic form and places the logic directly into an ontology. Pease and Murray (2003) utilized a controlled grammar: grammar structures that easily convert to first order logic using discourse representation theory [Pe03]. [Pe03] does not provide an experimental results, but, they do state that their grammar is unable to work on any potential test suites in its current state at the time. They [Pe03] claim that they [Pe03] may be able to extend their system to handle new grammar structures.

CELT is generally similar to the methodology presented in this paper in that they both accept multiple premises, convert to a logical form, and use an ontology to understand the information.

The Mikrokosmos Ontology developed by [Ma95] follows a process that is generally taken for NLP ontology based systems as shown in Figure 3-3. The steps are the following: accept in the text, parse the text, apply your logic to the parse tree and build the ontology.

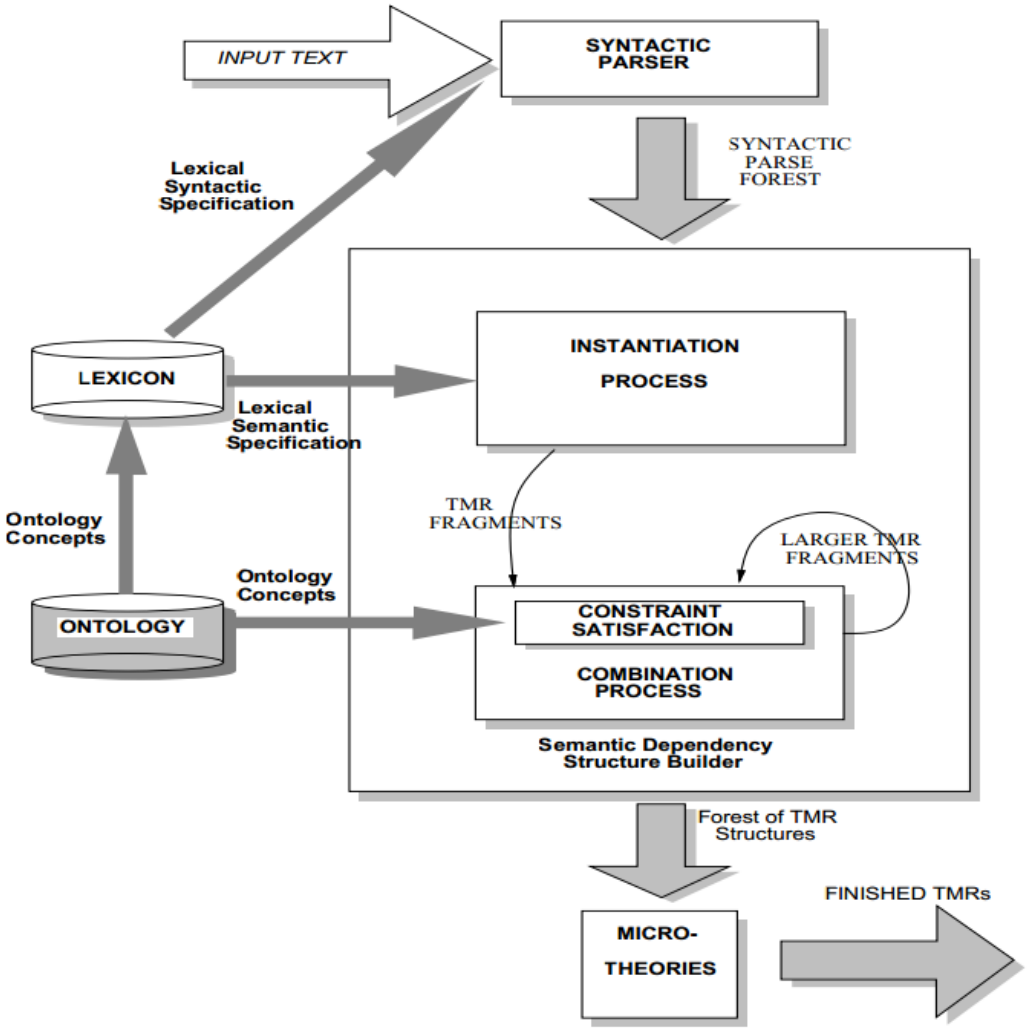


Figure 3-3 Mikrokosmos Ontology Process

Mikrokosmos Ontology was developed to look at documents involving company mergers. The primary objective of this ontology was to discern relationships between two companies and; identify the subjects, events, and relations between entities so it could be searched. [Ma95] identified situations and attached tags associated with it, such as UNIT-OF-TIME, MENTAL-EVENT, PHYSICAL-EVENT, SOCIAL-EVENT, etc. to help a user search.

Ontology theorem provers have similar problems to first order logic theorem provers, in the sense that the theorem provers can take large amounts of time to prove particular facts. Comparing it to a 3 SAT problem or any NP problem, it can take a very long time to find a set of

arguments that may satisfy a Boolean expression however checking to see if a candidate solution is a Boolean expression

Stop Words

Every piece of information from a statement is critical. When search engines perform searches, they often remove stop words such as *'the', 'a', 'an', 'it', etc in order to* help categorize terms and retrieve more relevant documents. Removal of stop words or common words across many different documents often increases recall of their search engine. However, removal of stop words causes information loss that would have made the information more contextually specific.

In the methodology presented in this paper, the stop words are not removed so that we can use them to help identify or perhaps infer if the statement(s), or question is relating to the same noun or not.

The problem

Even though researchers currently do not understand how language is understood by the human mind, nor able to replicate it within software, a system can be tested to discern actual language understandings by applying the system to a particular test suite in order to elicit system answers. Those answers are then compared to the known answer to the question. Depending on how the system performs on that test suite, you can claim that it understands the language contained within that test suite, and draw conclusions as it extends beyond the test suite.

Natural language understanding is no simple problem.

Chapter 4 - Methodology

The methodology presented in this paper is focused on extracting information out of a set of sentences and being able to ask natural language questions about information extracted from said premises. We extract and track everything that is possible[1], given our methodology, and try to build new pieces of information with that. We can see in Figure 4-1 a high level view of this algorithm.

To see examples of this system running, with this methodology there are several large examples in Appendices C-I.

Algorithm

Perhaps the best way to understand how the system works is by taking a look at the high level algorithm as to the steps the system must take to achieve an understanding.

```
1.  For each premise: Parse the premise and generate ordered list of
   grammar trees
      a. For each grammar tree for a given premise[2]
            i.  Generate intermediate object by pattern matching each set
                of children for all non-leaf nodes[3]
                      //These intermediate objects will hold additional
                      generated information
           ii.  Normalize words; nouns become singular, verbs become
                present tense, etc.[4]
          iii.  While there are changes to be made
                   1. Apply pos/word rules (listed on APPENDIX #) to
                      intermediate object.
                   2. Push information into temporary ontology
                   3. Type match as needed (notably for verbs)
                   4. Build relationships
                   5. Push relationships into temporary ontology
           iv.  Merge temporary ontology into main ontology
```

[1] It is not possible to extract everything with insufficient rules

[2] A grammar tree is valid when all sub steps are completed successfully

[3] If there is a set of children where there is no match in the grammar tree restart loop starting on next grammar tree

[4] This information is maintained for nouns to keep track of the quantity, the information is needed for verbs to maintain a partial ordering on the information as it is presented

```
                    1. Find matches use the pre-existing one as opposed to
                       the one in the temporary ontology //Careful on being
                       wrong
             v.  Generate new information based on structure of main
                 ontology
            vi.  Clear temporary ontology
2. For the question follow 1.a.i-vi
3. Find an answer to the question yes/no/unknown by matching the temporary
   ontology to the main ontology
```

Figure 4-1 Algorithm

We must look at each sentence individually and in order to be able to understand the information being presented. Given a particular sentence, we send the sentence to OpenNLP and StanfordNLP to obtain parse trees for the given premises. Both OpenNLP and StanfordNLP return a set of parse trees, which can contain part of speech information along with how the sentence is structured into its various components. Neither of these systems are perfect, which is also the reason why two of them are being used. StanfordNLP is supplementing the results of OpenNLP. When OpenNLP and StanfordNLP return their results, the confidence of each parse tree is also returned. This system orders the results from both systems, by first ordering from greatest to least confidence from OpenNLP first, followed by StanfordNLP with the same ordering.

The ordering from OpenNLP and StanfordNLP was chosen somewhat arbitrarily, but based off of anecdotal evidence from testing, that OpenNLP tends to return the most correct results, while StanfordNLP does tend to be correct a lot of the time, it does have its own troubles as well. Sometimes even the best results from StanfordNLP do not contain parts of the sentence, which is integral to the entire process presented in this paper. The ideal scenario is one where the various NLP tools produce 100% correct parse trees, though this is not currently practical and it must be addressed within a system such as this.

Pattern Matching Grammar Tree

We are validating that the grammar tree is a valid grammar tree via pattern matching. Pattern matching requires a part of speech, context, and the part of speech of the children. Pattern matching works in a depth first style, working from bottom left navigating right across the grammar tree. A sample parse tree, seen in Figure 4-2, of the initial steps taken to start to

build subject (noun phrase) of the statement. For a given pattern match we instantiate an intermediate object which holds the same structure along with some additional information regarding the phrase, this is discussed in the following section. To see what the various labels on the nodes contained within Figure 4-2, or throughout this paper, the label along with information what it represents is contained within Appendix B.

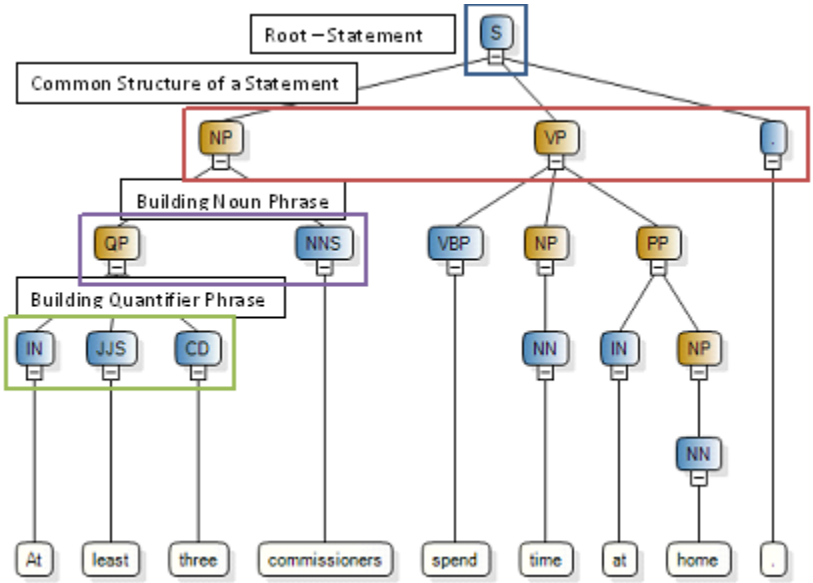


Figure 4-2 Sample Parse Tree for a premise

While there can be many different structures that make up given phrase type, or many different ways a statement can be organized yielding different structures, each of these structures generates an object that is the type of the node in question. For example, there are more than hundreds of different ways to build a noun phrase, must account for each type of these different ways to represent a noun phrase. There is a rule that takes into account when you have just NNS node vs. the one we see above in Figure 4-2 that contains both a quantifier phrase and a NNS.

Generally, we ignore the leaves (the words, except in a few cases mentioned later) initially. For a given pattern, we need to know the part of the speech of the current node, this reduces the possible matches, we also need to know if this is the root node or not, and given that information we look at the set of children and their respective part of speeches if there is a match

found within our list. If there is no match found, we clear out all information in the temporary ontology and restart with the next possible grammar tree if one exists.

Where we wouldn't ignore the leaf, specifically the word is when working with the part of speech – articles. We do this mostly as a step that can done right away as no further processing on the word need be done. It allows us to determine part or the entire qualifier for a noun phrase.

The reason we require the knowledge if we are at the root node is that we require the input to have valid punctuation. Additionally, this is a part that OpenNLP and StanfordNLP routinely have problems with, with this particular test suite.

When every subtree within the grammar tree has been pattern matched, then the grammar tree is a valid grammar tree though not necessarily sound.

Invalid Grammar Trees

If a grammar tree cannot be validated by pattern matching then we can conclude it is an invalid grammar tree and move to the next grammar tree if one exists. If a valid grammar tree does not exist then we are unable to answer the question. While in some cases, it may be possible to answer the question as enough information would be present, but what if the premise that was invalid contained information that would negate that information or the answer to the question? This is why if a premise or question is completely invalid, then we must throw out the problem entirely.

An example of a structurally invalid parse can be seen in Figure 4-3. The '?' is a part of a noun phrase. When we see this, or missing tags, or words missing, it is easy to say / claim that the parse tree is invalid.

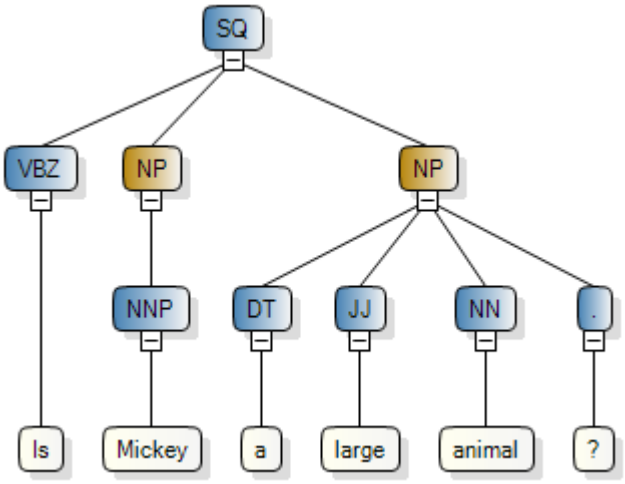


Figure 4-3 Invalid Grammar Tree

In Figure 4-4, we can see the correct usage of the punctuation mark as it is applied properly to the root node. Again, as a reminder, the parent nodes labels are defined in Figure 4-4.

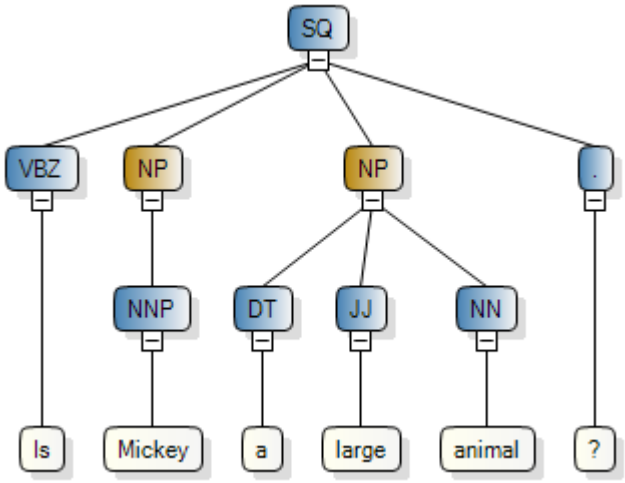


Figure 4-4 Valid Grammar Tree

Normalization

Given this algorithm requires a rule that can apply to every situation; we must reduce the number of the possible situations. This is done by stemming words i.e. converting all nouns to their singular form, and all verbs to their present tense form. We do this to have fewer lookups, particularly when matching. First, you would look up the singular version of the noun in the ontology for matching, does it even exist; if so then you can look at the quantity information

explained later to further match. For the verbs, the same thing applies. As opposed to copying a rule several times over for each tense of the root word, you have just one, and handle the time component separately.

Where this differs from stemming is that it also converts adverbs to a particular form e.g. slower to not faster.

Intermediate Step

It is useful to have an intermediate step to convert the grammar tree into the final form of the ontology. This intermediate step exists simply to reduce the amount of code and complexity of the software itself as it transforms. A simple example would be, to encode all the rules that may be applied to the word 'the'. To attempt to encode all of the rules we think for a second of all the possibilities that could come after it and think of all the situations those two concepts could be found within such as part of the subject of the sentence versus the object of the sentence. The general rules would be the same in both instances so it only makes sense to reuse the code.

The purpose of having intermediate objects is to facilitate the transformation of natural language to an ontology. The intermediate objects also compartmentalize the information that is relevant to a particular sub-tree within the parse tree. Take a noun phrase for example, the only thing that matters that comes from it is a reference to the noun or perhaps even a list of nouns. (Take note that we are working with the English language and there are exceptions almost everywhere).

Every word has a rule that it applies to the intermediate object it is contained within. This rule may be shared by many other words (such as in the case of a generic noun). An example of a broad rule, to apply a singular proper noun, 'Italian', to a noun object can be seen between figures Figure 4-5 and Figure 4-6. It does not matter what singular proper noun is in that location, it could be 'American', or perhaps 'European', it is flexible to accept any singular proper noun.

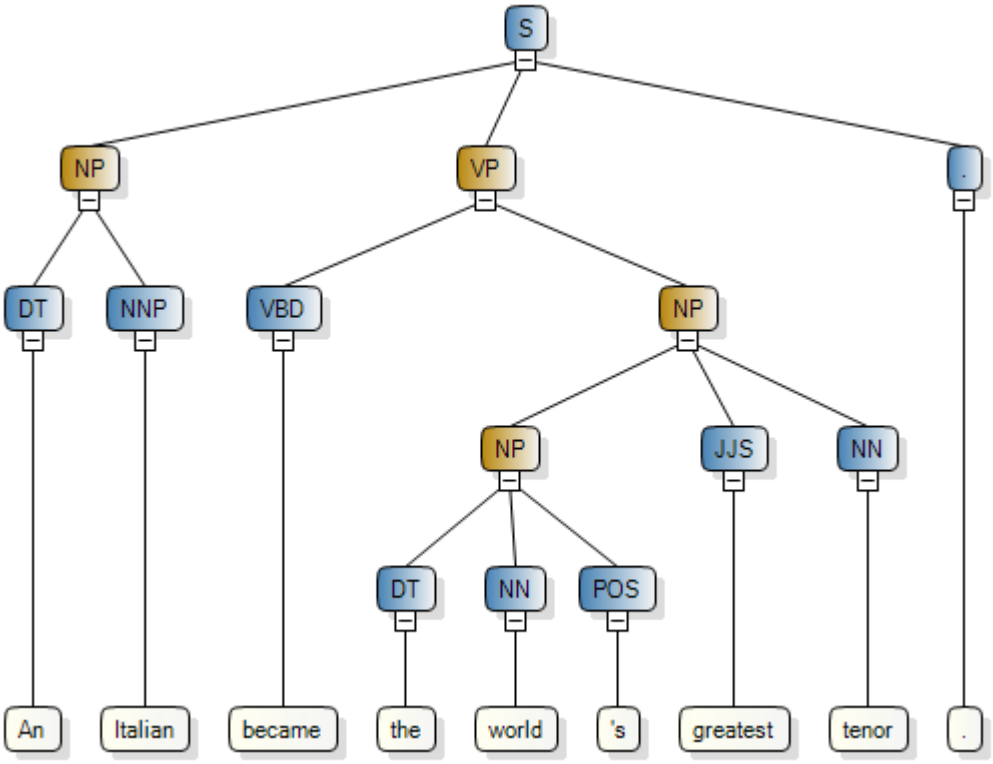


Figure 4-5 Example Parse Tree part 1

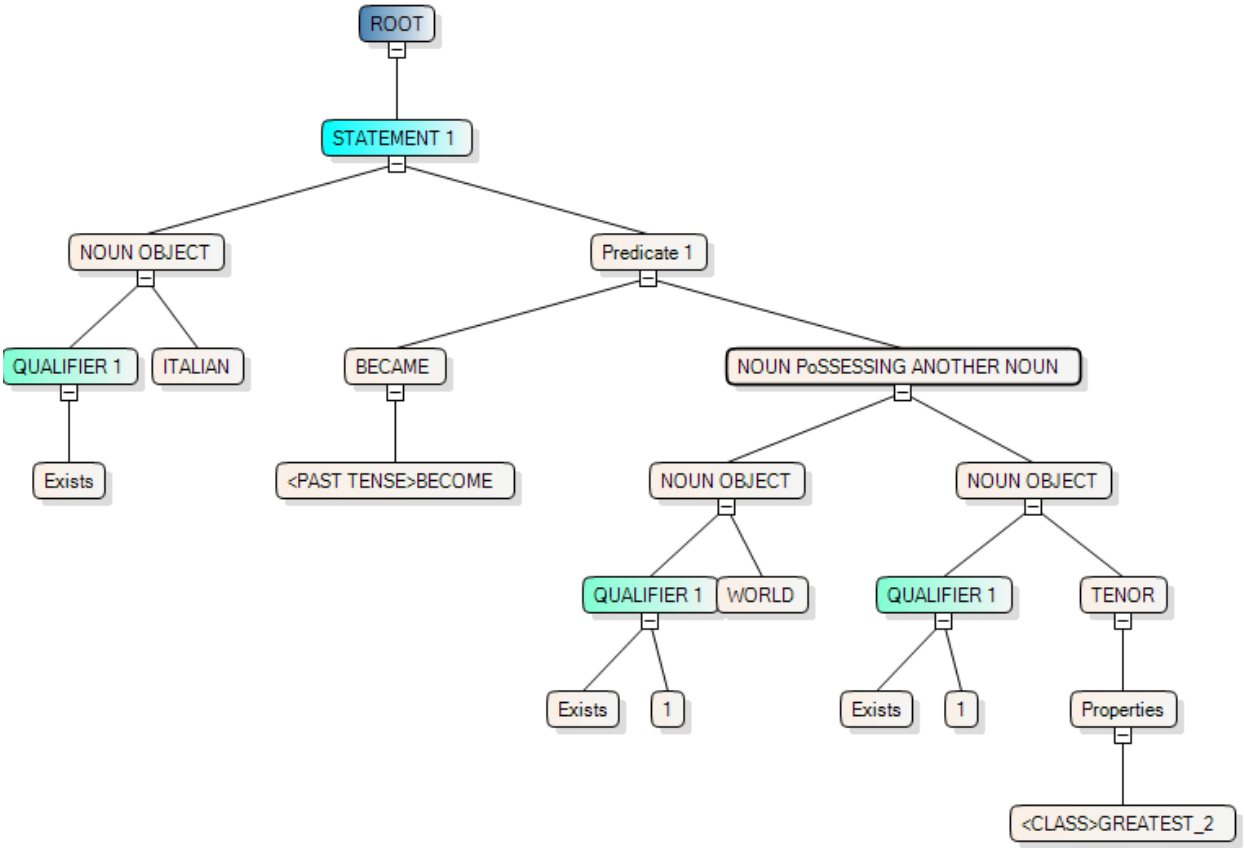


Figure 4-6 Corresponding Intermediate Object part 2

Some rules are very broad as the quick example above shows, others are very specific and narrow. Such as the use of the word 'the' immediately prior 'world' seen best in Figure 4-5, where it causes the system to look up to see if 'world' (quantity 1) exists within the ontology already and if so bind the reference to it after the intermediate has formed.

We are effectively using the meaning of the word as a rule and how it modifies or affects a particular intermediate object, or ontology.

Take the phrase 'computer', initially we have no idea if we are talking about a specific computer or a generic computer. We could change that and change the phrase to 'the computer' to refer to either a computer that was either previously mentioned (which the system would look up for a prior reference to this computer) or to create a new specific computer that is being

discussed. We could change the phrase to ‘a computer’ to make it more generic and there would be no look up based upon this phrase. Additionally, we can change the quantity of computers being discussed first just by changing the word computer to its plural form ‘computers’. When we are talking about that we know that there are at least 2 computers and so we represent the quantity by an inequality qty: >= 2. We could change the quantity to represent an exact number of computers by changing the phrase to ‘3 computers’ which would yield of a quantity of exactly 3. We can even add properties to further describe the noun object. We can change the phrase to ‘the fast computer’ and now we are referring to a specific computer that has been referenced before that is fast OR we are creating a new instance of a specific computer and adding the property to it that it is ‘fast’. We can add several properties to the noun object perhaps by changing the exemplar phrase to ‘the fast black computer’. Now the phrase is referencing a pre-existing computer that is both fast and black or creating a new instance of a specific computer and giving it properties of both fast and black.

When the system is looking for a pre-existing reference it must match on all available information otherwise there is no match and a new instance is created.

OpenNLP and StanfordNLP return at least some grammar tree for a given statement. It may not be correct or even valid but there is always something. In the event that nothing would be returned the question is clearly not answerable so steps 2-3 would be void. If it returns several part of speech trees they are ordered from highest confidence to lowest confidence. For each premise, we have a set of grammar trees, looking at them in order from top of the list downward; we must make sure that the parse tree is a valid grammar tree. We do this by pattern matching by taking a parent node and looking at its children node(s), if it does not match to one of our pre-existing patterns then this grammar tree is invalid. This can occur by a having unexpected children for a parent or just invalid set of children such as a [X] node which indicates from OpenNLP that the structure is unknown/error. We do this for each parent and child node(s) sub trees in a depth first order of processing.

In step 1.a.ii, there is currently no way for this to fail for this particular test suite, this would fail if there was not a correct lookup for the singular form of a noun or a present tense

form of a verb. The reason it is not applicable to this test suite is all of the words that are included into this test suite are known and their respective normalized forms are included. If there is an appropriate lookup it will have completed successfully.

The only place at which 1.a.iii would come fail would come from step 1.a.iii.1 for a rule that doesn't apply such as a setting a quantity to a singular object, such as the short phrase "3 dog" makes no sense. Dog is singular, yet the three implies that there should be well 3 dogs. This all assumes it gets past the step 1.a.i which matches patterns.

System Architecture

It will be helpful to understand the underlying architecture of the system. There is the main ontology (existing knowledge base), the temporary ontology (new premise, or question that is being processed), and several ordered lists that keep track of most recent subjects, objects, and verbs.

Underlying Representation

After the grammar tree has been declared successful from the pattern matching in the step prior, we are given effectively a tree view of the same sentence. Except each parent node has transformed into an intermediate object as can be seen in Figure 4-7. Initially, there is no difference, but as we work through this step, it begins to hold more information about the statement or question stored within both the intermediate object along with the temporary ontology. This step works to extract as much information as possible out of the statement or question.

There is the ontology which contains all of the nouns, relationships, and attributes. Additionally, there is an ordered list of nouns, ordered by last seen, a list of verbs which operates the same way. These lists are effectively pointers to the ontology in terms of what they represent. These ordered lists are needed to help solve problems related to inference.

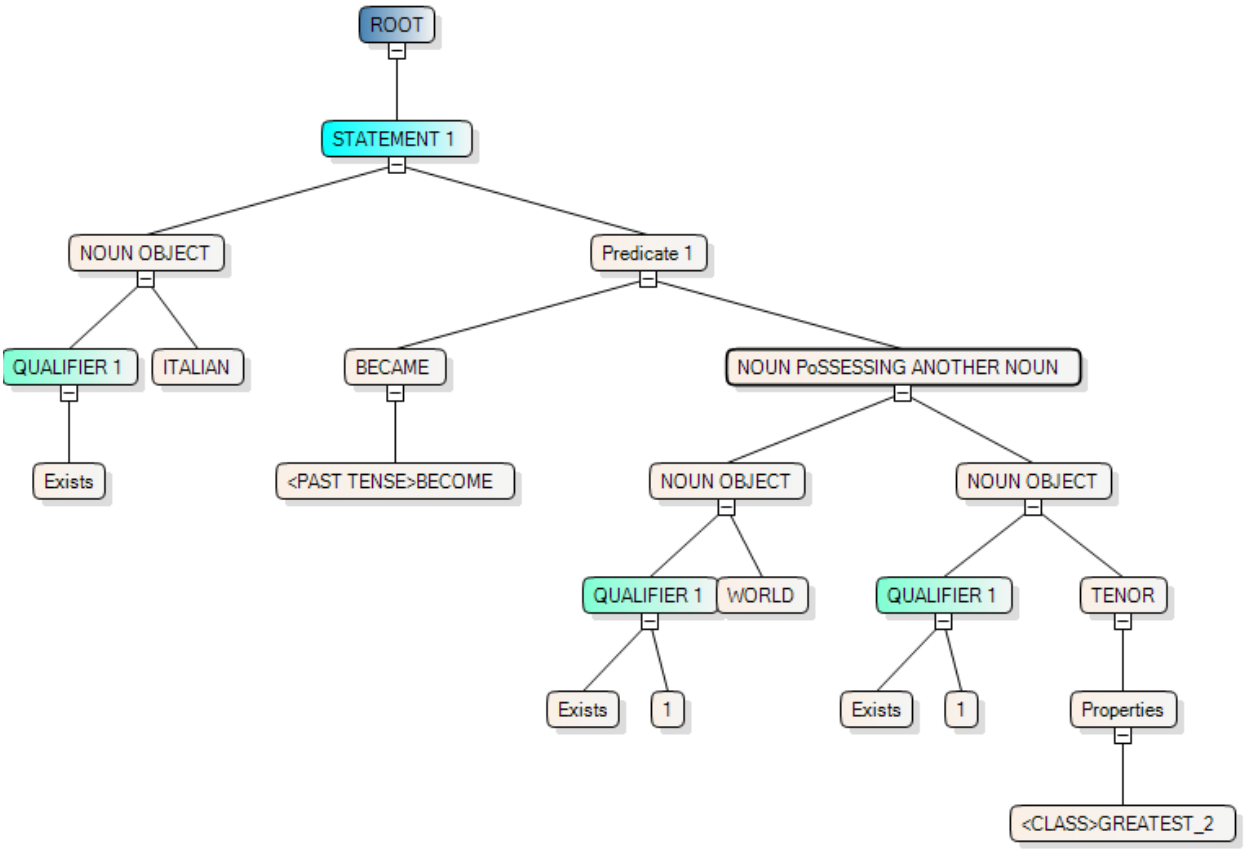


Figure 4-7 Intermediate Representation Example

In Figure 4-7, we see an example of the intermediate representation visually depicted. The various parents contain information about their children and what is of importance just from examining the grammar. Each premise or question will have its own particular intermediate representation and how it looks (if it is valid) is dependent upon the grammar tree that is returned from OpenNLP or StanfordNLP. There are many different types of each object, with the idea behind each object that it can understand how a few children components work together and how it can interact with different components in the tree.

Minimal information is attached to the intermediate representation, effectively only information that is attached/added is information that immediately dictated in part by the part of speech, perhaps a plural noun versus singular noun, or a past tense verb versus a present or future tense verb.

Following this will be the list of each part of speech/intermediate object type to how we transform from language to final representation.

Qualifiers

Table 4-1. Qualifiers Found in FraCaS Test Suite and their internal representation

Word	Qualifier Representation
ALL	All
EVERY	All
EACH	Each
BOTH	Each (2)
EITHER	Each (2)*
THE	Exists
AN	Exists
THIS	Exists
ANY	Exists
NEITHER	Neither
A	New
ANOTHER	New
NO	None
SOME	Some

With the qualifier representation found in Table 4-1, it can be seen that reduces the number of options we have to work with. Just because these are the list of articles does not mean these are the only types of qualifiers. There are other types of qualifiers such as when you have a specific quantity of an item. Such as when you have 3 computers, the number 3 would be represented as a qualifier. Also, there are times when you have a range (ex. 2 to 3), inequalities (ex. At least 3 computers), functional based such as counting the number of entities or more than

half. When trying to match qualifiers there are several things that can be returned {equal, superset, subset, partial intersection, no intersection, less than, greater than}.

The qualifier representation, as depicted in the above table, has associated rules for it. When working in the context of a noun phrase. When you have a noun phrase and the qualifier is 'exists' it is going to check if that particular noun phrase (match) has occurred previously, if so it is going to bind that reference. The order in which it looks in the ontology is looking backwards through an ordered list of nouns in order of which noun was presented first. This is to say things that are more recently talked about will be compared first for a match. If not, it will instantiate a new instance of the noun phrase. When there is a 'new', there is no attempt at matching with what exists in the ontology it just instantiates a new instance of the noun and when the phase to generate new information off the structure of the ontology occurs it is then attempted to be bound to a particular class type if one already exists it simply becomes another instance of that particular class.

The rule 'All', works with a given class of objects if they exist it pulls that in and applies additional properties or rules to it, otherwise if it does not exist in the ontology at this point, creates a new instance of the noun, and assigns a quantity ALL representing all possible past, present, and future instances of this object.

'Each' with or without a quantity associated with it, is always talking about a set. The properties, or rules are applied to each items within the set being described. Which where it specifically applies the additional information such as properties to, is dependent on the actual noun itself, It is always better to apply properties, rules, or constraints at a higher position in the ontology structure for a particular noun as it tends to apply to more situations. However, typically when dealing with the 'each' rule, a set is predetermined, or is a subset of some existing set of all items of a type of noun.

Matching Qualifiers

For two qualifiers to be equal the values must match exactly. For superset, the qualifier that is being compared to must contain all of the values of the other qualifier and contain more than those number of values. Conversely the subset must be a strict subset of the other qualifiers values. An example would be when you have two ranges that are being compared and the first range contains the numbers 2 to 5 while the second range contains the numbers 2 to 4. We can say that the first qualifier is a superset of the second qualifier. If we swap them then we can say that it would be a subset. If we change the values of the first qualifier to be 1 to 3 and leave the second qualifier range alone then we have a partial intersection.

Using the results of testing if one quantity intersects with another quantity tells us if we have a match. For example, in the ontology, if conceptually you have the premise '5 women are lawyers.', and the question, 'Are at least 3 women lawyers?' we are capable of determining this if the we can find a match where the inequality holds true.

Noun Phrase Objects

These intermediary objects purpose is to construct an instance of an object(s) assign all quantity information and property information about it or to simply perform inference if appropriate. Regarding error checking, if we receive a qualifier that indicates there is a singleton of an object yet the object is plural we know that there is a mismatch and we can report this error back.

If we must use inference, we have an ordered list of nouns as they are presented also notated by whether they are the subject/object of a statement. Additional information also must be checked, such as if we are looking for an 'it', we cannot accept the most recent noun, as it could be a person, so we must type check the noun. The order which we look is starting from the most recent and working backwards.

List of Noun Intermediate Objects

#	Intermediate Object / Form	Example Usage
1	Noun Adjective	Is Mickey small?
2	Adverb Noun	
3	Noun List	Jon, Mike, and Ron, are lawyers.
4	Verb Noun	The barking dog…
5	Noun Question	
6	Noun Statement	
7	Noun Object	A Scandinavian won a Nobel prize

Figure 4-8 Types of Nouns Encountered

In Figure 4-8, we see different types of nouns, 1-6 in the figure rely on #7. The noun object is the base level, it holds the quantity information from a possible qualifier, as well as the noun name itself and the type of noun that it is.

These different intermediate forms for nouns provide the basic instruction levels as to how to connect or piece together information along with storing the relevant information before it gets pieced together and placed into the temporary ontology.

Matching Noun Objects

First, to match a noun object(s), the name must match. Next, if the name matches, then we take a look at attributes, if we are looking at the new one and it has a subset of attributes, it still could be a match but we must consider the qualifier.

In short, we are testing to see if one noun matches exactly on all aspects, names, qualifier, and attributes, or whether it is a subset/subclass, or a superset/super class.

Verb Phrase Objects

The general approach for verbs is to provide a relation between one to three entities. Normally there is a subject and object of a premise and the verbs job is to be the relation between the two. The verb object will have ownership of a noun object (the object of the input).

All verbs exist in some part of time, whether past, present, or future, even more than that if we can gather enough information to for example say it is in the past up to and including the present. Each time the system see's a verb regardless of tense a global counter v_t (to this conversation/document) is incremented and attached to the verb relationship. If the verb is a past tense verb we say that the relationship occurred before time v_t. Conversely, if the verb is future tense then we say the relationship occurs after time v_t. Maintaining this tense information is relevant as we have normalized the verb and lost its original word. Additionally, this lets us rule in/out relationships based on the time they may have happened. For example, there is no reason to consider events that have yet to occur if you are looking for a relationship that should have existed prior.

There are exceptions where the verb can be referring to another statement entirely notably a relationship that exists from a prior statement. When the verb phrase object must infer what it is relating to, this is when the ordered verb list comes in handy and references the last referenced relationship.

Matching

To match a verb, you must match the relationship (verb in normalized form) and the time the verb existed in (v_t). Next, the terms must also match (need not be exact match). The matching term could be either a more generic form of the term or a more specific instance of it.

Adverb Phrase Objects

Adverbs can be expressed in a variety of phrases but in short what they do is simple. The general approach is to obtain the verb that they are modifying and attach the adverb as an attribute. If the adverb is modifying a clause or sentence attach it as an attribute to the whole.

Adverbs also go through a normalization process, for example slow becomes not fast. Since adverbs have no formal meaning any meaning attached so long as it is relative seems to work out.

Matching

In the simple case of matching adverbs as attributes attached to verbs: If the question has the adverb modifying a verb then for there to be a match, the matching verb must also have this same adverb attribute. It is not the same in reverse. For example, if we know John walks slowly. Slowly modifies walks. If we ask 'Does John walk?' we can still confirm that he does regardless of his speed (or adverb).

Prepositional Phrase Objects

The approach with prepositional phrases is to attach a relationship between some noun and the object of the prep phrase. It acts similarly to a verb phrase in terms of the relationship.

Matching

The prep phrase in a question would only pose as a constraint that must be satisfied. When it is required that the prep phrase must exist, matching behaves similarly to matching in verb phrases in that it could be an exact match/sub/super set/class.

Combining intermediate objects together

For premises, the root node is typically a statement of one type or another. There are two steps in bringing it all together. Step 1 is to generate obvious information and fill in anything that is obvious. Step 2 is to take any blanks or relationships that require pieces of information that can come from different parts of the premise and combine them via a relationship. Effectively a simple way to think of this is to first build any information about the nouns that you can, and then second is to figure out a way to connect them.

Knowledge Representation

Knowledge is represented in the form of an ontology with it taking several steps to convert from natural language to the internal representation to final form of the ontology.

Merging Ontologies

We take every element in the temporary ontology and we compare it for equality in the main ontology. If there is an exact match, then we replace any remaining instances in the

temporary ontology with the element from the main ontology. If no match is found, then we simply just add the element to the ontology. We use the procedures found in the prior sections to determine if there is an exact match.

Question Answering

When a question comes up the previous steps are taken as indicated above except upon entering step five the temporary ontology exists. Answers to questions can be derived by transforming language to the ontology and matching with the prior knowledge base. The problem then becomes to find a satisfiable mapping from the temporary ontology to the main ontology. Every object in the temporary ontology tries to find the potential matches it has in the main ontology. Effectively looking at each tuple and evaluating that to be either true or false or unknown depending on the information in the main ontology. The way the question is evaluated is, for every individual and class in the temporary ontology all connections are formed to the main ontology. Using these connections, an attempt to replace the temporary ontology individual or class with each specific related term, whether up or down or side to side in the hierarchy is made. At least as far as these problems are concerned, there is only one solution that can be found if it is either true or false. Unknown is the case where no such replacement was found to satisfy a particular predicate. Currently, for words that are under development, if they are detected the system can throw an error indicating that the system cannot attempt to solve this problem. It evaluates every relation under this assumption. If a result of either true or false is produced then that is the answer to the question and it returns. However, if it returns unknown then it continues to change another term and repeats this process until no more configurations are possible in which case the answer is truly unknown.

Example of System Running

Figure 4-9 shows one of the problems evaluated using the system, based on the above mentioned methodology. The question from figure 1, the word who generates the generic predicate WHO(A, [person]) where A is a particular noun phrase from this example, Italian to be exact. This shows that there is not sufficient information provided in the premise from Figure 4-9. Therefore we must add in another statement as shown in Figure 4-10. Without the additional information, this system would produce an unknown as an answer to the question.

An Italian became the world's greatest tenor.

Was there an Italian who became the world's greatest tenor?

Figure 4-9 Problem 1 from the FraCas Test Suite

An Italian is a person.

An Italian became the world's greatest tenor.

Was there an Italian who became the world's greatest tenor?

Figure 4-10 Updated Problem 1 from the FraCas Test Suite to provide sufficient knowledge to solve question

is_a(<Instance: QTY 1>ITALIAN_1, <Instance: QTY 1>PERSON_2)

Figure 4-11 Main Ontology after premise 1 is processed

instance_of(<Individual: QTY 1>ITALIAN_1, <Class: QTY 1>ITALIAN_1)

instance_of(<Individual: QTY 1>PERSON_2, <Class: QTY 1>PERSON_2)

subset_of(<Class: QTY 1>ITALIAN_1, <Class: QTY ALL>ITALIAN_3)

subset_of(<Class: QTY 1>PERSON_2, <Class: QTY ALL>PERSON_4)

child_of(<Class: QTY 1>ITALIAN_1, <Class: QTY 1>PERSON_2)

child_of(<Class: QTY ALL>ITALIAN_3, <Class: QTY ALL>PERSON_4)

Figure 4-12 New facts based on Main Ontology

Starting with the first premise in Figure 4-10, the system generates the main ontology shown in Figure 4-11. Figure 4-12 shows new facts that are generated from the main ontology. Figure 4-11 and Figure 4-12 show the extracted basic nouns and the generated additional information. The next premise generates the following facts shown in Figure 4-13.

Become<past tense, (t+1)>(<Individual: QTY 1>ITALIAN_3,
<Individual: QTY 1>World_4)
Possesses(<Individual: QTY 1>WORLD_4, <Individual: QTY
1>TENOR_5)
Attribute_of(GREATEST, <INDIVIDUAL: QTY 1>TENOR_5)

Figure 4-13 New facts added to the main ontology for premise 2

instance_of(<Individual: QTY 1>ITALIAN_3, <Class: QTY
1>ITALIAN_1)
instance_of(<Individual: QTY 1>WORLD_4, <Class: QTY
1>WORLD_5)
instance_of(<Individual: QTY 1>TENOR_5, <Class: QTY 1>TENOR_6)
instance_of(<Class: QTY 1>WORLD_4, <Class: QTY ALL>WORLD_7)
instance_of(<Class: QTY 1>TENOR_5, <Class: QTY ALL>TENOR_8)

Figure 4-14 New facts generated based on the main ontology for premise 2

Become<past tense, (t+2)>(<Individual: QTY 1>ITALIAN_6,
<Individual: QTY 1>World_7)
Possesses(<Individual: QTY 1>WORLD_7, <Individual: QTY
1>TENOR_8)
Attribute_of(GREATEST, <INDIVIDUAL: QTY 1>TENOR_8)
WHO(<Individual: QTY 1>ITALIAN_6, <Class: QTY 1>PERSON_1)

Figure 4-15 New facts in Temporary Ontology for Question

We are able to extract nineteen facts from just the first premise alone and twenty five facts from the second premise for a total of forty two facts from two rather small premises. Notice on Figure 4-13 that <Class: QTY 1>ITALIAN_1 is reused as there was a direct match from the existing ontology as depicted from the end result of Figure 4-12. Figure 4-15 is essentially a repeat of Figure 4-13 with the addition of the 'who' predicate. Each individual from Figure 4-15 finds relations to other individuals and classes in the main ontology. Each individual finds a parent such as <Class: QTY 1>ITALIAN_1 for <Individual: QTY

1>ITALIAN_6. From this relation, <Individual: QTY 1>ITALIAN_3 is found as the only possible replacement for it. How is this determined? Each individual listed in Figure 4-15 finds what parents (and other relations) it has in the main ontology. From there it finds what siblings it has which happen to map easily to Figure 4-13's configuration. The 'who' predicate is evaluated for truth by checking to see if there exists an <Individual: QTY 1>ITALIAN_6 is an instance of <Class: QTY 1>PERSON_1. We find that there is a replacement for it, which is <Individual: QTY 1>ITALIAN_3 which is an instance of <Class: QTY 1>PERSON_2. Become<past tense, (t+2)>(<Individual: QTY 1>ITALIAN_6, <Individual: QTY 1>World_7) has to be time matched as well. Not only must there exist a relation Become that has both individuals but, that relation has to hold true before (t+2), which is found as there is only one Become relation in the main ontology which was known to be true in at some time before (t+1) which satisfies this condition. Since each of these were successfully able to be evaluated to true the result to the question is therefore yes.

Chapter 5 - Overview of Results

The FraCaS Test Suite contains 346 NLI problems, divided into nine sections, each focused on a specific category of semantic phenomena [Co96][Ma14] and also in [Ma08]. For comparison to previous work, we will not remove multiple-premise problems, or problems that are missing a hypothesis as done in [Ma07][Ma14] and also appearing in [Ma08]. No modification to the test set has been made to accommodate my research. However, we do remove problems from the test suite that contain a bad parse on any one of the premises for the problem or the question. We will show a comparison based on percentage of problems that are answered correctly. This research focuses on six sections which represent Generalized Quantifiers, Plurals, Adjectives, Comparatives, Verbs and Attitudes respectively.

Figure 5-1 shows that the system performs exceptionally well. The accuracy is calculated based on the correct answer and remaining problems. There is one critical thing to be taken from this, that while this methodology is fully capable of solving these problems, obtaining a valid part of speech tree for each premise and question in each problem is paramount.

Section	Original Problems	Bad Parses	Remaining Problems	Correct Answer	Acc %
1	80	10	70	61[5]	87.00
2	33	11	22	21[6]	95.45
5	23	7	16	16	100.0
6	31	9	22	22	100.0
8	8	4	4	4	100.0
9	13	6	7	7	100.0
Total	188	47	141	131	92.90

Figure 5-1 Results

[5] The system realized that it could not answer the 9 questions out of the 70 remaining problems for section 1 so it produces a null answer; we count null answers as wrong.

[6] It can solve problem 87 from the test suite but due to this it cannot solve problem 88 due to word play.

If you would like to see how the system performs specifically on the sections within the FraCaS Test Suite, see Appendices C-I for those examples. Each problem that is displayed in the appendix contains the step by step process, from parsing an individual sentence to building the ontology to generating additional information to finally answering the question.

In total, the results mean that where our system supports the language, the system works well. The exception is when multiple problems in the test suite are the same but can be interpreted differently.

Comparisons

[Ma07][Ma14] achieve rather good results, however they removed problems with multiple premises as well as removed problems without a hypothesis. Removing problems with multiple premises takes us further away from a comprehensive natural language understanding system. If anything, there should be work towards multiple premises. Additionally, modifying the hypothesis using a set of rules to add or remove words from the hypothesis supposes that there is going to be a hypothesis (particularly close to the form of the question itself) which is certainly not natural.

Table 5-1. NatLog's accuracy on the FraCaS test suite, [MacCartney 2007]

§	Category	Count	% Acc.
1	Quantifiers	44	84.09
2	Plurals	24	41.67
3	Anaphora	6	50.00
4	Ellipsis	25	28.00
5	Adjectives	15	60.00
6	Comparatives	16	68.75
7	Temporal	36	61.11
8	Verbs	8	62.50
9	Attitudes	9	55.56
Applicable sections: 1, 5, 6		75	76.00
All sections		183	59.56

To compare MacCartney and Manning 2007 methodology more directly towards the methodology presented in this paper, we show the results of their system with multiple premises added in as if they got them wrong. This is not saying that it is not possible to extend their system to accommodate multiple premises, just as presented they did not.

MacCartney and Manning 2007 removed 151 problems due to multiple premises. If we add that back in we get the real accuracy of their system, which is closer to 32% correct. Simply answering yes to all without looking at the question at all would have achieved better results.

Table 5-2. Performance on FraCaS problems on sections: 1, 2, 5, 6, 9 compared

	Problems	Acc%
Most common class 'yes'	178	51.68
MacCartney 07	108	75.00
Natlog [Ma14]	108	87.04
This system	137	92.27

When our methodology is compared against the semantic containment and exclusion method as seen in [Ma14] we clearly see that when statements are analyzed in depth we gain greater accuracy overall, as shown in Table 5-2. With the notable exception to the first section, generalized quantifiers, where the system does not yet support the language contained in 9 of the problems despite it producing a valid parse. In addition to a higher accuracy rate on the FraCaS Test Suite we also are capable of working with problems that contain multiple premises. Hence the higher number of problems shown for our system.

It is not possible to do a direct comparison to the 108 problems that were done with Natlog. The FraCaS Test Suite is not versioned, and hypotheses were added at some point for some of the problems. Given that [Ma07][Ma08][Ma14] ruled out problems with multiple premises and problems with no hypothesis and we don't know which problems to compare to.

Intuition Behind Performance Gains

The methodology presented in this paper tracks every piece of information extracted by this methodology. Information is extracted from every word, phrase, and sentence. Being able to reason over this information using rules allows the system presented in this paper to 'comprehend' the material and provide answers to questions.

Systems which do not account for every detail, effectively are working with unknowns, are making educated guesses. The more information a system is unaware of the more of a guess it becomes. The more of a guess it becomes the more likely you are to generate a wrong answer.

Alternate Comparisons

A direct comparison cannot be made with Google search, nor could it be made directly with Apple's Siri.

Take Google's search, when an entire problem is entered, the search engine tries to find a document that contains the keywords from the problem, in the searchable web. Google applies a series of algorithms to obtain a list of potential candidate documents using everything from autocomplete, to synonyms, to query understanding, to search methods (document type: image, text, video, etc.) [Go14]. These documents are then ranked using Google's ranking algorithm like PageRank to determine if the page is relevant. Followed by a series of other metrics to make sure the document is 'fresh' and catered towards the user specifically, using their previous search history and location information [Go14]. Since this type of search at present does not aim to produce answers to questions it is not comparable.

Another piece of software that deals with language is Apple's Siri. Siri is a voice activated assistant that can run applications or assist a user when it can [Jo13]. Siri's functionality is designed around being able to handle a user's request. Siri was not designed around understanding premises, only commands or actions that it could possibly take to assist you in finding a restaurant, help perform some calculation, or interact with the phones functionality like making a call by using a person's name [Jo13].

Chapter 6 - Conclusion and Future Work

Making machines understand natural language at any level is a challenging problem. We've developed a methodology that converts natural language into ontology while leveraging the ontology to help solve questions posed in natural language about the facts in the ontology. We've shown that our methodology which works around extending the semantics of language, by keeping track of inferences, quantities, inheritance, properties, and set related information, produces a high degree of accuracy. Using more information than is directly seen in the statements allows us to help match terms in a natural way, which allows for questions to be proved correct (yes or no) or unsolvable (unknown).

There is room for improvement currently on how answering a question is done to make it slightly better on average, if you just think if everything were prioritized to look at either the last referenced or most successfully accessed or even some combination, it stands to reason that you would not necessarily have to look at all the elements. Clearly, you would still have to look at all combinations in the worst case, but it is my belief that you should be able to find the answer faster in an average situation.

The next logical step is to see how well our methodology adapts and performs to the other sections that are not addressed in this paper. Also, there is a maximum of just five premises in the largest problem in this problem set; analyzing a full page document is a direction that needs to be pursued.

Additionally, it would be interesting to pursue a game theory style approach to answering questions and understanding information. Keep track of all possible states as the system reads through the premises and questions. Any time you have a decision or question that yields multiple answers, simply copy the ontology the number of times you have answers. Each of these ontologies then proceeds with the understanding of that decision being interpreted a particular way. Present the answers to yet another system that would select the answer based on some perceived state.

Finding a set of operators that language works over, to be able to understand and learn new words based off of the starting set of "words", or operators so that you do not need to implement all of the rules, just a sufficient amount to start the system so that it could be taught the rest of the language.

References

Barwise, Jon, and Robin Cooper. "Generalized quantifiers and natural language." Linguistics and philosophy 4.2 (1981): 159-219.

Bateman, J. A. (1993). Ontology construction and natural language. In Proc. International Workshop on Formal Ontology. Padua, Italy, pp. 83-93.

Bisson, Gilles, Claire Nédellec, and Dolores Canamero. "Designing Clustering Methods for Ontology Building-The Mo'K Workbench." ECAI workshop on ontology learning. Vol. 31. 2000.

Braine, Martin DS. "The "natural logic" approach to reasoning." Reasoning, necessity, and logic: Developmental perspectives (1990): 133-157.

Brill, Eric, and Raymond J. Mooney. "An overview of empirical natural language processing." AI magazine 18.4 (1997): 13.

Bunt, Harry, Johan Bos, and Stephen Pulman. "Computing Meaning: Annotation, Representation, and Inference." Computing Meaning. Springer Netherlands, 2014. 1-9.

Cabrio, Elena, and Bernardo Magnini. "Decomposing Semantic Inferences."Linguistic Issues in Language Technology 9 (2013).

Carbonell, Jaime G., and Ralf D. Brown. "Anaphora resolution: a multi-strategy approach." *Proceedings of the 12th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1988.

Chao, Wynn. 1987. On ellipsis. Doctoral Dissertation, University of Massachusetts, Amherst.

Charniak, E. 1993. Statistical Language Learning. Cambridge, Mass.: MIT Press.

Charniak, Eugene, and Micha Elsner. "EM works for pronoun anaphora resolution." *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2009.

Chatzikyriakidis, S., and Z. Luo. "Formal Semantics in Modern Type Theories: Theory and Implementation." way 11 (2013): 23.

Chatzikyriakidis, Stergios, and Zhaohui Luo. "Natural Language Inference in Coq." Manuscript to be submitted to Jolliff (2013). Chung, Sandra, William A. Ladusaw, and James McCloskey. 1995. Sluicing and Logical Form. Natural Language Semantics 3: 239-282.

Collins, Michael. "Ranking algorithms for named-entity extraction: Boosting and the voted perceptron." *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002.

Dalrymple, Mary, Stuart M. Shieber, and Fernando CN Pereira. "Ellipsis and higher-order unification." *Linguistics and Philosophy* 14.4 (1991): 399-452.

de Marneffe, Marie-Catherine, et al. "Learning to distinguish valid textual entailments." Second Pascal RTE Challenge Workshop. 2006.

Ding, Chris, et al. "Orthogonal nonnegative matrix t-factorizations for clustering." Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006.

Esuli, Andrea, and Fabrizio Sebastiani. "Sentiwordnet: A publicly available lexical resource for opinion mining." Proceedings of LREC. Vol. 6. 2006.

Etzioni, Oren, et al. "Unsupervised named-entity extraction from the web: An experimental study." *Artificial Intelligence* 165.1 (2005): 91-134.

Ferrucci, David, et al. "Building Watson: An overview of the DeepQA project." AI Magazine 31.3 (2010): 59-79.

Fitting, Melvin. First-order logic and automated theorem proving. Springer Verlag, 1996.

Fyodorov, Yaroslav, Yoad Winter, and Nissim Francez. "Order-based inference in natural logic." *Logic Journal of IGPL* 11.4 (2003): 385-416.

Gabbay, Dov M., and Franz Guenthner, eds. Handbook of philosophical logic. Vol. 4. Springer, 2001.

Goodall, Grant. 1987. Parallel structures in syntax. Cambridge: Cambridge University Press.

Google. How Search Works - The Story - Inside Search - Google. Web. 3 Mar. 2014. ‹https://www.google.com/insidesearch/howsearchworks/thestory/›.

Grishman, Ralph. "Information extraction: Techniques and challenges."*Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*. Springer Berlin Heidelberg, 1997. 10-27.

Hardt, Daniel. "Verb Phrase Ellipis: Form, Meaning, and Processing." (1993).

Heim, Irene. "E-type pronouns and donkey anaphora." Linguistics and philosophy 13.2 (1990): 137-177.

Hofstadter, Douglas, and Douglas R. Hofstadter. Surfaces and Essences. Basic Books, 2010.

Hurd, Joe. "First-order proof tactics in higher-order logic theorem provers."Design and Application of Strategies/Tactics in Higher Order Logics, number NASA/CP-2003-212448 in NASA Technical Reports (2003): 56-68.

J. Sukkarieh. 2003. Mind your Language! Controlled Language for Inference Purposes. Oral presentation, Presented at the Joint Conference of the Eighth International Workshop of the European Association for Machine Translation and the Fourth Controlled Language Applications Workshop, Dublin, Ireland.

Johnson, Bernadette. "How Siri Works" 06 February 2013. HowStuffWorks.com. <http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/siri.htm> 14 March 2014.

Jurafsky, Dan, et al. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Vol. 2. Upper Saddle River: Prentice Hall, 2000.

Kaplan, Ronald M., et al. "Speed and accuracy in shallow and deep stochastic parsing." *Proceedings of HLT-NAACL*. Vol. 4. 2004.

Koller, Alexander, and Manfred Pinkal. "108. Semantic research in computational linguistics." (2012).

Lakoff, George. Linguistics and natural logic. Springer Netherlands, 1973.

Lapata, Mirella, and Frank Keller. "The Web as a Baseline: Evaluating the Performance of Unsupervised Web-based Models for a Range of NLP Tasks." HLT-NAACL. 2004.

Lasnik, Howard. 2001. When can you save a structure by destroying it? In Proceedings of the North East Linguistic Society 31, eds. Minjoo Kim and Uri Strauss, 301-320. Georgetown University: GLSA.

Lobeck, Anne. 1995. Ellipsis: Functional heads, licensing and identification. New York: Oxford University Press.

Ljunglöf, Peter, and Magdalena Siverbo. "A bilingual treebank for the FraCaS test suite." *CLT project report, University of Gothenburg* (2011).

Ljunglöf, Peter. "Practical Parsing of Parallel Multiple Context-Free Grammars."*TAG+ 11, 11th International Workshop on Tree Adjoining Grammar and Related Formalisms*. 2012.

MacCartney, Bill, and Christopher D. Manning. "Natural logic for textual inference." *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, 2007.Benthem, Johan. "Meaning: interpretation and inference." *Synthese* 73.3 (1987): 451-470.

MacCartney, Bill, and Christopher D. Manning. "Modeling semantic containment and exclusion in natural language inference." *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2008.

MacCartney, Bill, and Christopher D. Manning. "Natural Logic and Natural Language Inference." *Computing Meaning*. Springer Netherlands, 2014. 129-147.

Mahesh, Kavi, and Sergei Nirenburg. "A situated ontology for practical NLP." Proceedings of the IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing. Vol. 19. 1995.

Mani, Inderjeet, and George Wilson. "Robust temporal processing of news." Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2000.

Manning, Christopher D., and Hinrich Schütze. Foundations of statistical natural language processing. Vol. 999. Cambridge: MIT press, 1999.

Markert, Katja, Natalia Modjeska, and Malvina Nissim. "Using the web for nominal anaphora resolution." *EACL Workshop on the Computational Treatment of Anaphora*. 2003.

Merchant, Jason. 2001. The syntax of silence: sluicing, islands, and the theory of ellipsis. Oxford: Oxford University Press.

Merchant, Jason. "Variable island repair under ellipsis." Topics in ellipsis (2008): 132-153.

Molina, Antonio, and Ferran Pla. "Shallow parsing using specialized hmms."*The Journal of Machine Learning Research* 2 (2002): 595-613.

Mitkov, R.: Pronoun resolution: the practical alternative. In: Proceedings of the Discourse Anaphora and Anaphor Resolution Colloquium (DAARC), Lancaster, UK (1996)Mitkov, R.: Robust pronoun resolution with limited knowledge. In: Proceedings of the 18th International Conference on Computational Linguistics (COLING'98/ACL'98), Montreal, Quebec, Canada, August 10-14, 1998, pp. 867–875 (1998)

Mitkov, Ruslan, et al. "Anaphora resolution: To what extent does it help NLP applications?." Anaphora: Analysis, Algorithms and Applications. Springer Berlin Heidelberg, 2007. 179-190.

Montague, R. "Formal Philosophy: Selected Papers of Richard Montague", R. Thomason (ed), Yale University Press. (1974),

Nguyen, Dat PT, Yutaka Matsuo, and Mitsuru Ishizuka. "Relation extraction from wikipedia using subtree mining." *Proceedings of the National Conference on Artificial Intelligence*. Vol. 22. No. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

Nirenburg, S., Raskin, V. and Onyshkevych, B. (1995). Apologiae Ontologiae. Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation. Leuven, Belgium, July.

Noy, Natalya F. "Semantic integration: a survey of ontology-based approaches." ACM Sigmod Record 33.4 (2004): 65-70.

Ogneva, M, http://mashable.com/2010/04/19/sentiment-analysis/, retrieved 2013-09-30. How Companies Can Use Sentiment Analysis to Improve Their BusinessGodbole, Namrata, Manja Srinivasaiah, and Steven Skiena. "Large-Scale Sentiment Analysis for News and Blogs." ICWSM 7 (2007).

Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004.

Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and trends in information retrieval 2.1-2 (2008): 1-135.

Partee, Barbara. "Noun phrase interpretation and type-shifting principles."*Studies in discourse representation theory and the theory of generalized quantifiers* 8 (1987): 115-143.

Pease, Adam, and William Murray. "An English to Logic Translator for ontology-based knowledge representation languages." Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on. IEEE, 2003.

Ravichandran, Deepak, Patrick Pantel, and Eduard Hovy. "Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering." Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005.

R. Cooper, R. Crouch, J. Van Eijck, C. Fox, J. Van Genabith, J. Jaspars, H. Kamp, M. Pinkal, D. Milward, M. Poesio, and S. Pulman. 1996. Using the Framework. Technical report, FraCaS: A Framework for Computational Semantics, FraCaS deliverable D16.

Reeve, Lawrence, and Hyoil Han. "Survey of semantic annotation platforms." Proceedings of the 2005 ACM symposium on Applied computing. ACM, 2005.

Riemsdijk, Henk van. 1982. A case study in syntactic markedness. Dordrecht, The Netherlands: Foris.

Romero, Maribel. 1998. Focus and reconstruction effects in wh-phrases. Doctoral Dissertation, University of Massachusetts, Amherst.

Ross, John Robert 1969. Guess who? In Chicago Linguistics Society, eds. Robert I. Binnick, Alice Davison, Georgia M. Green, and Jerry L. Morgan, 252-286. Chicago, IL: The Chicago Linguistics Society, University of Chicago.Leven, Lori. 1982. Sluicing: A lexical interpretation procedure. In The mental representation of grammaticalrelations, ed. Joan Bresnan, 590-654. Cambridge, MA: MIT Press.

Schmid, Helmut. "Probabilistic part-of-speech tagging using decision trees."*Proceedings of international conference on new methods in language processing*. Vol. 12. 1994.

Schubert, Lenhart K., and Francis Jeffry Pelletier. "From English to Logic: Context-free computation of conventional logical translation." *Computational Linguistics* 8.1 (1982): 26-44.

Sha, Fei, and Fernando Pereira. "Shallow parsing with conditional random fields." Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003.

Soon, Wee Meng, Hwee Tou Ng, and Daniel Chung Yong Lim. "A machine learning approach to coreference resolution of noun phrases." *Computational linguistics* 27.4 (2001): 521-544.

Steedman, Mark. 1990. Gapping as constituent coordination. Linguistics and Philosophy 13:207–264.

Steedman, Mark. 1996. Surface structure and interpretation. Cambridge, Massachusetts: MIT Press.

Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2005.

Thijsse, Elias. "On some proposed universals of natural language." *Studies in modeltheoretic semantics* (1983): 19-36.

Vargas-Vera, Maria, Enrico Motta, and John Domingue. "AQUA: An Ontology-Driven Question Answering System." New Directions in Question Answering. 2003.

van Benthem, J. F. A. K. "A brief history of natural logic." (2008).

Wiebe, Janyce, et al. "An empirical approach to temporal reference resolution." J. Artif. Intell. Res. (JAIR) 9 (1998): 247-293.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity in phrase-level sentiment analysis." Proceedings of the conference on Human Language Zelenko, Dmitry, Chinatsu Aone, and Anthony Richardella. "Kernel methods for relation extraction." *The Journal of Machine Learning Research* 3 (2003): 1083-1106.

Zoerner, Cyril Edward, III. 1995. Coordination: The syntax of &P. Doctoral Dissertation, University of California at Irvine.

Zhuang, Li, Feng Jing, and Xiao-Yan Zhu. "Movie review mining and summarization." Proceedings of the 15th ACM international conference on Information and knowledge management. ACM, 2006.

Appendix A - Glossary

Conservativity	Q As are Bs == Q As are As who are Bs
Monotonicity (upwards on second argument)	Q As are Bs and all Bs are Cs < Q As are Cs
Monotonicity (downwards on second argument)	Q As are Bs and all Cs are Bs < Q As are Cs
Monotonicity (upwards on first argument)	Q As are Bs and all As are Cs < Q Cs are Bs
Monotonicity (downwards on first argument)	Q As are Bs and all Cs are As < Q Cs are Bs
Conjoined Noun Phrases	Several noun phrases combined together
Definite Plurals	Definite plurals can often be non-anaphoric and behave like universally quantified noun phrases (90). However, as with (generic) bare plurals, the force of the quantification can also be less than universal (91). Whether this lessening of quantificational force is due to the noun phrase or to the predicate of which the NP is an argument is unclear (92, 93).
Bare Plurals	Bare plurals can exhibit existential, (quasi) universal, generic or dependent plural behaviour.
Dependent Plurals	Define all of something then inquire about something within that set.
Expectation Maximazation Algorithm	is an iterative method for finding maximum likelihood or maximum a posteriori

Appendix B - NLP information

The following information is taken from http://bulba.sdsu.edu/jeanette/thesis/PennTags.html

Clause Level

S - simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a *wh*-word and that does not exhibit subject-verb inversion.
SBAR - Clause introduced by a (possibly empty) subordinating conjunction.
SBARQ - Direct question introduced by a *wh*-word or a *wh*-phrase. Indirect questions and relative clauses should be bracketed as SBAR, not SBARQ.
SINV - Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ - Inverted yes/no question, or main clause of a *wh*-question, following the *wh*-phrase in SBARQ.

Phrase Level

ADJP - Adjective Phrase.
ADVP - Adverb Phrase.
CONJP - Conjunction Phrase.
FRAG - Fragment.
INTJ - Interjection. Corresponds approximately to the part-of-speech tag UH.
LST - List marker. Includes surrounding punctuation.
NAC - Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP - Noun Phrase.
NX - Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to N-bar level but used quite differently.
PP - Prepositional Phrase.
PRN - Parenthetical.
PRT - Particle. Category for words that should be tagged RP.
QP - Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC - Reduced Relative Clause.
UCP - Unlike Coordinated Phrase.
VP - Vereb Phrase.
WHADJP - *Wh*-adjective Phrase. Adjectival phrase containing a *wh*-adverb, as in *how hot*.
WHAVP - *Wh*-adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a *wh*-adverb such as *how* or *why*.
WHNP - *Wh*-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some *wh*-word, e.g. *who*, *which book*, *whose daughter*, *none of which*, or *how many leopards*.
WHPP - *Wh*-prepositional Phrase. Prepositional phrase containing a *wh*-noun phrase (such as *of which* or *by whose authority*) that either introduces a PP gap or is contained by a WHNP.
X - Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing *the...the*-constructions.

Word level

CC - Coordinating conjunction
CD - Cardinal number
DT - Determiner
EX - Existential there
FW - Foreign word
IN - Preposition or subordinating conjunction
JJ - Adjective
JJR - Adjective, comparative
JJS - Adjective, superlative
LS - List item marker
MD - Modal
NN - Noun, singular or mass
NNS - Noun, plural
NNP - Proper noun, singular
NNPS - Proper noun, plural
PDT - Predeterminer
POS - Possessive ending
PRP - Personal pronoun
PRP$ - Possessive pronoun (prolog version PRP-S)
RB - Adverb
RBR - Adverb, comparative
RBS - Adverb, superlative
RP - Particle
SYM - Symbol
TO - to
UH - Interjection
VB - Verb, base form
VBD - Verb, past tense
VBG - Verb, gerund or present participle
VBN - Verb, past participle
VBP - Verb, non-3rd person singular present
VBZ - Verb, 3rd person singular present
WDT - Wh-determiner
WP - Wh-pronoun
WP$ - Possessive wh-pronoun (prolog version WP-S)
WRB - Wh-adverb

Function tags

Form/function discrepancies

-ADV (adverbial) - marks a constituent other than ADVP or PP when it is used adverbially (e.g. NPs or free ("headless" relatives). However, constituents that themselves are modifying an ADVP generally do not get -ADV. If a more specific tag is available (for example, -TMP) then it is used alone and -ADV is implied. See the Adverbials section.
-NOM (nominal) - marks free ("headless") relatives and gerunds when they act nominally.

Grammatical role

-DTV (dative) - marks the dative object in the unshifted form of the double object construction. If the preposition introducing the "dative" object is *for*, it is considered benefactive (-BNF). -DTV (and -BNF) is only used after verbs that can undergo dative shift.
-LGS (logical subject) - is used to mark the logical subject in passives. It attaches to the NP object of *by* and not to the PP node itself.
-PRD (predicate) - marks any predicate that is not VP. In the *do so* construction, the *so* is annotated as a predicate.
-PUT - marks the locative complement of *put*.
-SBJ (surface subject) - marks the structural surface subject of both matrix and embedded clauses, including those with null subjects.
-TPC ("topicalized") - marks elements that appear before the subject in a declarative sentence, but in two cases only:

1. if the front element is associated with a *T* in the position of the gap.
2. if the fronted element is left-dislocated (i.e. it is associated with a resumptive pronoun in the position of the gap).

-VOC (vocative) - marks nouns of address, regardless of their position in the sentence. It is not coindexed to the subject and not get -TPC when it is sentence-initial.

Adverbials

Adverbials are generally VP adjuncts.

-BNF (benefactive) - marks the beneficiary of an action (attaches to NP or PP).
This tag is used *only* when (1) the verb can undergo dative shift and (2) the prepositional variant (with the same meaning) uses *for*. The prepositional objects of dative-shifting verbs with other prepositions than *for* (such as *to* or *of*) are annotated -DTV.
-DIR (direction) - marks adverbials that answer the questions "from where?" and "to where?" It implies motion, which can be metaphorical as in *"...rose 5 pts. to 57-1/2"* or *"increased 70% to 5.8 billion yen"* -DIR is most often used with verbs of motion/transit and financial verbs.
-EXT (extent) - marks adverbial phrases that describe the spatial extent of an activity. -EXT was incorporated primarily for cases of movement in financial space, but is also used in analogous situations elsewhere. Obligatory complements do not receive -EXT. Words such as *fully* and

completely are absolutes and do **not** receive -EXT.
-LOC (locative) - marks adverbials that indicate place/setting of the event. -LOC may also indicate metaphorical location. There is likely to be some varation in the use of -LOC due to differing annotator interpretations. In cases where the annotator is faced with a choice between -LOC or -TMP, the default is -LOC. In cases involving SBAR, SBAR should not receive -LOC. -LOC has some uses that are not adverbial, such as with place names that are adjoined to other NPs and NAC-LOC premodifiers of NPs. The special tag -PUT is used for the locative argument of *put*.
-MNR (manner) - marks adverbials that indicate manner, including instrument phrases.
-PRP (purpose or reason) - marks purpose or reason clauses and PPs.
-TMP (temporal) - marks temporal or aspectual adverbials that answer the questions *when, how often*, or *how long*. It has some uses that are not strictly adverbial, auch as with dates that modify other NPs at S- or VP-level. In cases of apposition involving SBAR, the SBAR should not be labeled -TMP. Only in "financialspeak," and only when the dominating PP is a PP-DIR, may temporal modifiers be put at PP object level. Note that -TMP is not used in possessive phrases.

Miscellaneous

-CLR (closely related) - marks constituents that occupy some middle ground between arguments and adjunct of the verb phrase. These roughly correspond to "predication adjuncts", prepositional ditransitives, and some "phrasel verbs". Although constituents marked with -CLR are not strictly speaking complements, they are treated as complements whenever it makes a bracketing difference. The precise meaning of -CLR depends somewhat on the category of the phrase.

- **on S or SBAR** - These categories are usually arguments, so the -CLR tag indicates that the clause is more adverbial than normal clausal arguments. The most common case is the infinitival semi-complement of *use*, but there are a variety of other cases.
- **on PP, ADVP, SBAR-PRP, etc** - On categories that are ordinarily interpreted as (adjunct) adverbials, -CLR indicates a somewhat closer relationship to the verb. For example:
    - Prepositional Ditransitives
      In order to ensure consistency, the Treebank recognizes only a limited class of verbs that take more than one complement (-DTV and -PUT and Small Clauses) Verbs that fall outside these classes (including most of the prepositional ditransitive verbs in class [D2]) are often associated with -CLR.
    - Phrasal verbs
      Phrasal verbs are also annotated with -CLR or a combination of -PRT and PP-CLR. Words that are considered borderline between particle and adverb are often bracketed with ADVP-CLR.
    - Predication Adjuncts
      Many of Quirk's predication adjuncts are annotated with -CLR.
- **on NP** - To the extent that -CLR is used on NPs, it indicates that the NP is part of some kind of "fixed phrase" or expression, such as *take care of*. Variation is more likely for NPs than for other uses of -CLR.

-CLF (cleft) - marks it-clefts ("true clefts") and may be added to the labels S, SINV, or SQ.
-HLN (headline) - marks headlines and datelines. Note that headlines and datelines always constitute a unit of text that is structurally independent from the following sentence.
-TTL (title) - is attached to the top node of a title when this title appears inside running text. -TTL implies -NOM. The internal structure of the title is bracketed as usual.

Appendix C - Generalized Quantifiers

Section 1.1 - Example 1 – Problem 29 from FraCaS Test Suite

Table C-1. Problem 1 from Section 1.1

P1	An Italian is a person.
P2	An Italian became the world's greatest tenor.
Q	Was there an Italian who became the world's greatest tenor?

In Table C-1, we have two premises and one question. Following the algorithm we parse the first premise and generate an ordered list of grammar trees using the OpenNLP and StanfordNLP framework.

In Table C-1, Premise 1 is added in to provide sufficient information for this question. Without this additional premise the system would not be able to answer the question as it would not know that an Italian is a person.

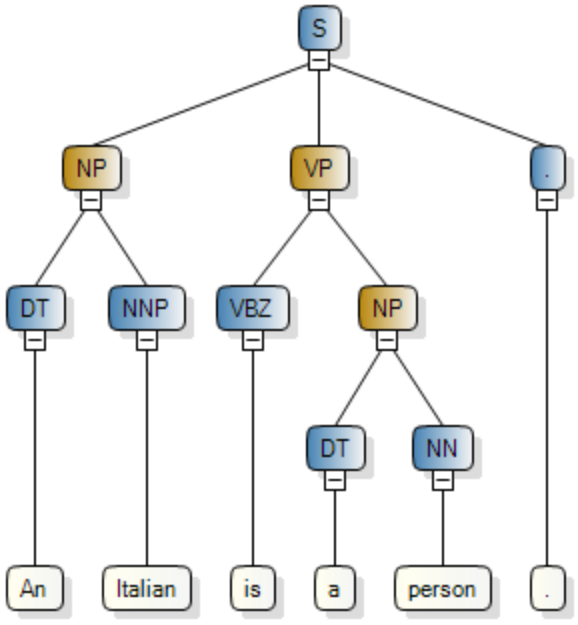


Figure C-1 Premise 1 - Problem 1

The first grammar tree for premise 1, as shown in Figure C-1, in the ordered list is a valid grammar tree as will be shown by following steps 1.a.i-1.a.iv.

The set of parse trees are returned back from openNLP and stanfordNLP. For this particular input, only one parse tree is actually returned and is depicted Figure C-1. This parse tree is also the best valid parse tree for this input.

The system starts at the top of the tree and works its way down left to right looking at each node and analyzing what its children nodes are (if applicable). First, we look at the S node. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object. Now looking at the [NP] node (Noun Phrase), it has two children DT node, which stands for article, and NNP node (Proper Noun, singular). Now looking at [An], we know that this particular noun is referring to a generic [NNP], not some specific [NNP], which is important for matching purposes later on. Finally, we finish this noun object by placing Italian as the name for this particular object. While, we can load other properties of the noun (if any) at this point, this is omitted as this example does not utilize any inference, nor require background knowledge from the verbs assuming particular attributes. At this point, we have placed into the temporary ontology an instance of an Italian, where the quantity is only 1 and that it contains no attributes at this point in time (or at all in this example).

Next, we look at the [VP] node (Verb Phrase). The [VP] node contains two children, both a [VBZ] node (Verb, third-person singular present), and a [NP] node. We look at the verb 'is' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a present tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurs in present time at

time t = 0. (The time related information for the specific verb 'is' is shown later to not be of value).

We now have the final noun object for this verb phrase to process. From how the previous noun object was constructed, this one follows the exact same except that we have a [NN] (Noun, singular or mass) instead of [NNP]. Given this, we create a generic instance of person. We add this generic instance of person to the temp ontology as well.

Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of Italian to the instance of person via the word 'is'.

The system now takes a look at everything derived from the first premise as a whole and begins to build the temp ontology shown in Figure C-2. It looks at every predicate begins to assimilate the data. The only predicate (of interest) in this particular premise is the 'is' predicate. Since there are no conditions on the data shown in this premise it is a rather basic application of one of the rules for 'is'. The rule applied in this case is chosen based on the type of the parameters used in this predicate (other predicates/rules can look at not just the type but also properties/attributes/etc. to identify which rule to use). In this case, they are both instances, which is to say we need to create a parent child relationship between 'Italian' and 'Person'.

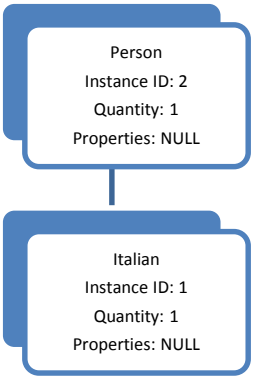


Figure C-2 Temp Ontology

Now that all predicates within the first premise have been processed, we merge this temp ontology into the main ontology. This is a trivial process for this particular case as there currently exists nothing in the main ontology. The system simply copies every instance/class/relation over.

Once the ontologies have been merged, the main ontology is analyzed to look for new relationships that can be constructed by this new knowledge. Since an instance can only be matched to another instance if it matches a parent class, aka they are of the same concept, and we must construct a parent class for any instance that currently has no parent class. For this premise, we have two instances both 'Italian' and 'Person' neither has a parent class. We construct a parent class for each of these maintaining the exact properties as seen in them. Which in this case, neither instance has any properties and note that the quantity is 1, so we create a class 'Italian' that has no properties and has a quantity of 1. We make this class the parent of the instance version of 'Italian'. We do the same for instance 'Person' and we end up with what we see in Figure C-3.

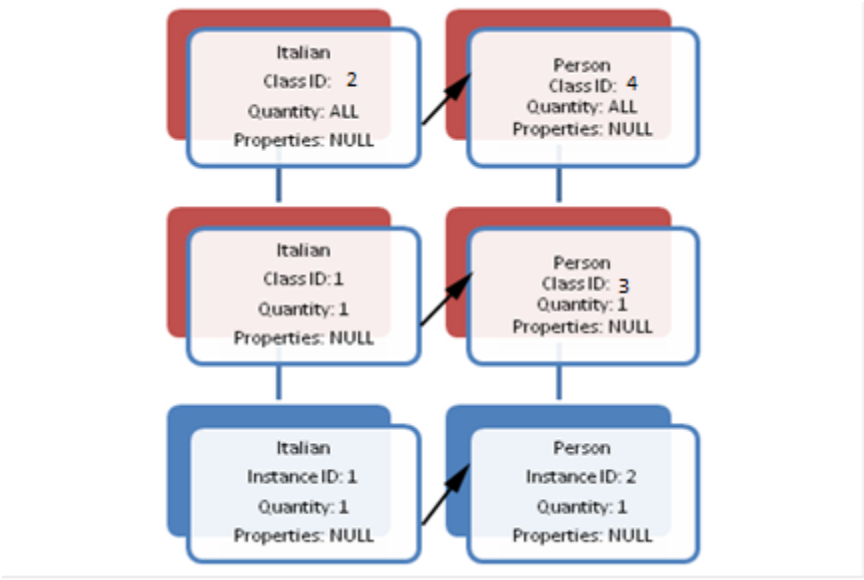


Figure C-3 Generated Knowledge

If there can be one of something such as 1 person, there can be the set of all persons. So we apply this concept to all classes that have a quantity less than 'all', we build a parent for each

of these and set the quantity to ‘all’. While not shown in this example, the application of this helps in matching.

The same general steps are followed for premise 2, ‘An Italian became the world’s greatest tenor.’. The system retrieves all parses from openNLP and stanfordNLP. In this case again, only 1 parse is returned as shown in Figure C-4. This parse also happens to be a valid parse.

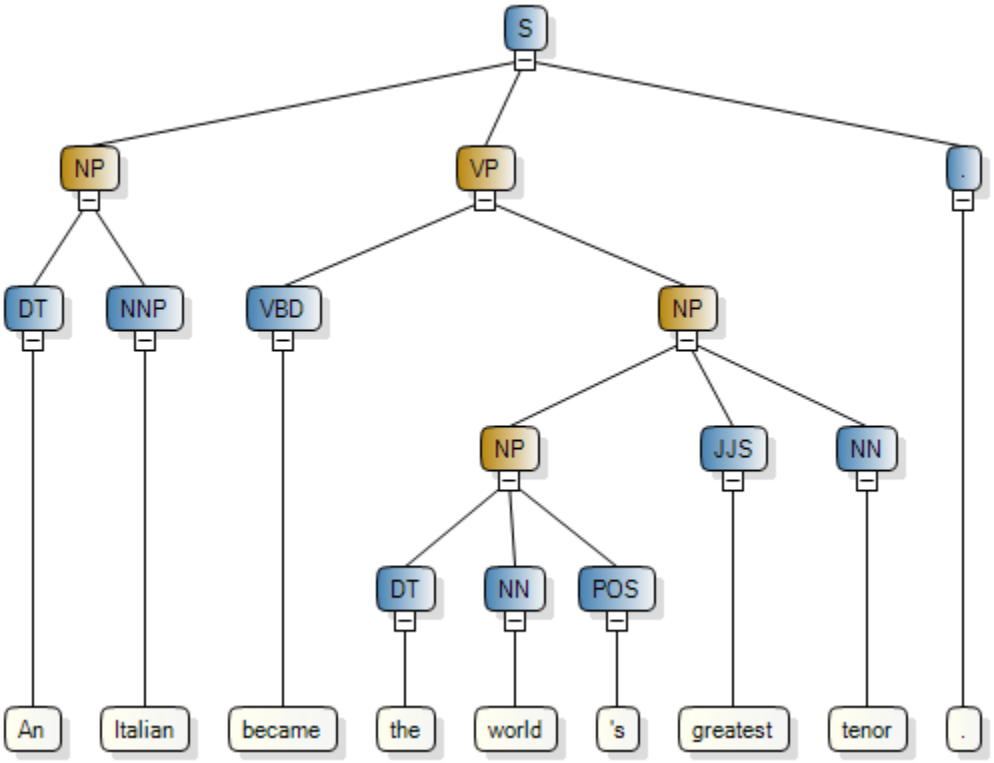


Figure C-4 Parse for Premise 2 - Problem 1

This premise has some similarities to the first premise. From now on out, obvious details such as if it is a statement (premise) or a question will just be assumed the reader understands how it is derived. This statement has two children, both [NP] and [VP] nodes. The [NP] node generates an instance of ‘Italian’ as we saw previously.

The [VP] node contains a new node type [VBD] (Verb, past tense). At this point the system begins to build a new relationship, but first it must analyze the [NP] node. This [NP]

node is complicated in the fact that it has another [NP] node as a child of it. Let's take a look at the [NP] node that contains three children, the [DT], [NN], and [POS] (Possessive ending). These three children nodes determine that a possessive noun object must be built, but with this particular [NP] node, it is unknown at this point in time what it actually possesses. This is determined at the parent [NP] node, it checks the type of the child [NP] node (noun object) and sees that the child [NP] node is a possessive noun object and builds a possessive relationship between the child [NP] node and the [NN] (tenor). There is an instance 'tenor' that is constructed that has an attribute of greatest from the [JJS] node. When the attribute is constructed, it is first analyzed to see if the adjective has any other meanings or modifications that must be made to the part of the statement. Now that the instance 'tenor' has been built it can be added into the possessive relationship between it and the instance world which completes this [NP] node within the [VP] node. The system builds the relationship between the instance 'Italian' and the instance 'world' via the predicate became. Though in doing this it normalizes the verb became to the present tense verb become and like any verb adds in a time component that maintains when in time this information occurs according to the system.

Since the verb was a past tense verb, and this is 2nd verb encountered by the system, the system increments the system clock and indicates that this information occurred prior to the system clock as that is all that can be gathered from this information. This information is important for matching purposes later on as certain pieces of information are only valid during certain points in time.

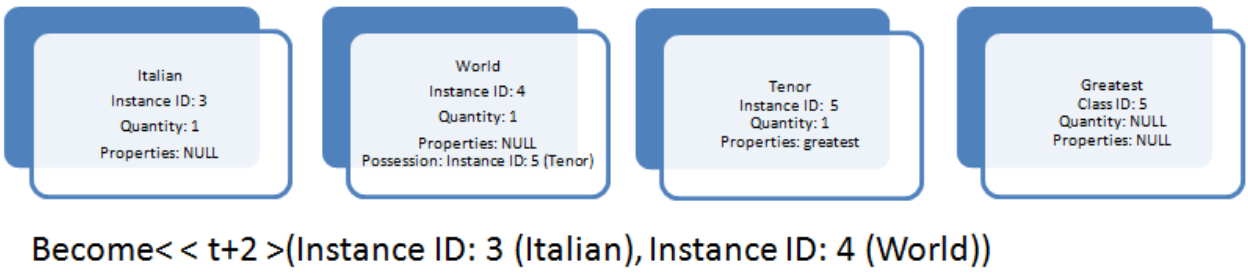


Figure C-5 Temporary Ontology for Premise 2 - Problem 1

Now that the temporary ontology for the premise has been constructed we must merge this into the current working ontology before we try to extract new information from it as that information may already be present so there is no need to perform duplicate work. We take a look at every

instance/class/relationship that is generated in the temporary ontology and attempt to insert it into the main ontology. All instances will always be added into the main ontology. Relationships are only inserted if they provide new information in anyways. As the relationship from 'become' does not currently exist in the ontology it is trivially added in. After merging all the elements into the onology we are left with the new ontology shown in Figure C-6.

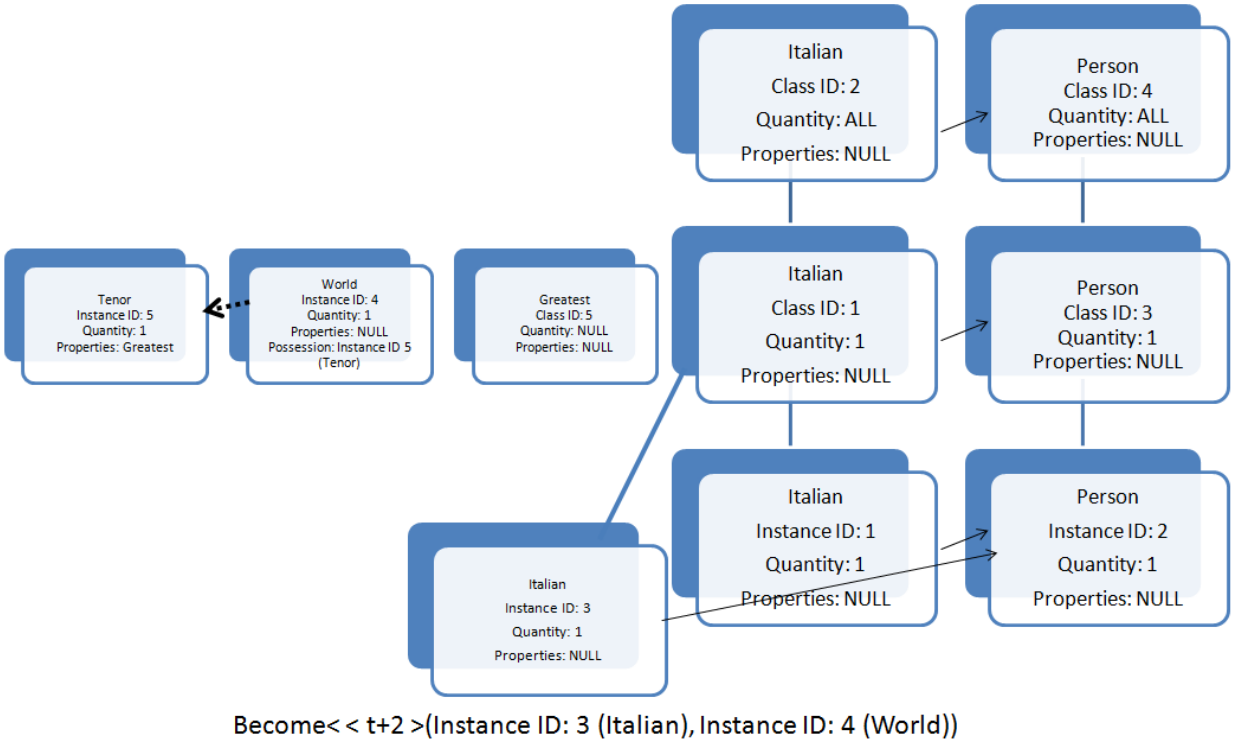


Figure C-6 Merged Onotology from Premise 2 - Problem 1

After merging the temporary ontology with the main ontology we must try to see if there is any new information to add or construct in the ontology. We take a look at every element within the ontology to see what we can add and where. All instances, that don't have any parents will have a parent that has the minimal quantity set, with the same attributes. Then a parent for that is added with the same set with no attributes (only if attributes were applicable in the first place), then yet another with the parent being where it is the set of ALL instances of a particular type, aka the parent concept that contains all children concepts. All new information that is constructed is shown in red in Figure C-7.

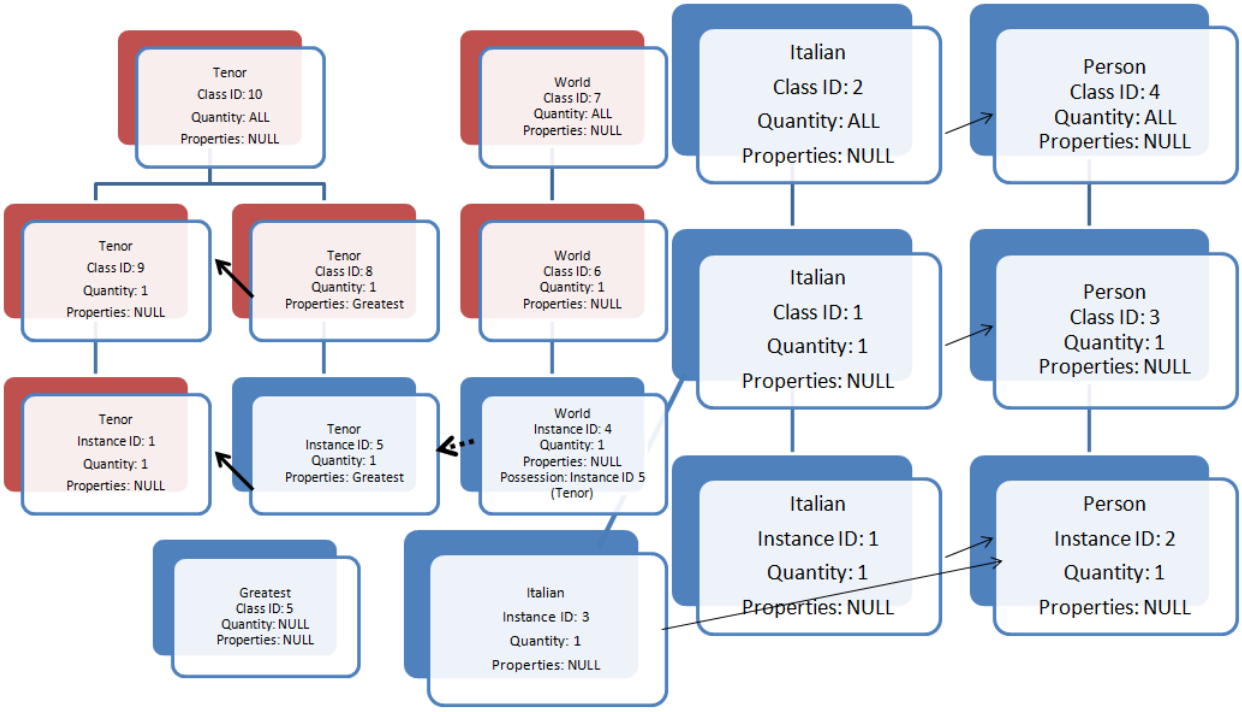


Figure C-7 Updated Onotology from Premise 2 - Problem 1

The question is the final input to analyze. The question has four parses shown in Figure C-8, Figure C-9, Figure C-10, and Figure C-11. Since there is more than one parse, the system must go through each parse until it finds a valid parse. The problem in Figure C-8 is that the word there is of the wrong type. It should not be an [RB] within a noun phrase. (Reminder: adverbs modify a verb, adjective, or another adverb.) So we must rule this particular parse out.

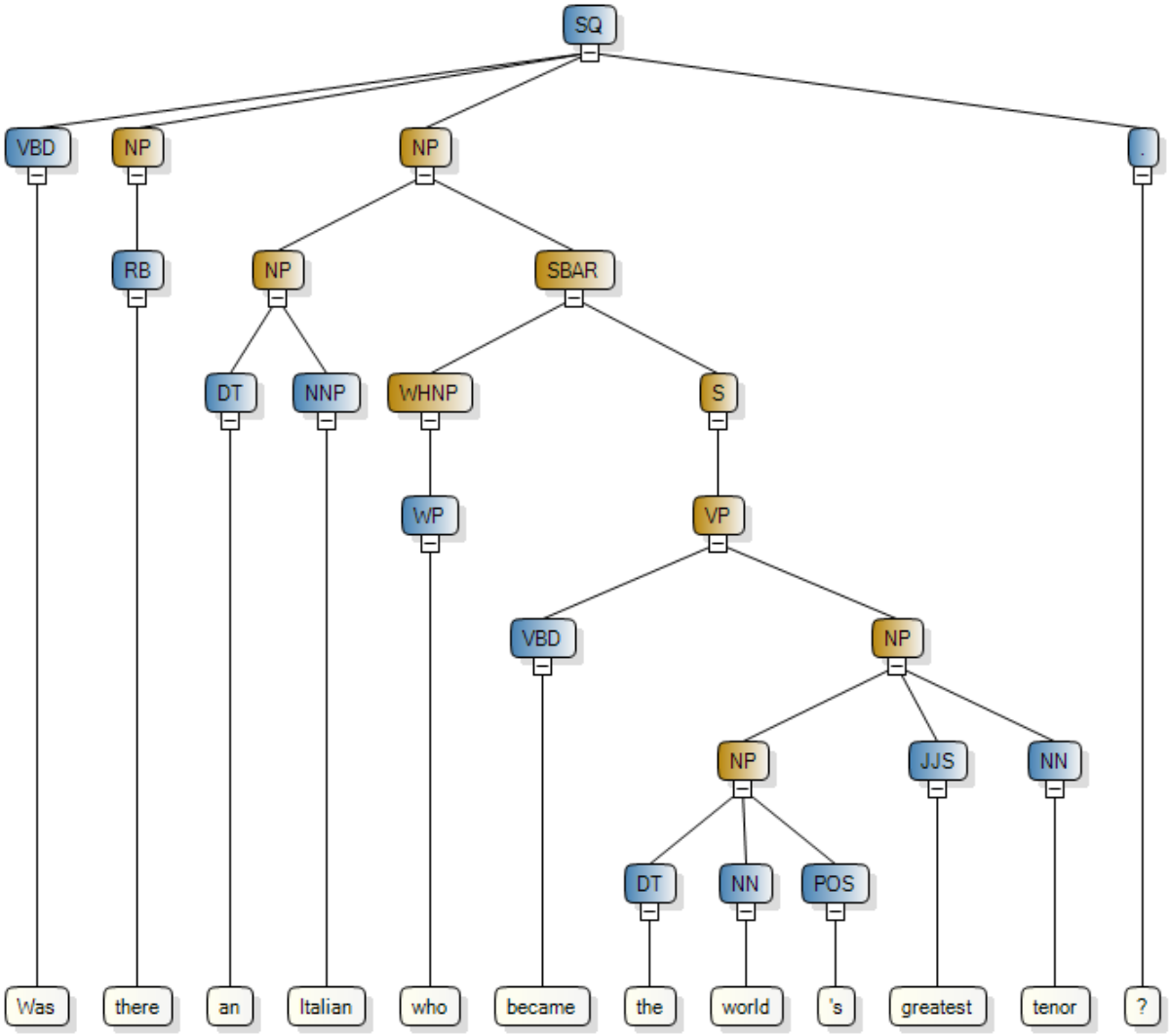


Figure C-8 Parse 1 for Question - Problem 1

The same problem exists in Figure C-9. The change in this parse though, which doesn't fix the original problem, is that the word Italian is an adjective [JJ]. So the system must rule this particular parse out as well.

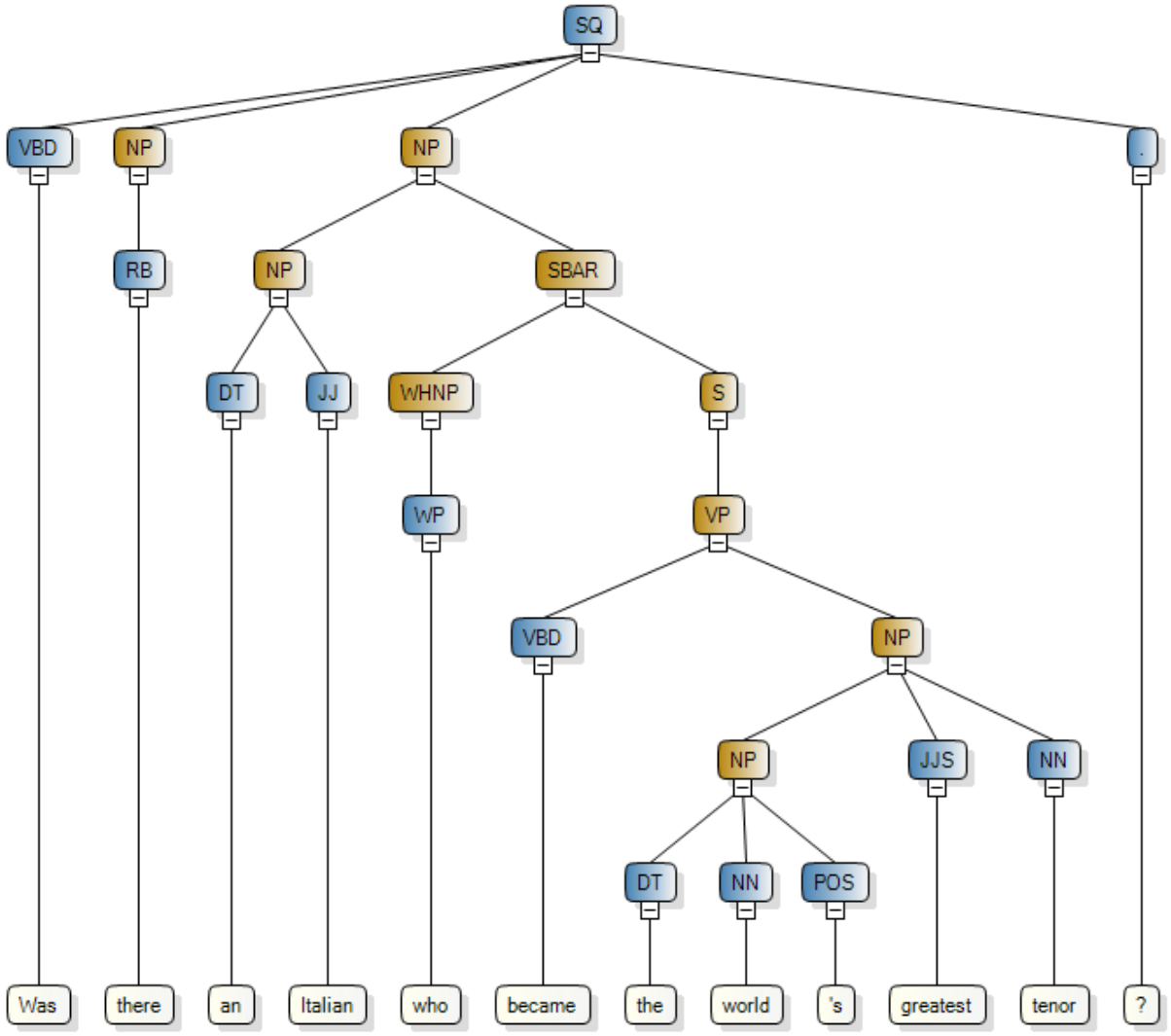


Figure C-9 Parse 2 for Question - Problem 1

Figure C-10 is actually a correct parse. The word ‘there’ has a correct type of [EX] (Existential there). The change in the type for the word ‘there’ uses the meaning effectively as, does there exist an Italian who …, which is what the question is actually asking.

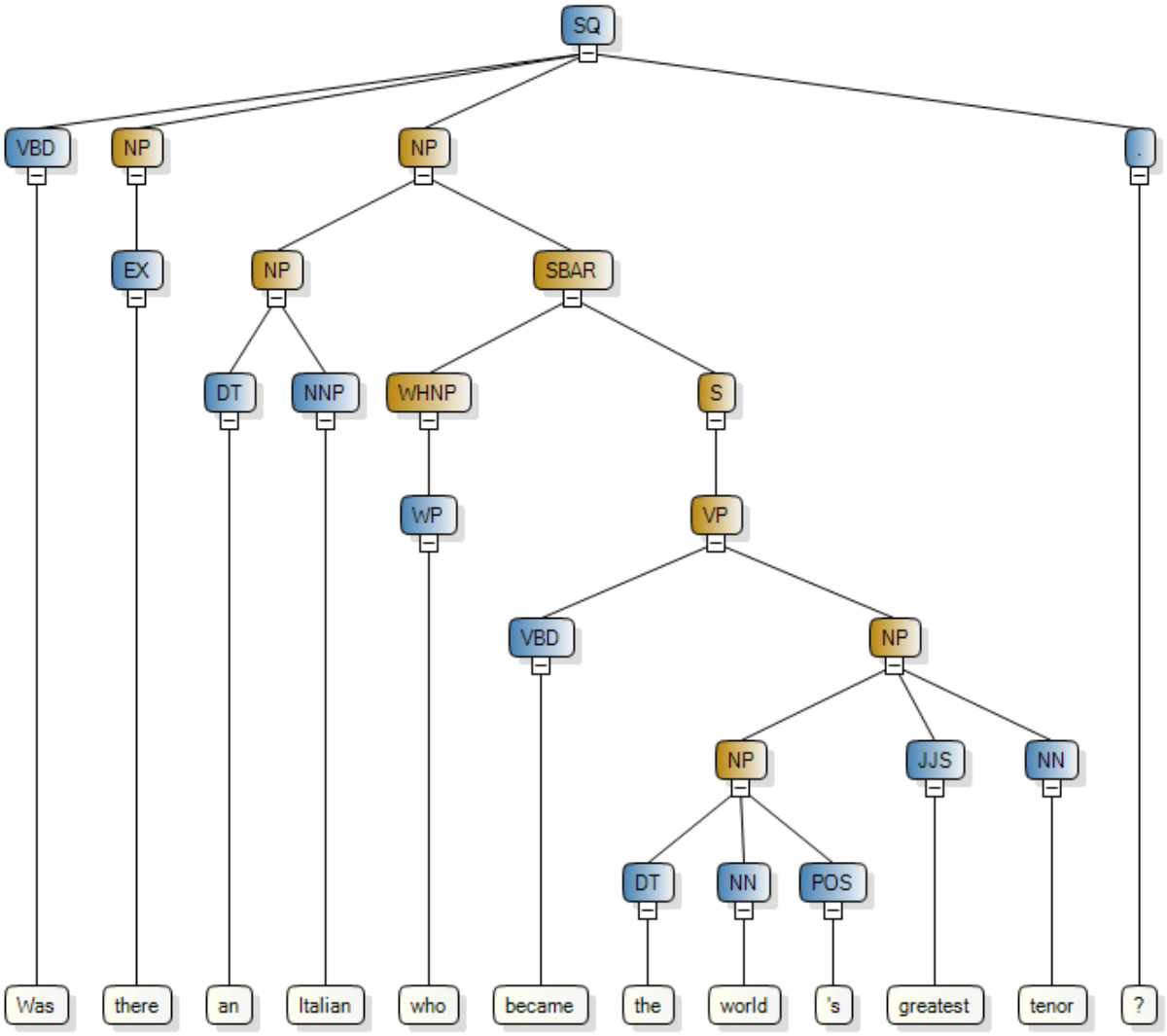


Figure C-10 Parse 3 for Question - Problem 1

Since a valid parse was found in Figure C-10, there is no need to look at the parse found in Figure C-11 regardless of if the parse has a valid parsing or not.

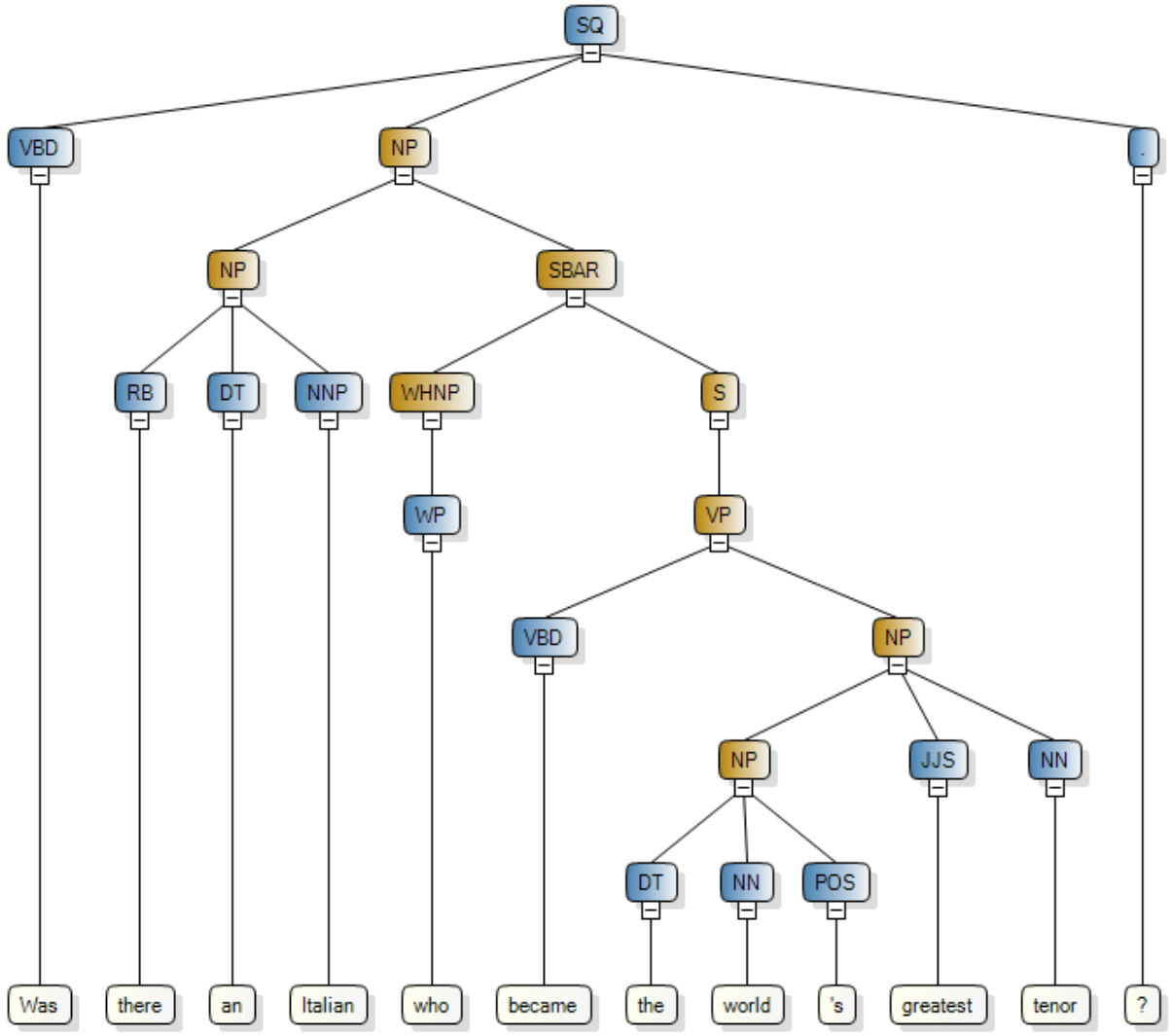


Figure C-11 Parse 4 for Question - Problem 1

Since we have found a valid parse in Figure C-10, we need to process it to see if it produces a valid output. It is possible even with a valid parse, that the words within the parse could mean that the parse was in fact not correct. If you notice, there is a new parent for this particular parse tree [SQ] (Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ). It has four children, [VBD], [NP], [NP], and [.]. These four types indicate that it is a yes no type question without further inspection of the nodes. We see the first node is [VBD] (past tense verb) and load the verb into the yes no question. Next, we have the first of two [NP]. We load a noun object which is an existential there and insert this into the yes no question object. The 2nd [NP] from the [SQ] perspective contains a [NP] and a [SBAR]

(variable type). Often times [SBAR] plays the role of adding in a condition or restriction on the [NP]as it is paired with in this case. Working within [SQ] [NP] (the 2nd NP), we create a noun object from left child [NP] as we have before in the previous premise. A single instance of 'Italian' is inserted into the temporary ontology. Now the [SBAR], in this case, is going to be a question object as defined by the [WHNP] (Any Prep phrase which has a relative pronoun, Prepositional phrase) and the [S] (statement). For this question object, we first add the question word 'who' in. The word 'who' has special meaning associated with it as any noun it is trying to relate, must be of type person (easily changed to something that is alive but a person is sufficient for this test suite). The [S] is loaded into the question object which consists only of a [VP] (verb phrase). This verb phrase is loaded just the same as seen in the previous premise. To rehash how that works: An instance of world is created and an instance of tenor is created and inserted into the temporary ontology. The instance of world contains the possession of tenor. An attribute 'greatest' is added to tenor. A relation with 'became' and world cannot quite exist just yet as we need to know what fills in that relation. Now that we have finished loading the [SBAR] which was a question in this particular case, we can complete the parent to [SBAR] which is [NP] which will build a Noun Question Object which binds the 'Italian' as the subject to the question that is posed. Now that the Noun Question object has been constructed, it knows the subject of the question to insert and it can complete the relationship with became. In addition to the relationship became(Italian, world) a constraint is added to the subject 'Italian' due to the word who, which tells us that the subject Italian is a person. As always, everything is normalized and the verb 'became' becomes the word 'become' noting the fact that it is past tense occurring before time 3 (3rd input to the system). Through processing this question we have constructed an ontology that is shown in Figure C-12.

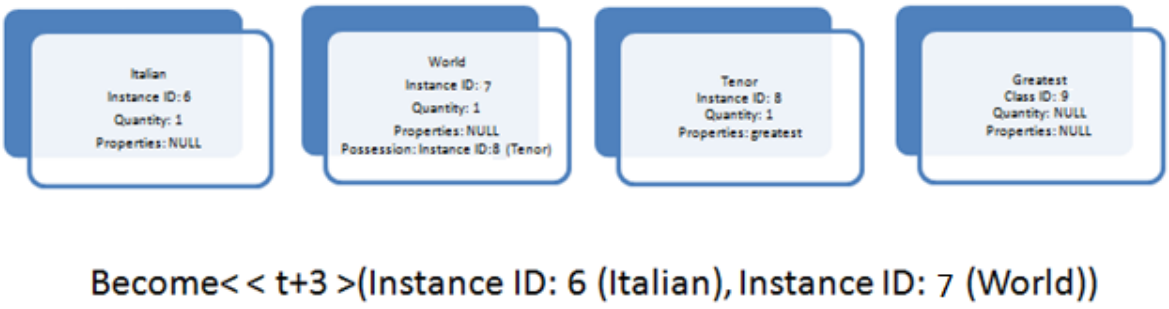


Figure C-12 Temporary Ontology for Question for Problem 1

To answer this particular question, we see that there is no missing elements or unknowns to fill in; we must first match each particular instance with everything in the ontology to see which items are potential candidates for matching the particular constraints of the temporary ontology (aka the question). We look at all possible combinations (very expensive operation) to see when if we have a match a complete match across all constraints on the temp ontology.

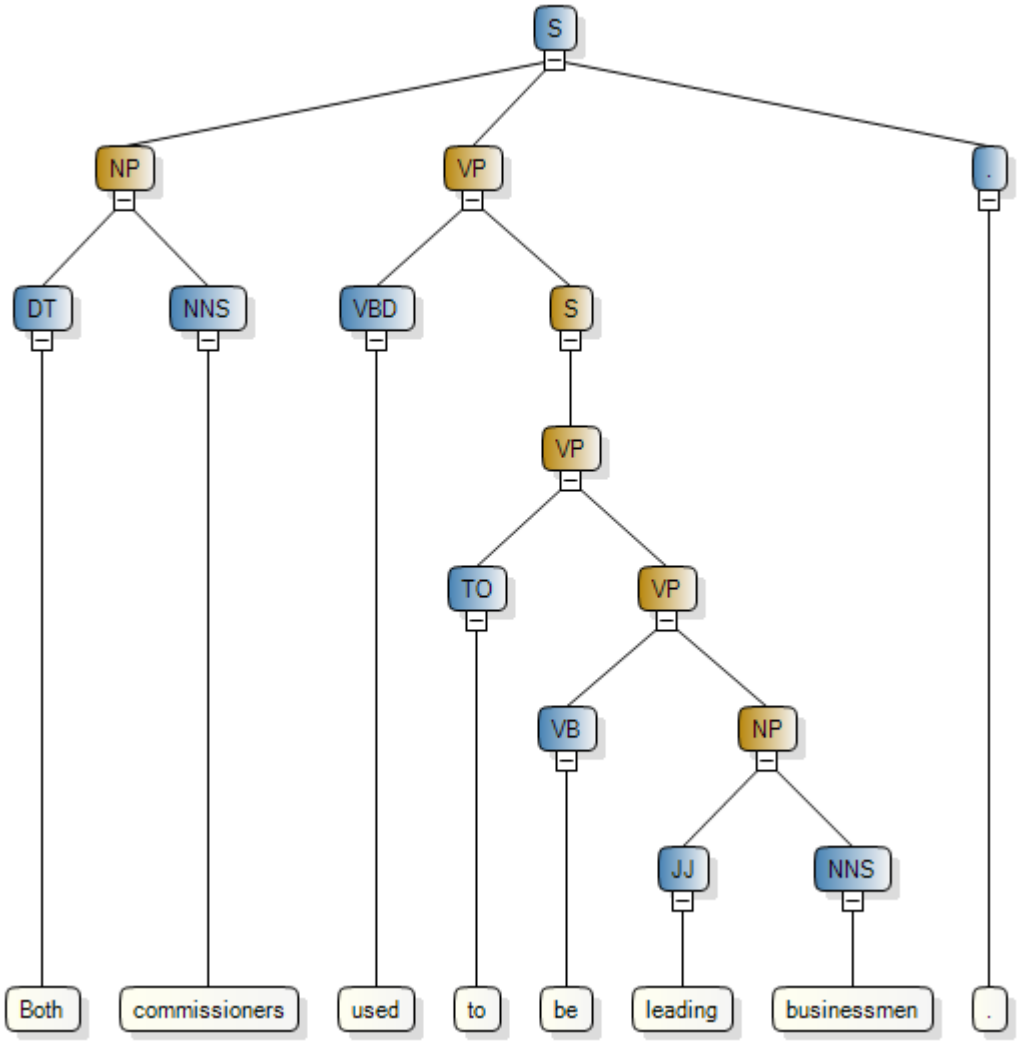
If we can find an instance of Italian that is also a person in the existing ontology then we have a match. There are two to be found, instance ID 1 and 3. Previously, we learned that any Italian is a person as depicted in Figure C-2. There is only one other instance of world, which is ID 4 and it matches the quantity, and it also has the same possession that I will show matches as well. Tenor also matches in the fact there is an instance of a tenor found that has the property of greatest. Greatest simply exists in the ontology and trivially matches. Once we have found all matches for the instances, we must check do these particular combination of matches match the relationship(s). To match this relationship 'become', first we are looking a relationship in the main ontology that has the same name. This is found, next we see if the time components of each match. This question is looking for the past tense of become (which is already past tense). So we know that the time we are looking for in which this question could be satisfied if the time of the pre-existing relationship has any time that occurs before time 3. The relationship from the 2nd premise generated a relationship that occurred before time 2. So this is a match, next we move on to check both terms (Italian, and World). Those both have matches that exist in the relationship thus we have found a successful answer to the question. We can at this point answer yes to the question which is the correct answer.

Section 1.2 - Example 2 – Problem 29 from FraCaS Test Suite

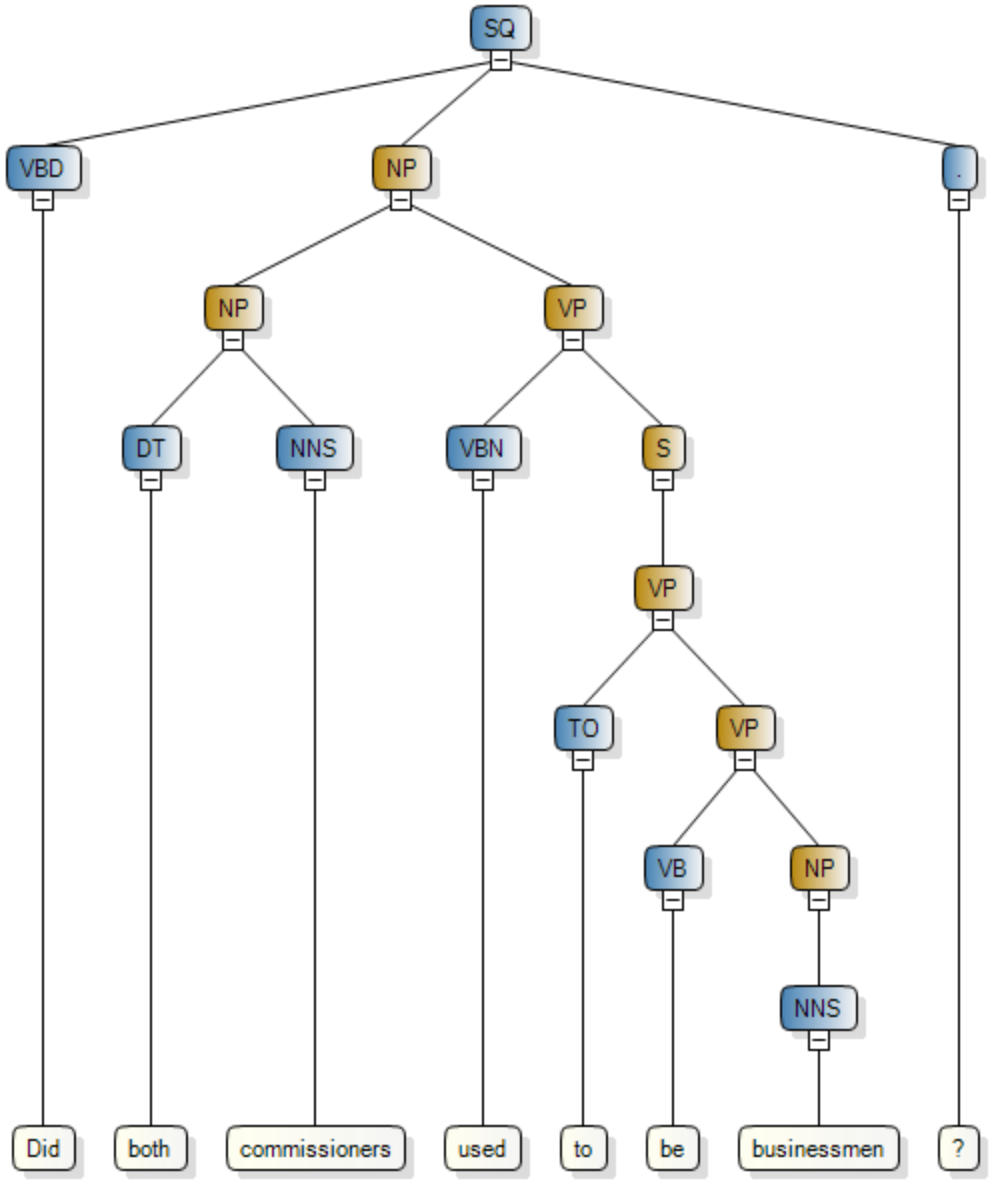
Table C-2. Problem 29 from Section 1.2 - FraCaS Test Suite

P1	Both commissioners used to be leading businessmen.
Q	Did both commissioners used to be businessmen?

Parses P1 (1 of 1)



Parses Q (1 of 1)



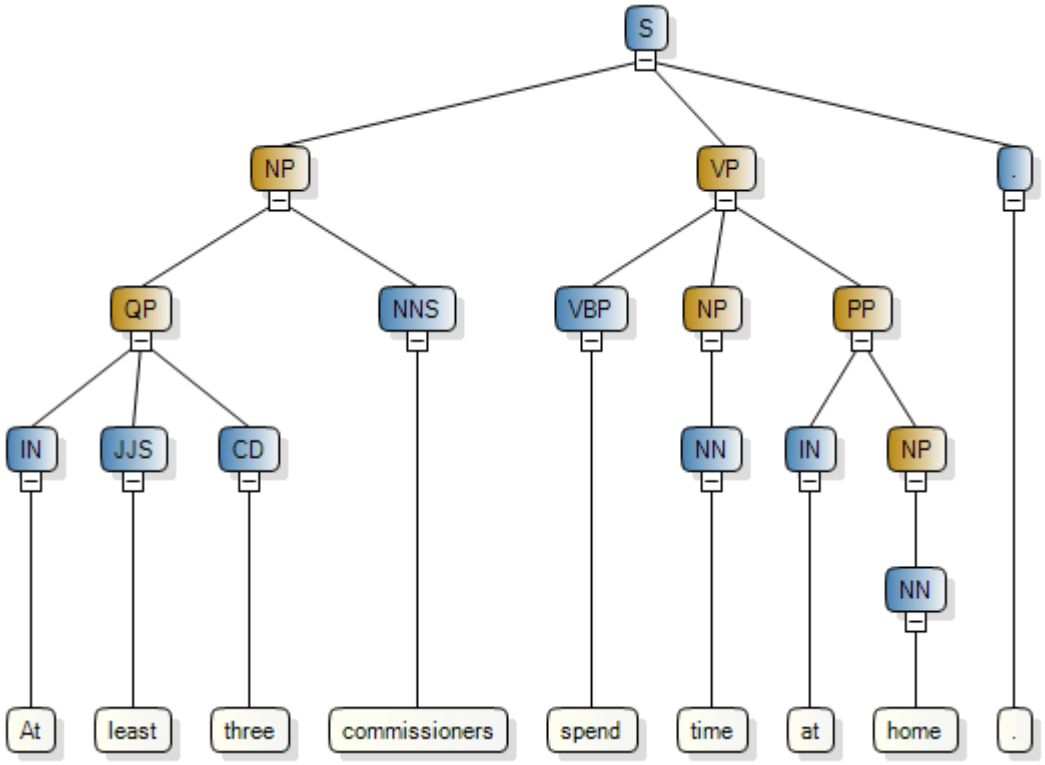
Section 1.3 - Example 3 – Problem 47 from FraCaS Test Suite

Problem 47 come from section 1.3 of the FraCaS Test Suite. This particular section focuses on Monotonicity (downwards on second argument). Here, we illustrate how the system handles this particular case.

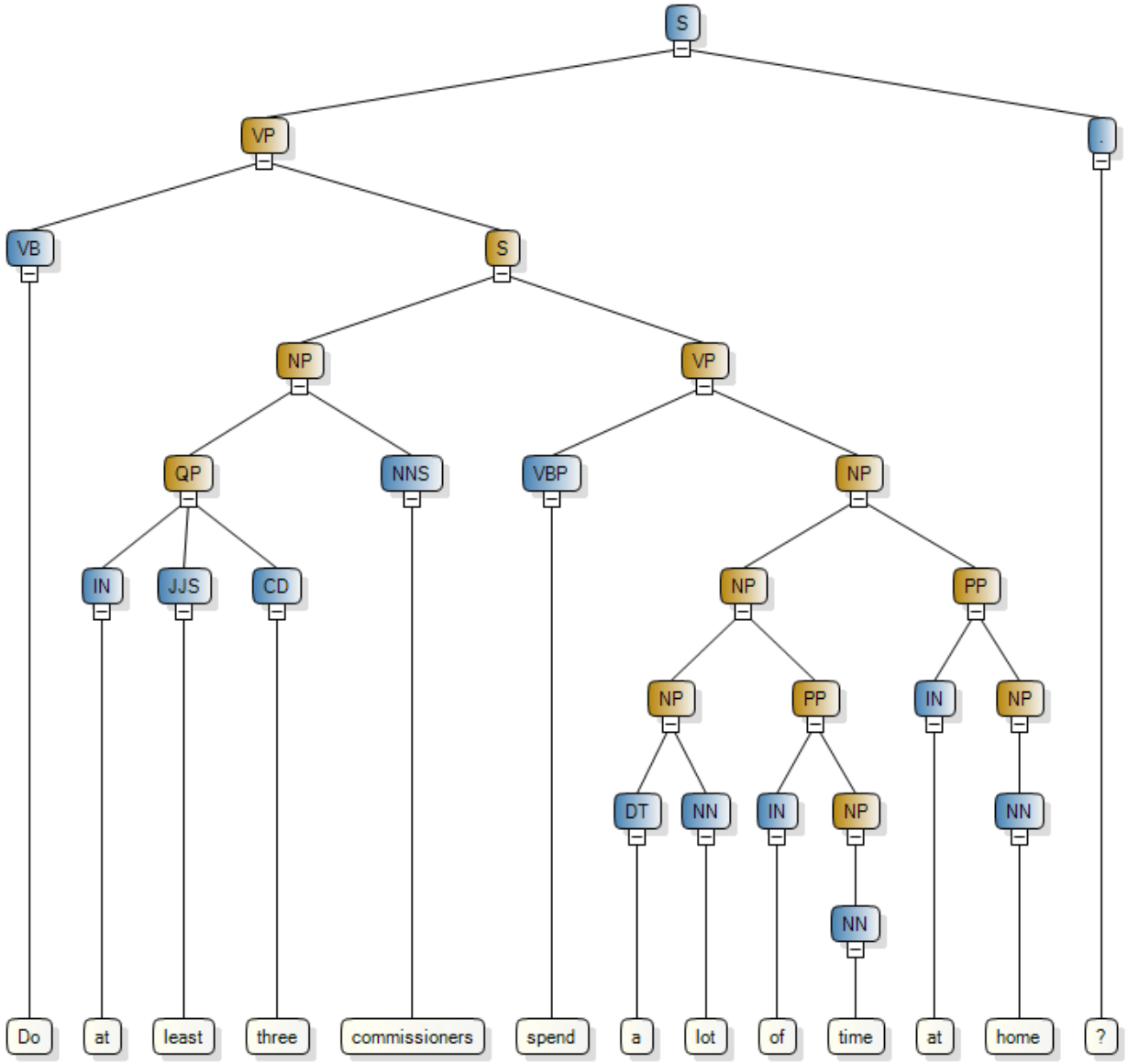
Table C-3. Problem 47 from Section 1.3 - FraCaS Test Suite

P1	At least three commissioners spend time at home.
Q	Do at least three commissioners spend a lot of time at home?

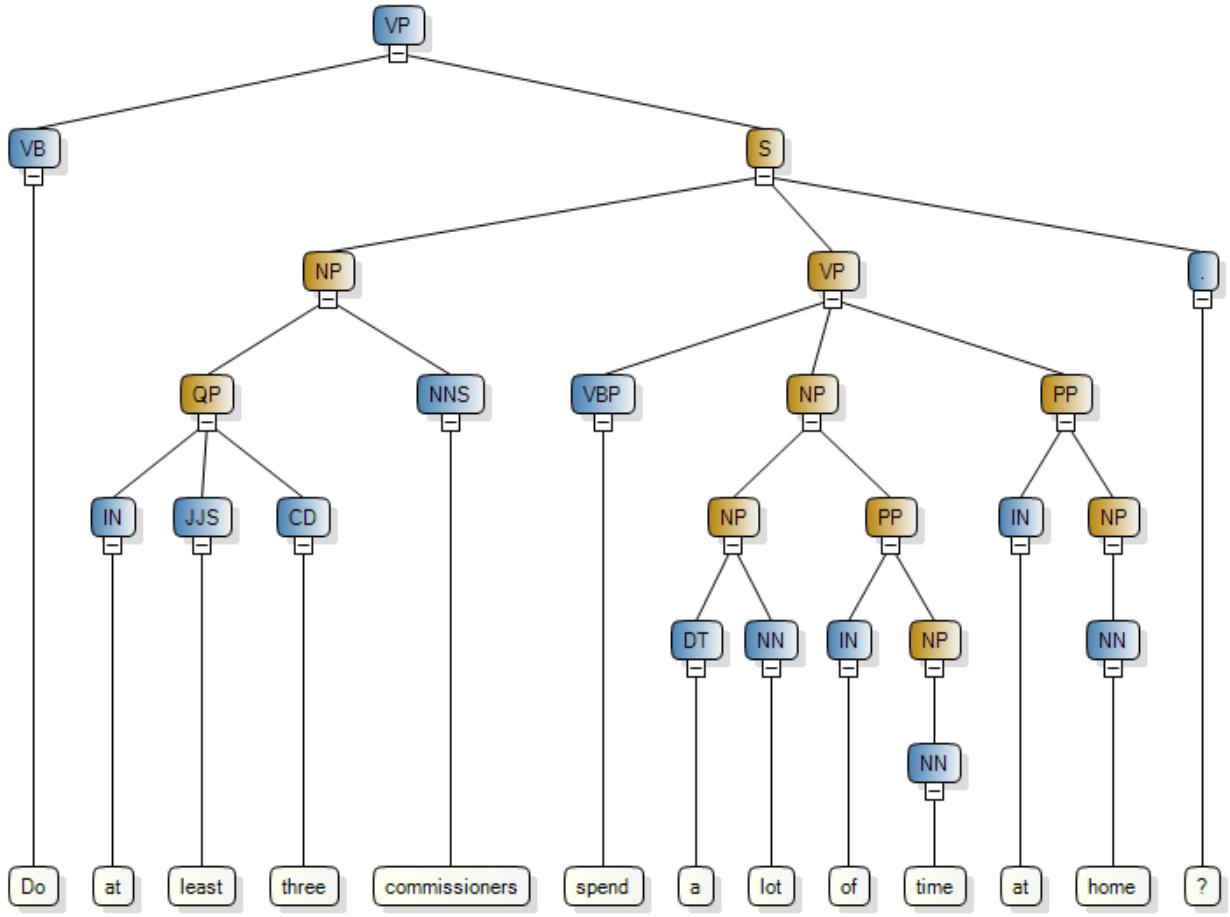
Parses P1 (1 of 1)



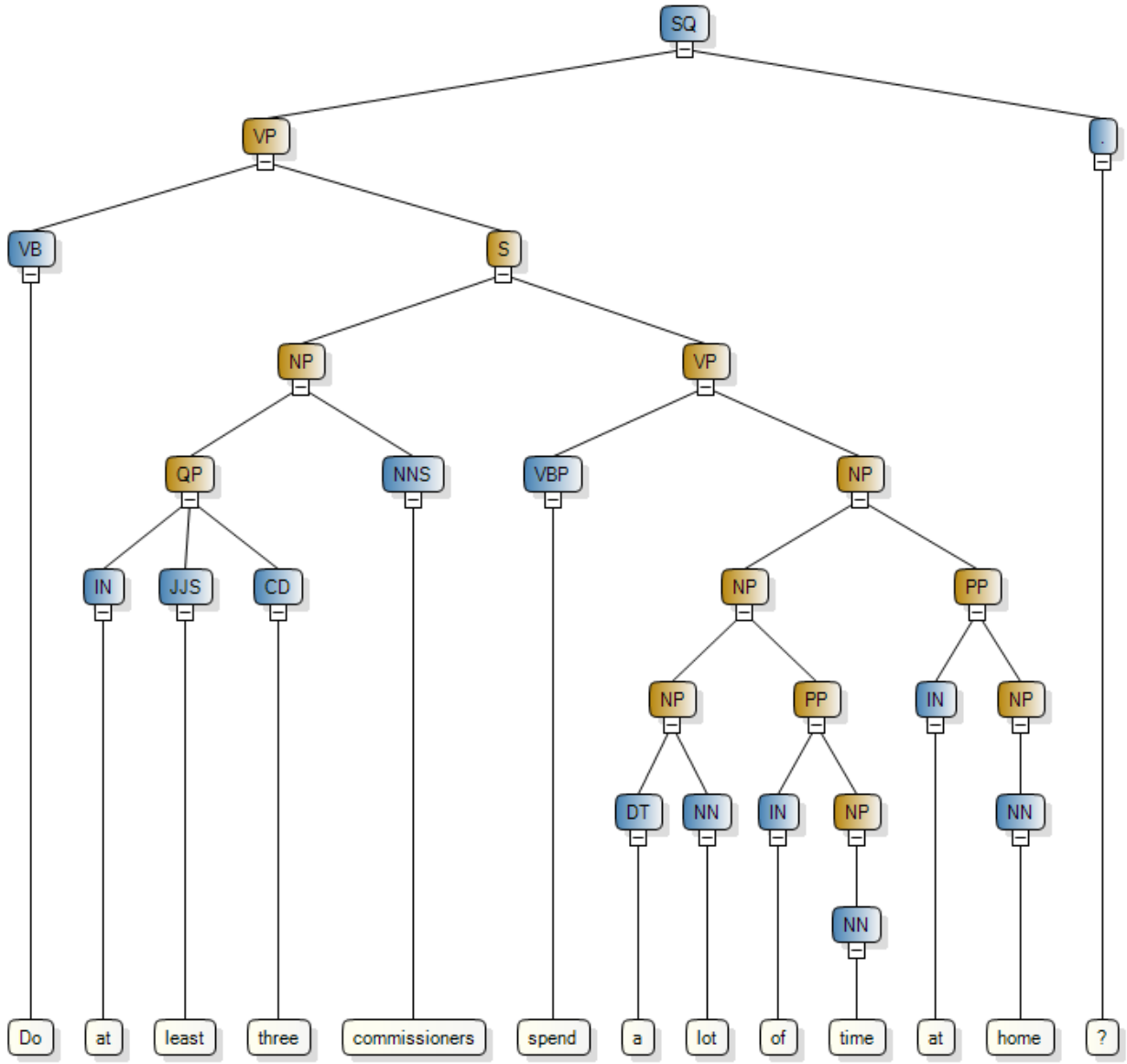
Parses Q (1 of 5)


S
VP
.
VB
S
NP
VP
QP
NNS
VBP
NP
IN
JJS
CD
NP
PP
NP
PP
IN
NP
DT
NN
IN
NP
NN
NN
Do
at
least
three
commissioners
spend
a
lot
of
time
at
home
?

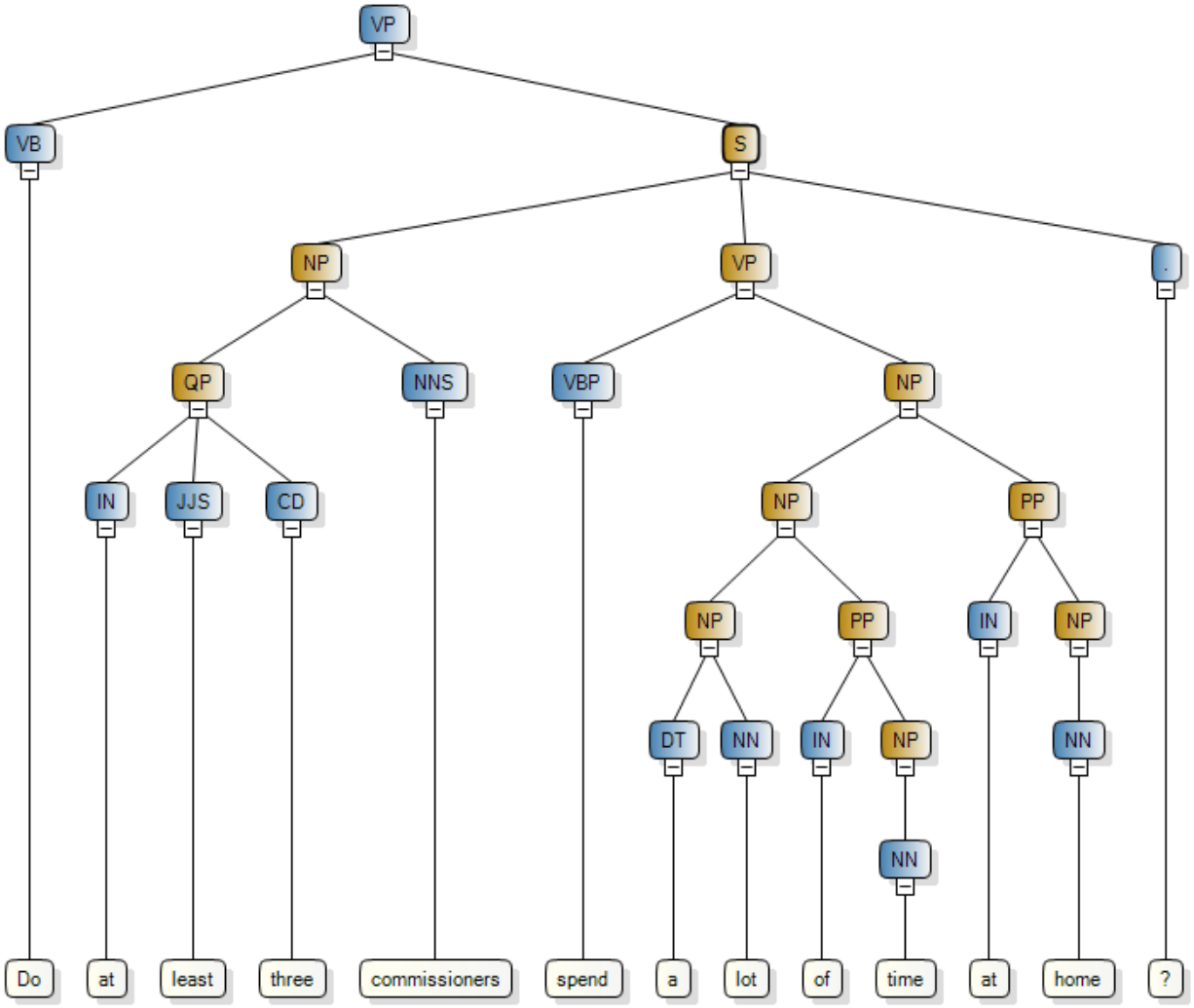
Parse Q (2 of 5)


VP
VB
S
NP
VP
.
QP
NNS
VBP
NP
PP
IN
JJS
CD
NP
PP
IN
NP
DT
NN
IN
NP
NN
NN
Do
at
least
three
commissioners
spend
a
lot
of
time
at
home
?

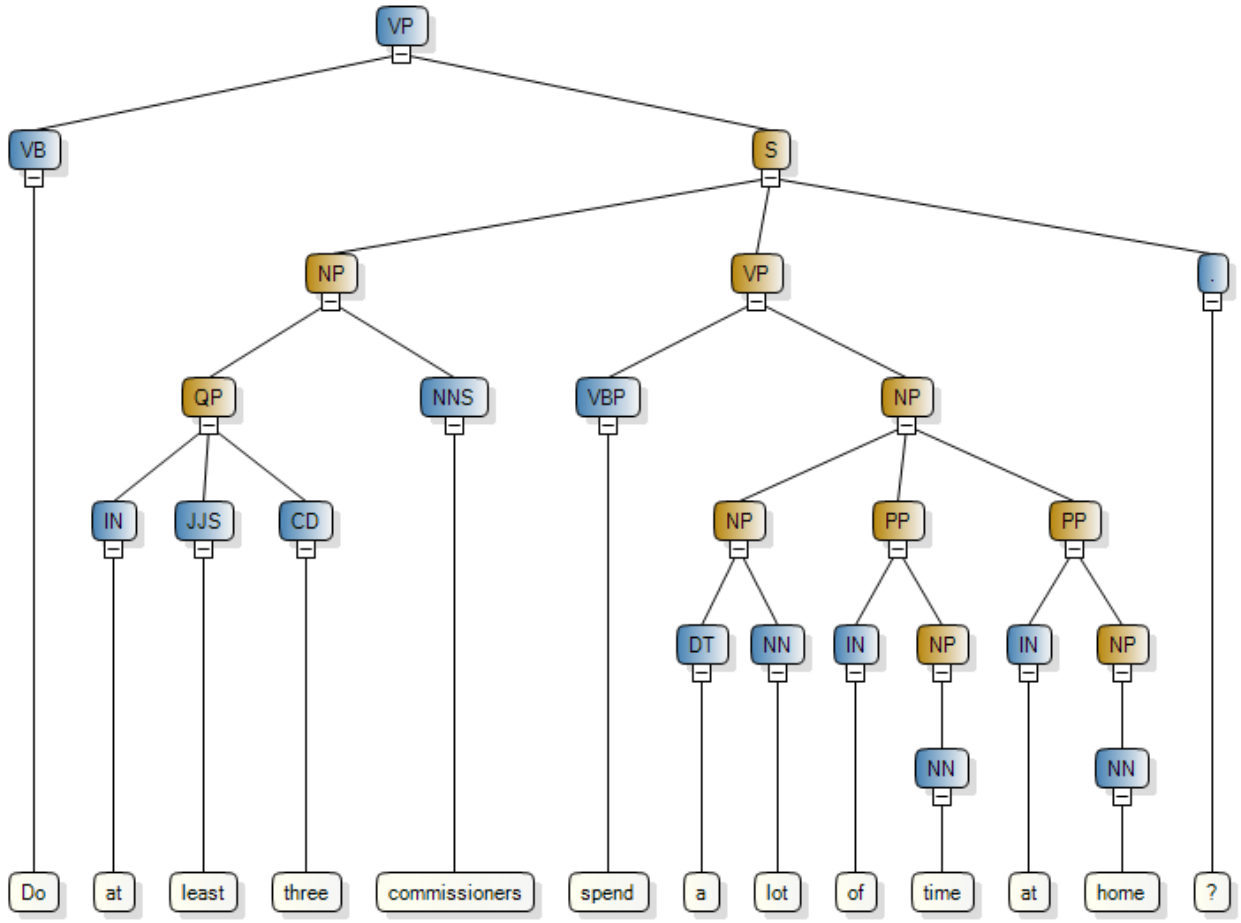
Parse Q (3 of 5)


SQ
VP
.
VB
S
NP
VP
QP
NNS
VBP
NP
IN
JJS
CD
NP
PP
NP
PP
IN
NP
DT
NN
IN
NP
NN
NN
Do
at
least
three
commissioners
spend
a
lot
of
time
at
home
?

Parse Q (4 of 5)


VP
VB
S
NP
VP
.
QP
NNS
VBP
NP
IN
JJS
CD
NP
PP
NP
PP
IN
NP
DT
NN
IN
NP
NN
NN
Do
at
least
three
commissioners
spend
a
lot
of
time
at
home
?

Parse Q (5 of 5)

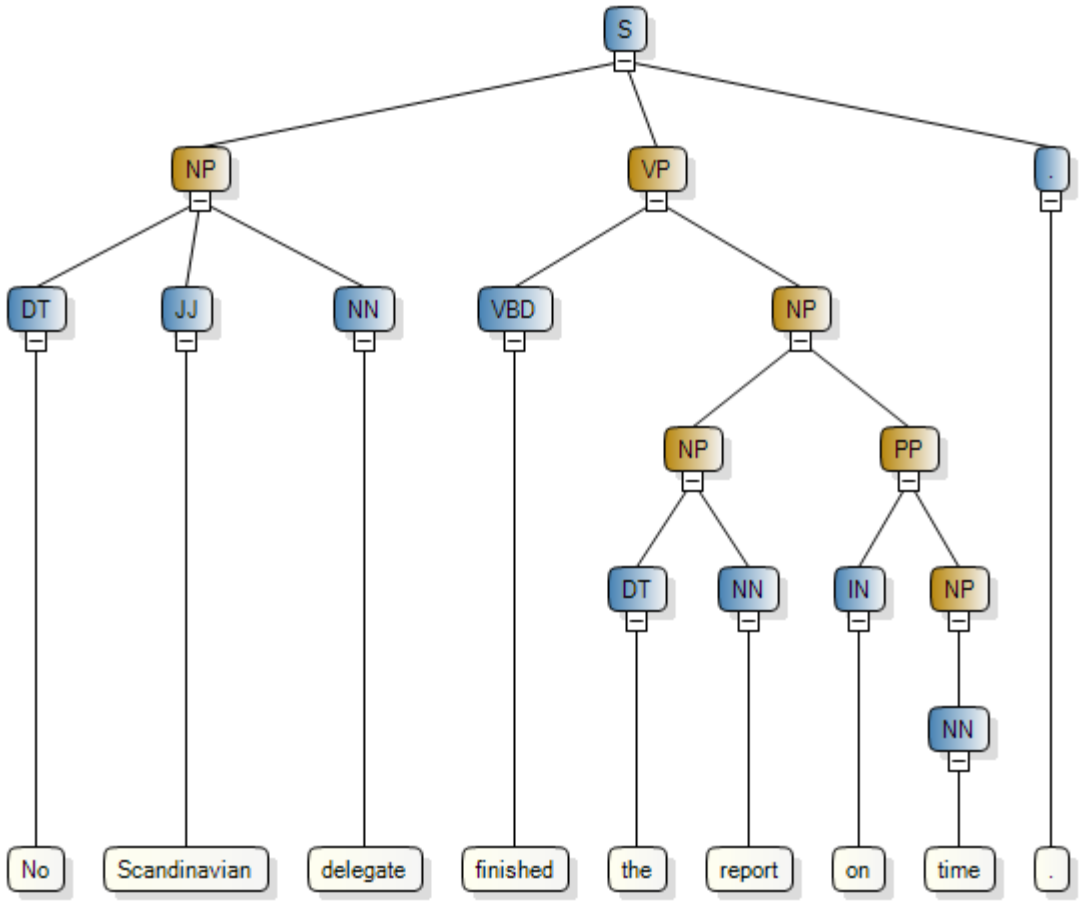


Section 1.4 - Example 4 – Problem 54 from FraCaS Test Suite

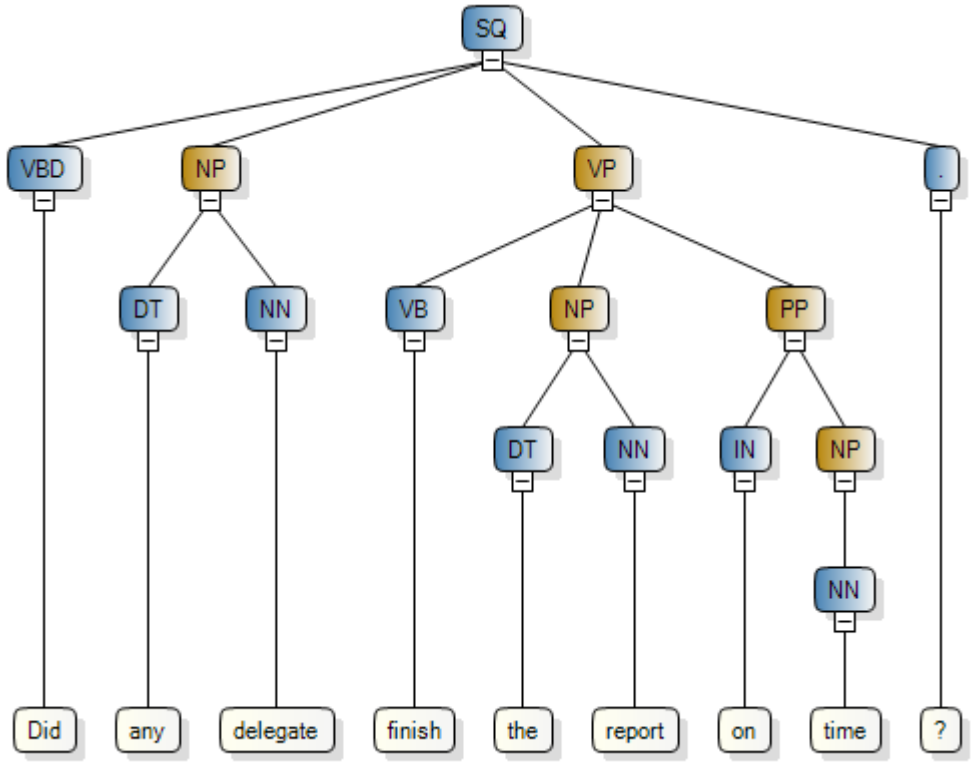
Problem 54 come from section 1.4 of the FraCaS Test Suite. This particular section focuses on Monotonicity (upwards on first argument). In this case, a lot of time is effectively a child of time. Here, we illustrate how the system handles this particular case.

Table C-4. Problem 54 from Section 1.4 - FraCaS Test Suite

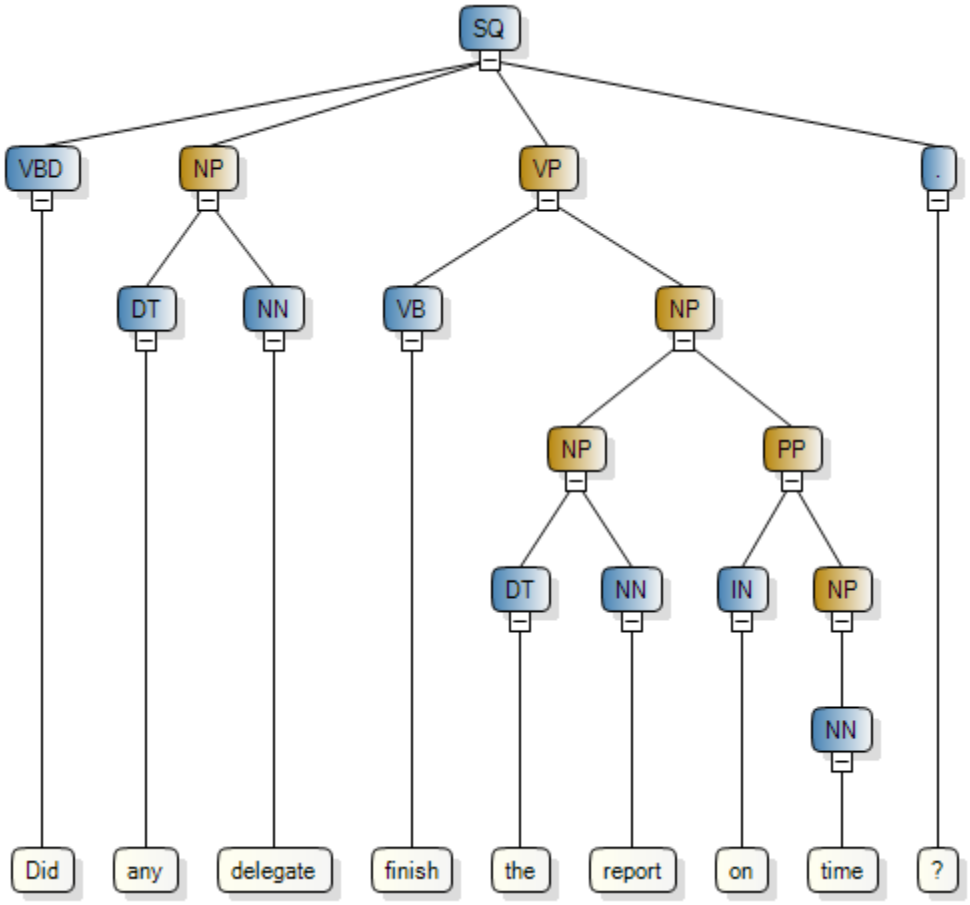
P1	No Scandinavian delegate finished the report on time.
Q	Did any delegate finish the report on time?



Parse P1 (1 of 1)


SQ
VBD
NP
VP
.
DT
NN
VB
NP
PP
DT
NN
IN
NP
NN
Did
any
delegate
finish
the
report
on
time
?

Parse Q (1 of 2)



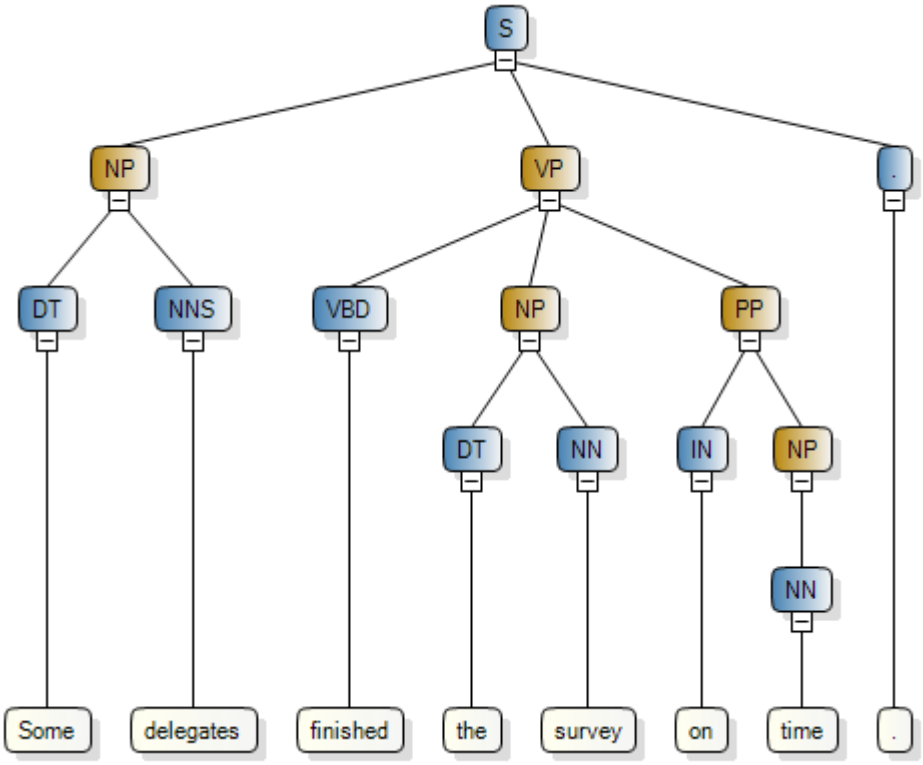
Parse Q (2 of 2)

Section 1.5 - Example 5 – Problem 71 from FraCaS Test Suite

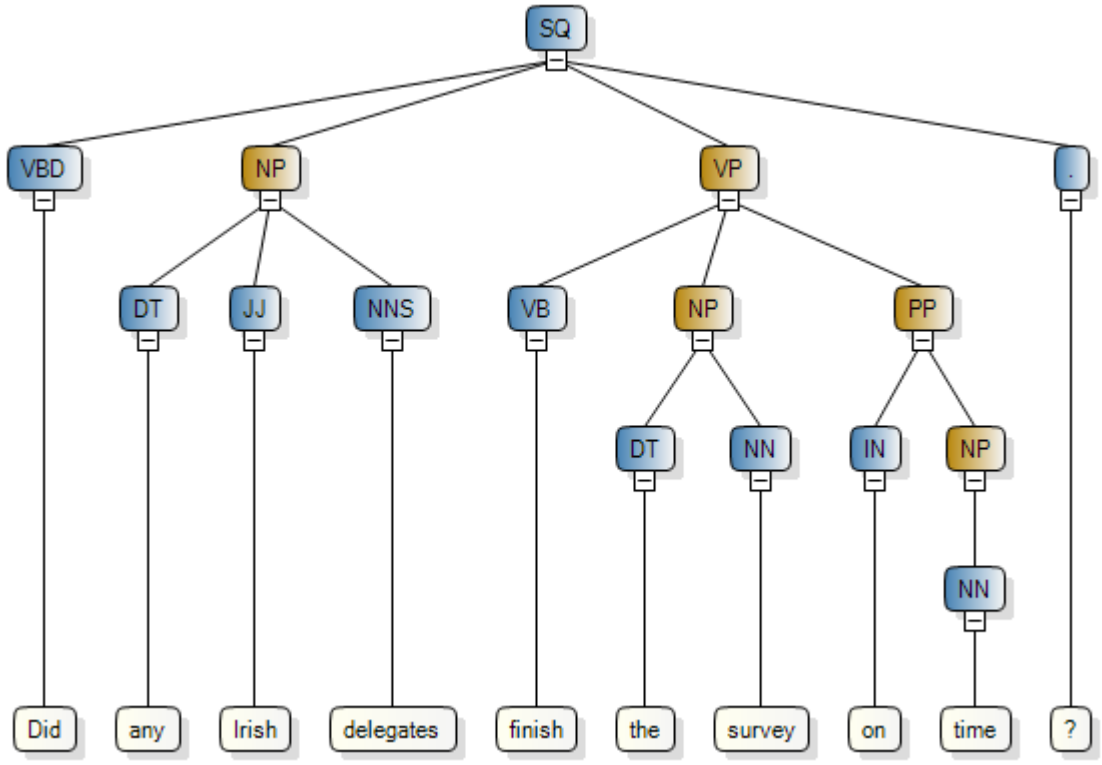
Problem 71 comes from section 1.5 of the FraCaS Test Suite. This particular section focuses on Monotonicity (downwards on first argument).

Table C-5. Problem 71 from Section 1.5 - FraCaS Test Suite

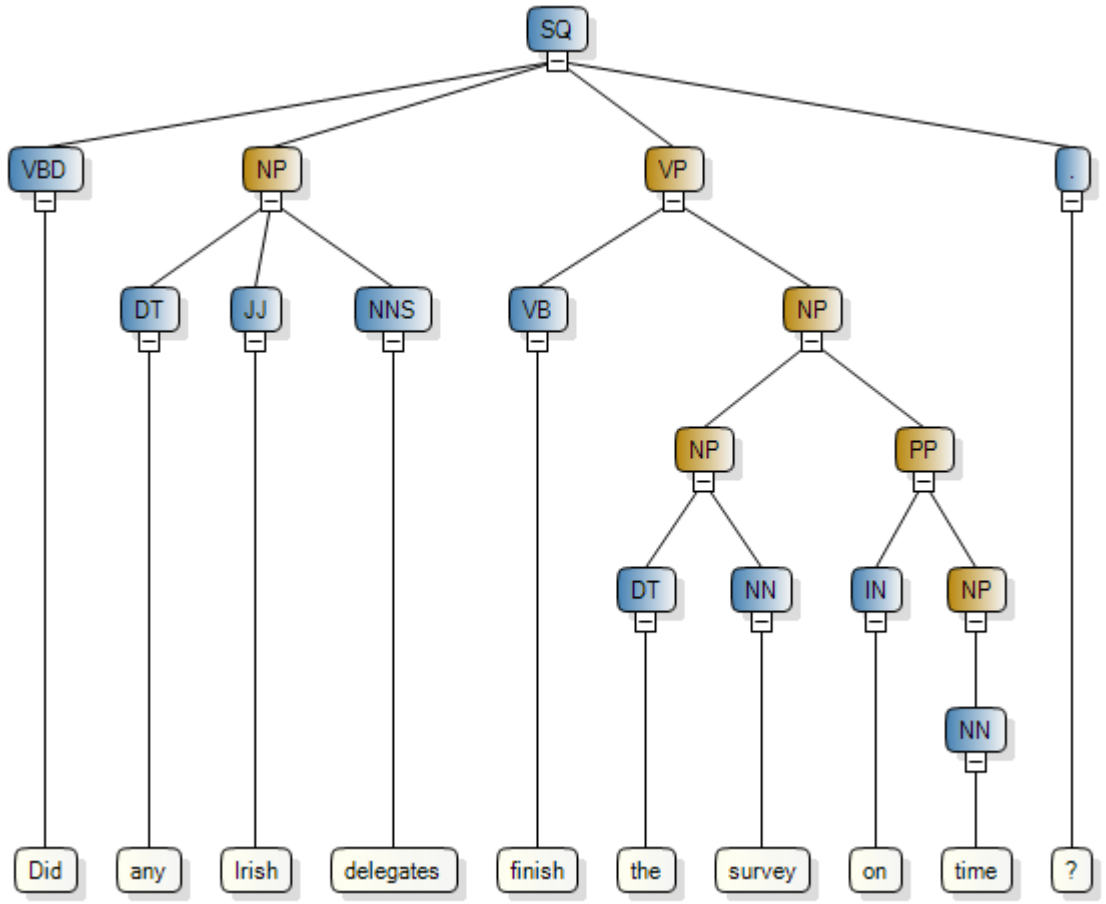
P1	Some delegates finished the survey on time.
Q	Did any Irish delegates finish the survey on time?



Parse P1 (1 of 1)



Parse Q (1 of 2)



Parse Q (2 of 2)

Appendix D - Plurals

Section 2.1 - Example 6 – Problem 81 from FraCaS Test Suite

Problem 81 comes from section 2.1 of the FraCaS Test Suite. This particular section focuses on Conjoined Noun Phrases.

Table D-1. Problem 81 from Section 2.1 - FraCaS Test Suite

P1	Smith, Jones and Anderson signed the contract.
Q	Did Jones sign the contract?

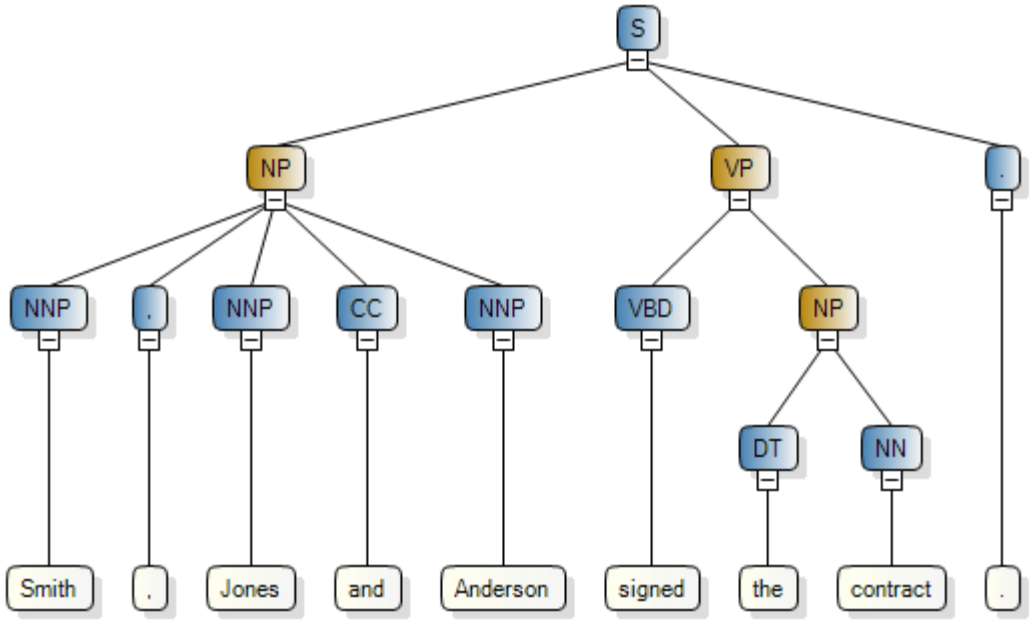


Figure D-1 Parse of Premise 1 (1 of 1) for Problem 81

First, we look at the S node in Figure D-1. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object.

Now looking at the [NP] node (Noun Phrase), it has five children, DT, ., NNP, CC, NNP. This form indicates that there is a list of nouns separated by some conjunction. First we look at NNP, we know that this particular noun is referring to a specific [NNP], as there is no article indicating a generic reference be created, which is important for matching purposes later on. Finally, we finish this noun object by placing Smith as the name for this particular object. Observe that there is no article modifying the NNP and we know that the NNP stands for a singular Proper Noun; we know the quantity of this particular object is 1. Since there are no other properties or attributes to attach to the Object Smith at this point we proceed with the processing the next term in the noun list from the [NP]. Next, we have the [.] which is just a quick check to make sure it is a comma to ensure we actually have a comma separated list of terms, in this case we do. In the event we had something other than a comma, we could, given the situation throw a bad parse and discard the parse tree. Next, we have [NNP] which is for Jones and created similarly as Smith. Next we have [CC] which stands for Coordinating Conjunction. This term, only has 1 child which is 'and'. So we return this new and conjunction to the noun list to tell it how the rest of the sentence can affect the set that is formed from the noun list. At this point, we have placed into the temporary ontology an instance of a set that contains a reference to each of the follow: Smith, Jones and Anderson combined with an and conjunction where the quantity is only 1 and that it contains no attributes at this point in time. Next, we look at the [VP] node (Verb Phrase). The [VP] node contains two children, both a [VBD] node (past tense), and a [NP] node. We look at the verb 'signed' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a past tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurred at some point in the past we can say that it occurred at time $t < 0$.

We now have the final noun object for this verb phrase to process. The node [NP] has two children [DT] and [NN]. The [DT] contains 'the', and [NN] contains 'contract'. We know that given the [DT] 'the' we are referring to a specific item, where the quantity of said item is set

to 1. (This is known from the fact that [NN] is a singular noun). The system tries to find an exact match in the main ontology first since it could exist already. Since it does not find a match, it creates an instance of contract. We add this instance of contract to the temp ontology as well.

Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of Italian to the instance of contract via the word 'signed'. At this point, it doesn't matter when constructing the temp ontology what definition 'signed' is referencing, so long as it has valid parameters so as to not reject the statements parse outright. During the inference stage is when the definition is chosen and any additional inferences are inferred at this point.

The system now takes a look at everything derived from the first premise as a whole and begins to build the temp ontology shown in Figure D-2. It looks at every predicate and begins to assimilate the data. The only predicate (of interest) in this particular premise is the 'signed' predicate. There is no additional information or facts generated from the use of the word 'signed'. A dictionary look up of the word 'signed', there could be up to 7 possibilities, none of which provide any gain for this particular test suite. Since signed is working with a left hand side of a noun list, we apply the verb to the set itself, and to each item in the set connected by the and conjunction in this case.

SIGNED< < t + 0 >(Instance ID: 1 (Smith), Instance ID: 4 (contract)) and SIGNED< < t + 0 >(Instance ID: 2 (Jones), Instance ID: 4 (contract)) and SIGNED< < t + 0 >(Instance ID: 3 (Anderson), Instance ID: 4 (contract))
SIGNED< < t + 0 >(SET_1, Instance ID: 4 (contract))

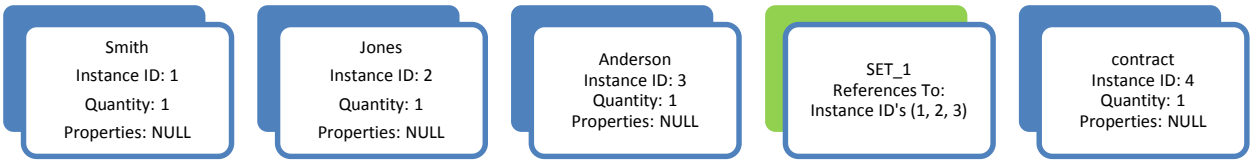


Figure D-2 Premise 1: Temporary Ontology

Now that all predicates within the first premise have been processed, we merge this temp ontology into the main ontology. This is a trivial process for this particular case as there currently exists nothing in the main ontology. The system simply copies every instance/class/relation over.

Once the ontologies have been merged, the main ontology is analyzed to look for new relationships that can be constructed by this new knowledge. Since an instance can only be matched to another instance if it matches a parent class, aka they are of the same concept, and we must construct a parent class for any instance that currently has no parent class(es). For this premise, we have four instances 'Smith', 'Jones', 'Anderson', and 'contract' and none of them have a parent class. We construct a parent class for each of these maintaining the exact properties as seen in them. Which in this case, no instance has any properties and note that the quantity is 1, so we create a class 'Smith' that has no properties and has a quantity of 1. We make this class the parent of the instance version of 'Smith'. We also must check to see if we can create or associate with a class 'Smith' with quantity ALL, that represents all Smiths that could exist. Since it originally had none, we know to create class Smith with quantity all (Class ID: 1) and insert a parent child relationship between the Class ID: 1 and class ID: 2. If said class exists we just provide add the parent child relationship between quantity 1 of class Smith (Class ID: 2). We do the same for instance 'Jones', 'Anderson', and 'contract' and we end up with what we see in Figure D-3. We take a look at each relation within the ontology to see if there are

any new facts that can be inferred. We first note that we can immediately rule out any verb phrases involving 'signed'. We are left with some proper noun(s), in the past tense act 'signed', and an instance of an object. This means that we can infer that that the proper noun(s) are also of type person. We also know that is possible for any person (represented by class Person) can sign a contract.

SIGNED< < t + 0 >(Instance ID: 1 (Smith), Instance ID: 4 (contract)) and SIGNED< < t + 0 >(Instance ID: 2 (Jones), Instance ID: 4 (contract)) and SIGNED< < t + 0 >(Instance ID: 3 (Anderson), Instance ID: 4 (contract))

SIGNED< < t + 0 >(SET_1, Instance ID: 4 (contract))

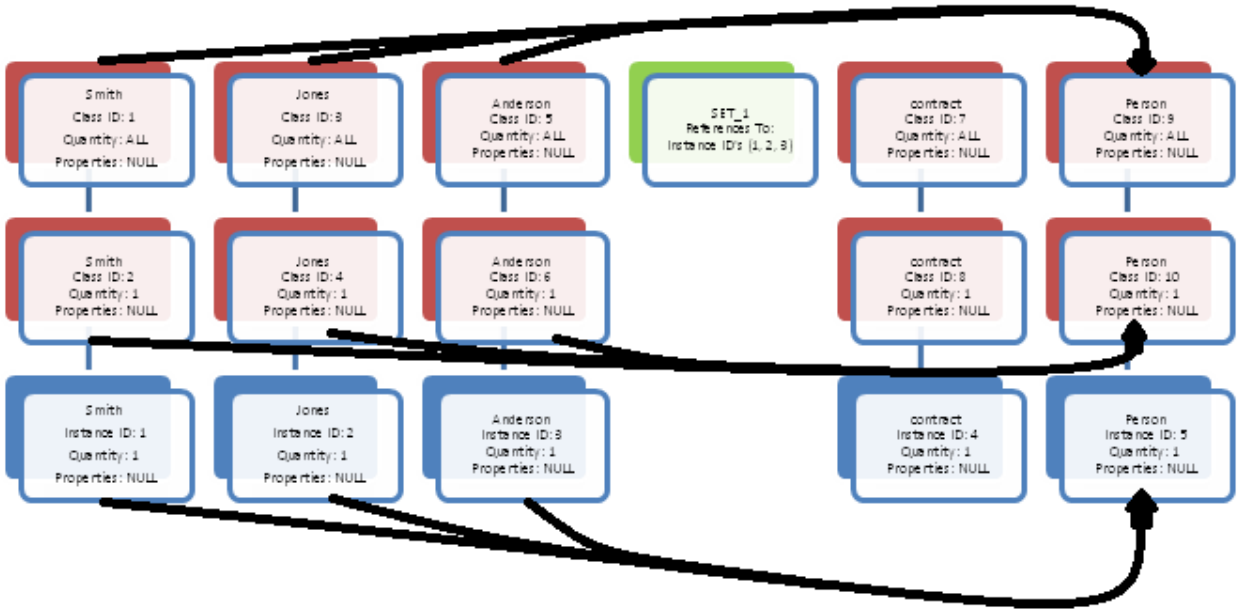


Figure D-3 Updated Ontology for Premise 1

The question is the final input to analyze. The question has two parses shown in Figure D-4 and Figure D-5. Since there is more than one parse, the system must go through each parse until it finds a valid parse. There is no problem with the first parse shown in Figure D-4.

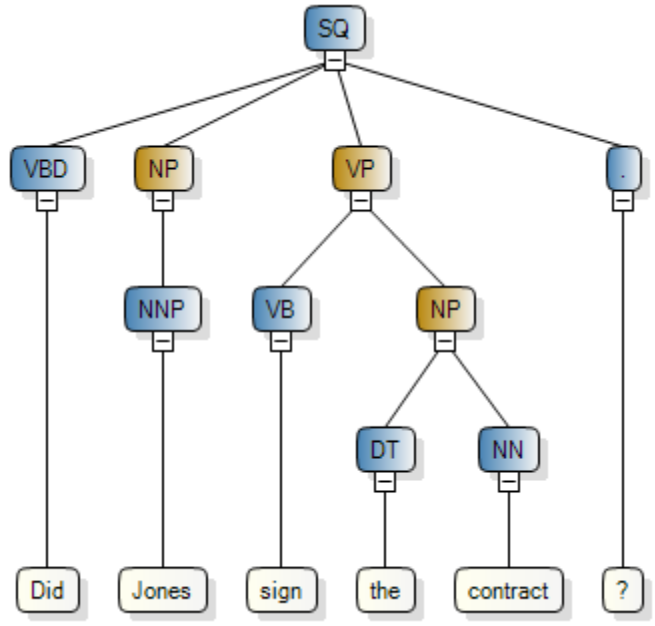


Figure D-4 Parse Q (1 of 2) for Problem 81

Since the system found a valid parse already for this problem it would skip the parse in Figure D-5. Had this one come before the correct parse, it would have found that it would have not been able to interpret this structure due to sign being a noun and there being a lack of a VP aside from the initial word in the sentence.

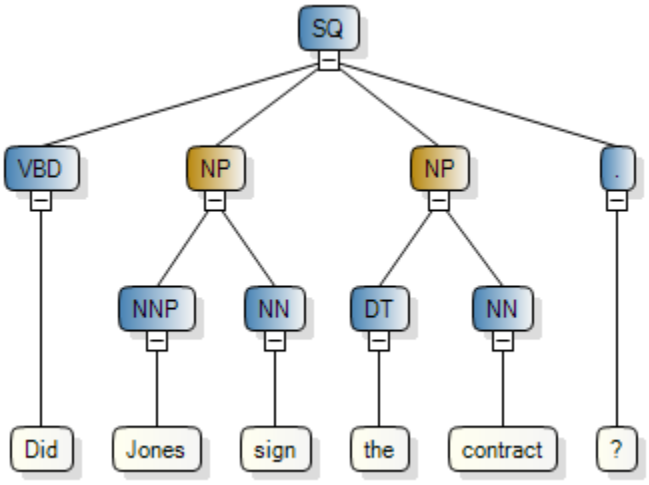


Figure D-5 Parse Q (2 of 2) for Problem 81

Using the parse found in Figure D-4 we need to process it to see if produces a valid output. It is still possible that the system could kick back that this parse is actually invalid through analyzing components and how they relate. We start with the root node as always, the [SQ] (Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ), and we must determine which type it is. To do that, we look at the children. It has four children [VBD], [NP], [VP], and [.]. Again, seeing these 4 nodes we see that the type of the question is a yes/no question. We look at the children from left to right, except at the root level node where we look at the [.] to confirm it is a question for a simple rejection test. Looking at [VBD], says that we have a past tense verb and we load this verb into the yes/no question. Next, we look at [NP]. From this we know that we are going to build some noun object. We must look at the children to see what properties we can learn just from this information alone. We see that it only contains one node [NNP] which is a singular proper noun. The singular proper noun is 'Jones'. The [NNP] is added to the new noun object that has been created and an instance object is created 'Jones'. As you can see there are no other properties of the noun object and we know that 'Jones' is singular from [NNP] so the instance object contains a quantity of 1, and no attributes associated with it. At this point we don't know any parent information about this instance 'Jones'. Upon completion of the [NP], the system looks at the node [VP]. The node [VP] under [SQ], contains two children, [VB] (base form of a verb), and another [VP]. The [VB] loads up the base form of the verb 'sign' and stores that into the predicate. The predicate also contains the object, which comes from the [NP], which itself has two children, both a [DT] and [NN]. So given that [NP], we know must create a noun object for the predicate, and we populate it with a [DT], which yields 'the' and add that as a field into the noun object. Next we look at the [NN], which contains 'contract' a singular noun which is also loaded into the noun object. No other nodes are left for the noun object. So at this point we can determine that the combination of the [DT] 'the', and the [NN] that a specific contract, that should already exist that is being referred to. So we check to see if we can find a contract that exists in the ontology that matches all the properties (name, quantity, attributes). If we find a match, then we use that object, which helps us to build a quicker understanding of what is being referred to. A match is found by looking through all objects in reverse order of being referenced. If no match is found, then we simply instantiate a new instance object 'contract' that contains the quantity of 1 and no attributes associated with it. In this case, we did find a pre-existing instance 'contract' that

matches on all counts. Now that the [NP] is finished, the noun object is loaded into the predicate. The[VP] is finished at this point, then this is loaded into the yes/no question. At this point, the system goes back through and tries to build more relationships. First, it combines the past tense 'did' with the verb base sign to make 'sign' past tense in terms of time. That is to say, since this is the second input the system has received, the time is set to 1, so to say it occurred before time 1, the system interprets it as '< t + 1'. It then builds the relationship sign< < t + 1 >(Instance ID: 5 'Jones', Instance ID: 6 'contract'). The information that is generated into the temporary ontology is shown in Figure D-6. Notice that Instance ID: 6 'contract' is not shown, as it exists in the main ontology.

sign< < t + 1>(Instance ID: 5 'Jones', Instance ID: 6 'contract')

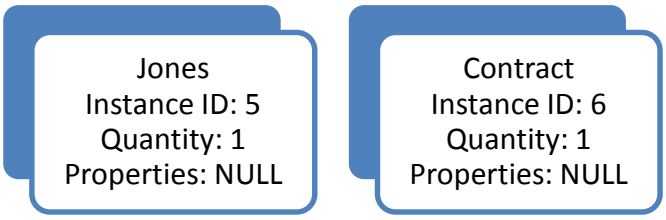


Figure D-6 Temporary Ontology for Question for Problem 81

To answer this particular question, we see that there is no missing elements or unknowns to fill in; we must first match each particular instance with everything in the ontology to see which items are potential candidates for matching the particular constraints of the temporary ontology (aka the question). For every potential candidate, we check against all other potential candidates, (another expensive operation) to see if a particular relationship is satisfied. Remember, that only information that is gained or useful is added to the ontology. This typically means, little to no information will be added to the ontology when asking questions.
First, the system finds potential candidates for 'Jones' (Instance ID: 5). There is only one potential candidate for 'Jones' which is (Instance ID: 2). We do the same for 'Contract' (Instance ID: 6) and the system yields (Instance ID: 4).

Next, we must see if the parameters for 'sign' are satisfiable with what exists in the ontology. The only combination is valid as it is found in the ontology. Finally we check to see for this combination of signed if the time is also valid (looking back in order starting from most

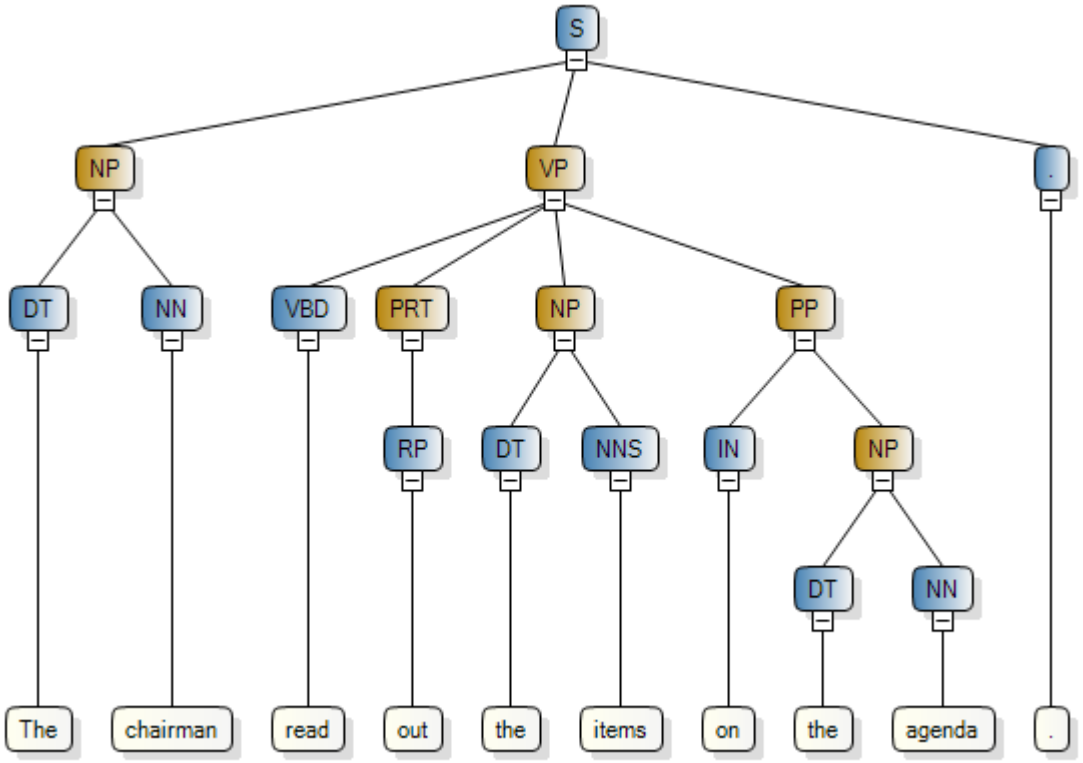
recent). Since it is last known that 'Jones' signed the 'contract', we can answer yes to this question.

Section 2.2 - Example 7 – Problem 90 from FraCaS Test Suite

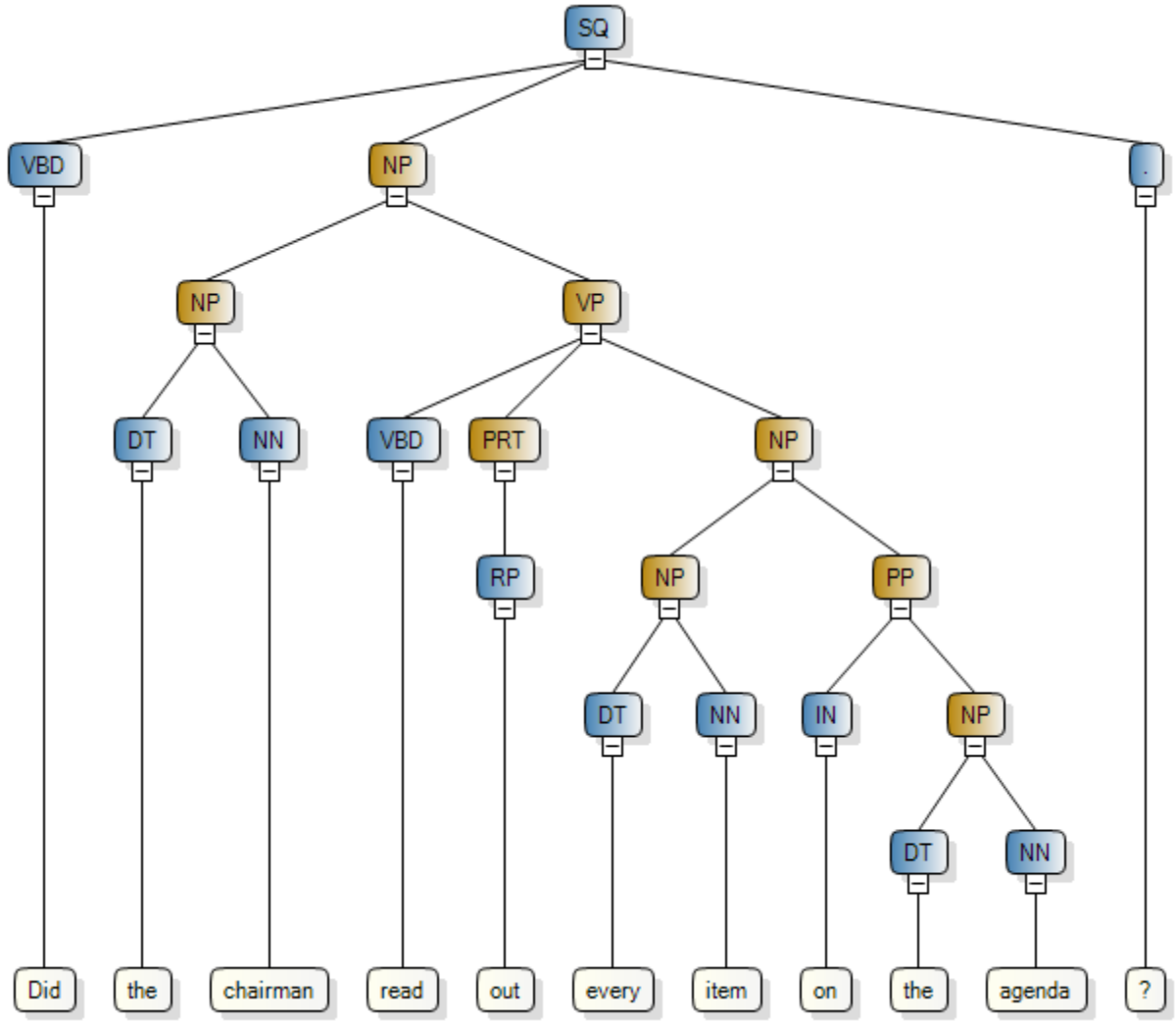
Problem 90 comes from section 2.2 of the FraCaS Test Suite. This particular section focuses on Definite Plurals. This problem is actually the only problem with a valid parse from the FraCaS Test Suite for my system.

Table D-2. Problem 90 from Section 2.2 - FraCaS Test Suite

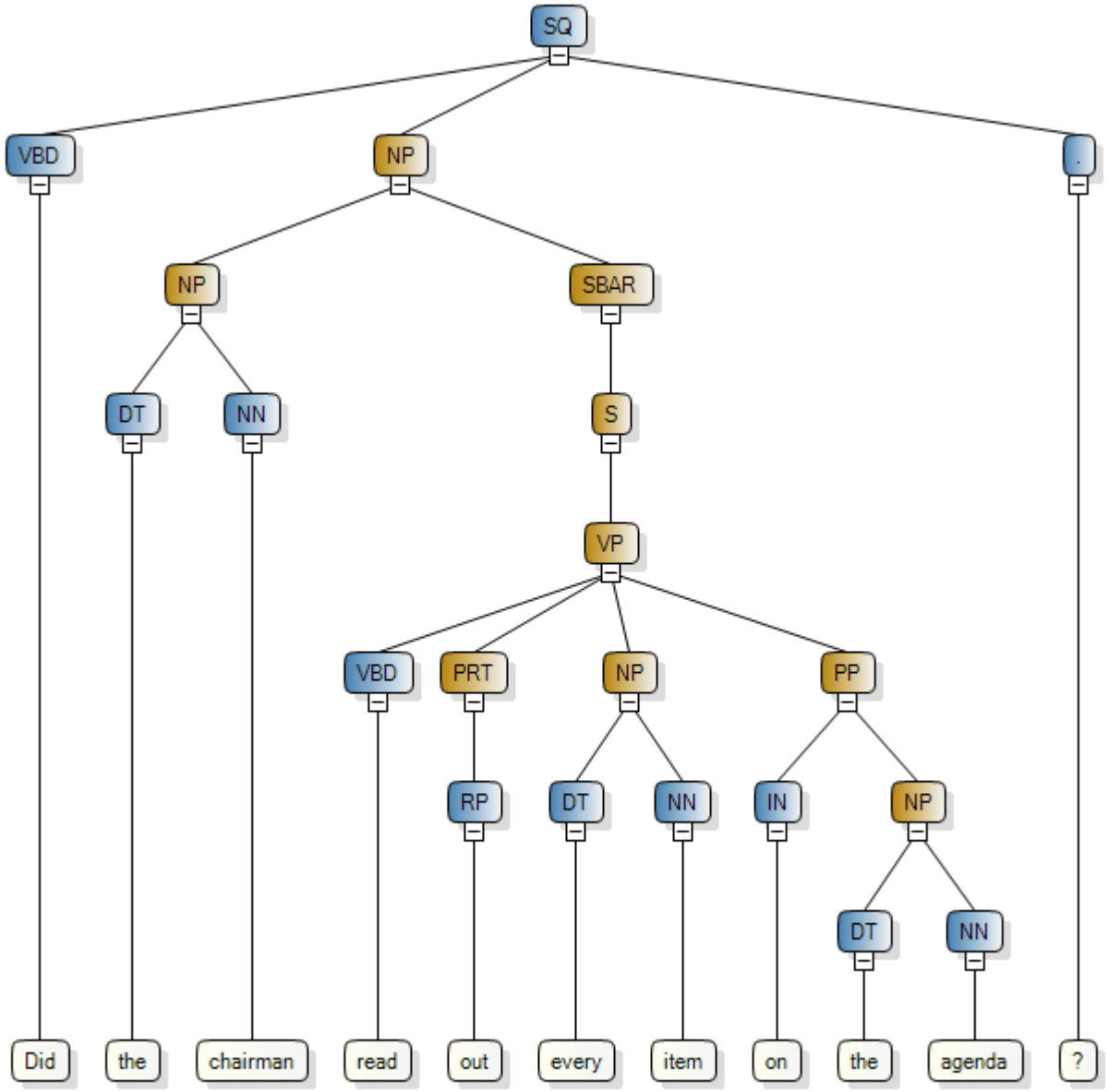
P1	The chairman read out the items on the agenda.
Q	Did the chairman read out every item on the agenda?



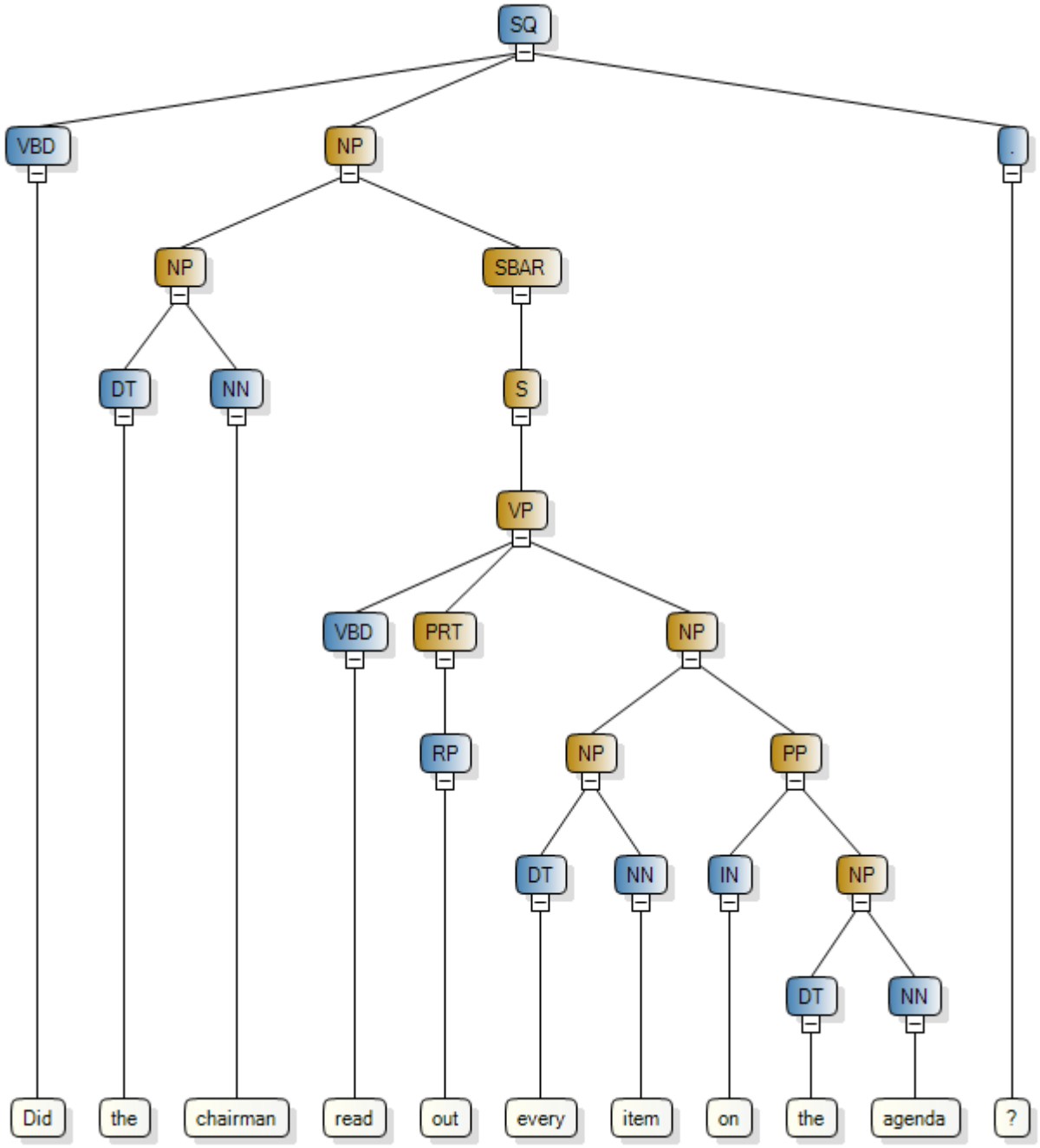
Parse P1 (1 of 1)


SQ
VBD
NP
.
NP
VP
DT
NN
VBD
PRT
NP
RP
NP
PP
DT
NN
IN
NP
DT
NN
Did
the
chairman
read
out
every
item
on
the
agenda
?

Parse Q (1 of 3)


SQ
VBD
NP
.
NP
SBAR
DT
NN
S
VP
VBD
PRT
NP
PP
RP
DT
NN
IN
NP
DT
NN
Did
the
chairman
read
out
every
item
on
the
agenda
?

Parse Q (2 of 3)



Parse Q (3 of 3)

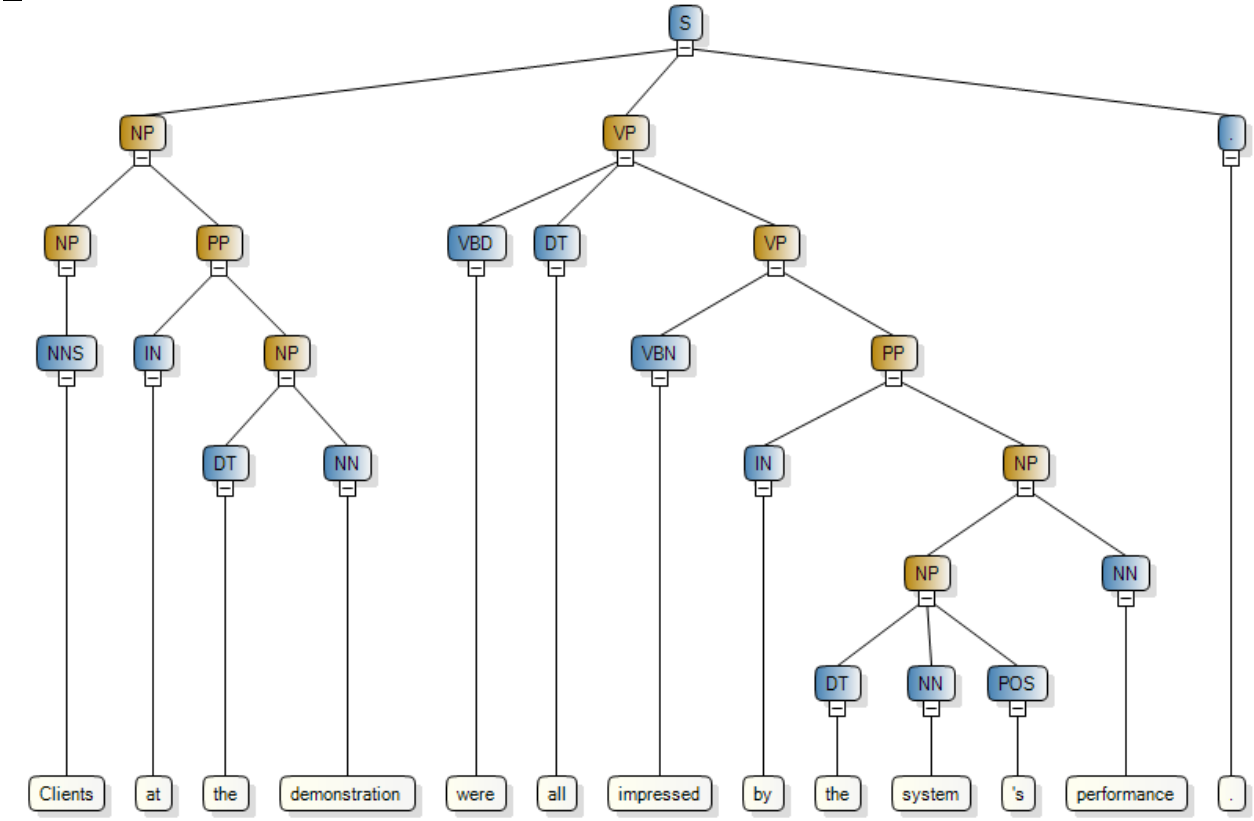
There is actually one more parse which is identical to the 3rd parse of the question that has been omitted.

Section 2.3 - Example 8 – Problem 99 from FraCaS Test Suite

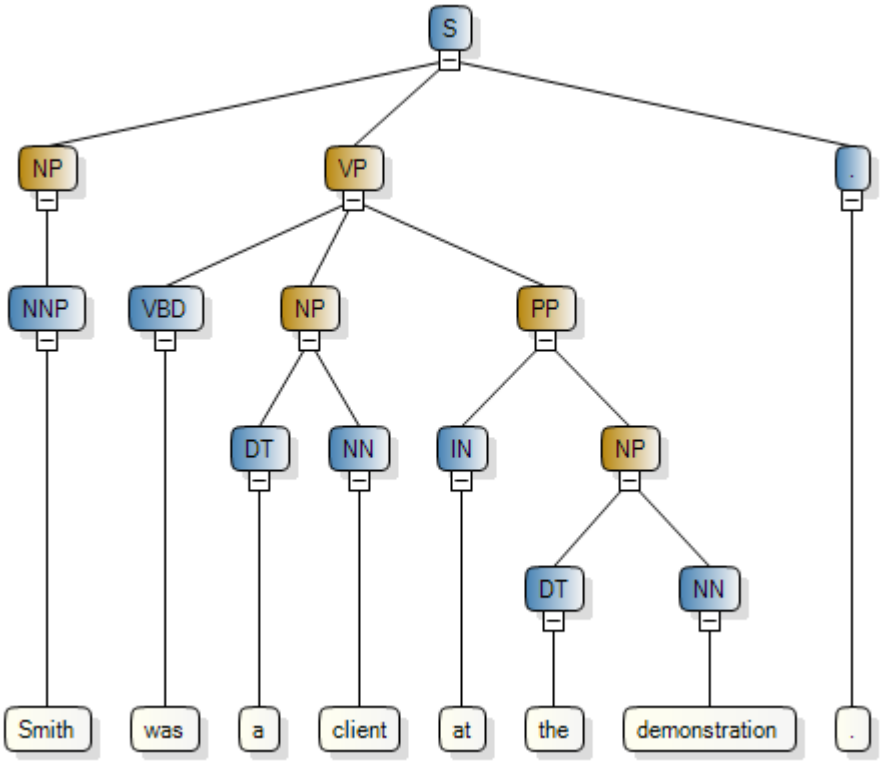
Problem 99 comes from section 2.3 of the FraCaS Test Suite. This particular section focuses on Bare Plurals.

Table D-3. Problem 99 from Section 2.3 - FraCaS Test Suite

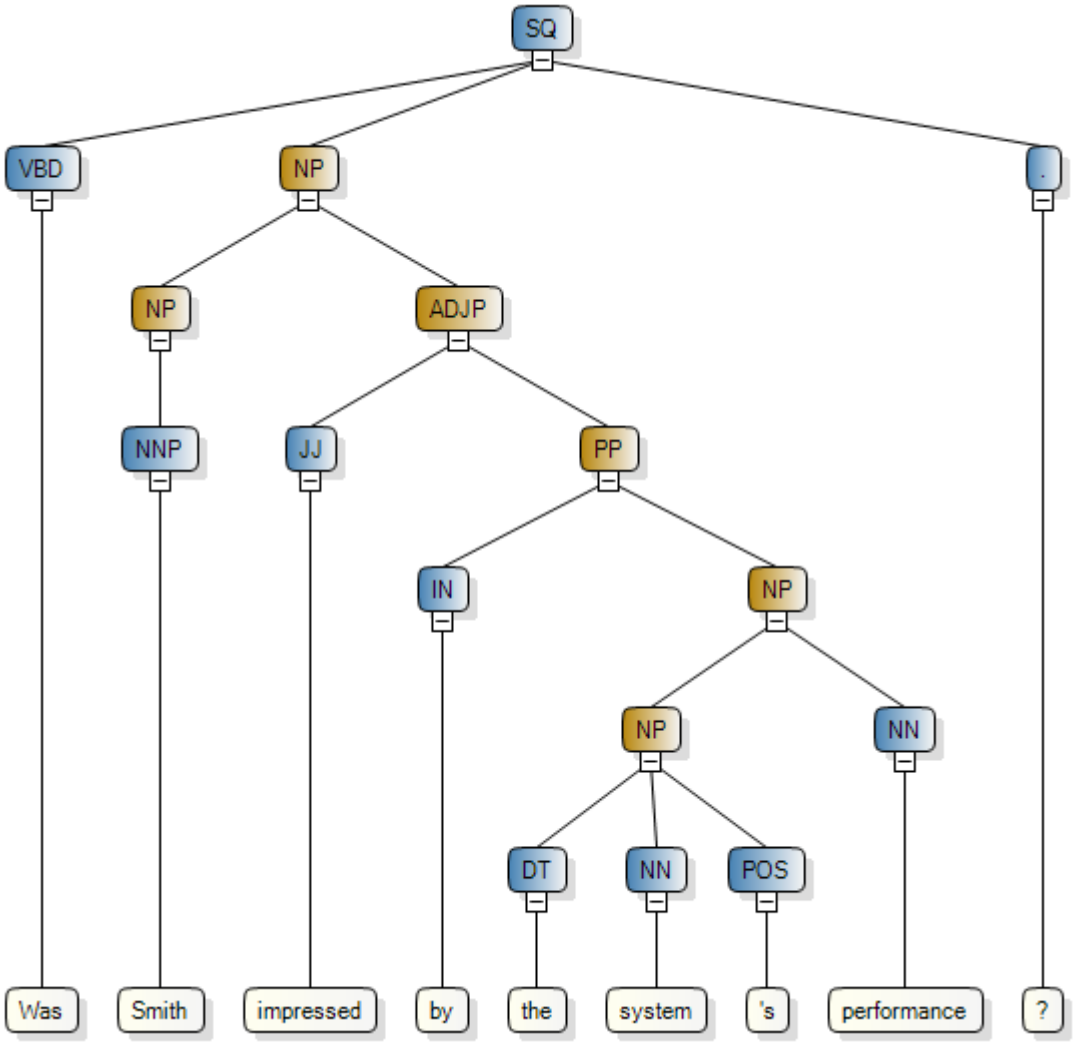
P1	Clients at the demonstration were all impressed by the system's performance.
P2	Smith was a client at the demonstration.
Q	Was Smith impressed by the system's performance?



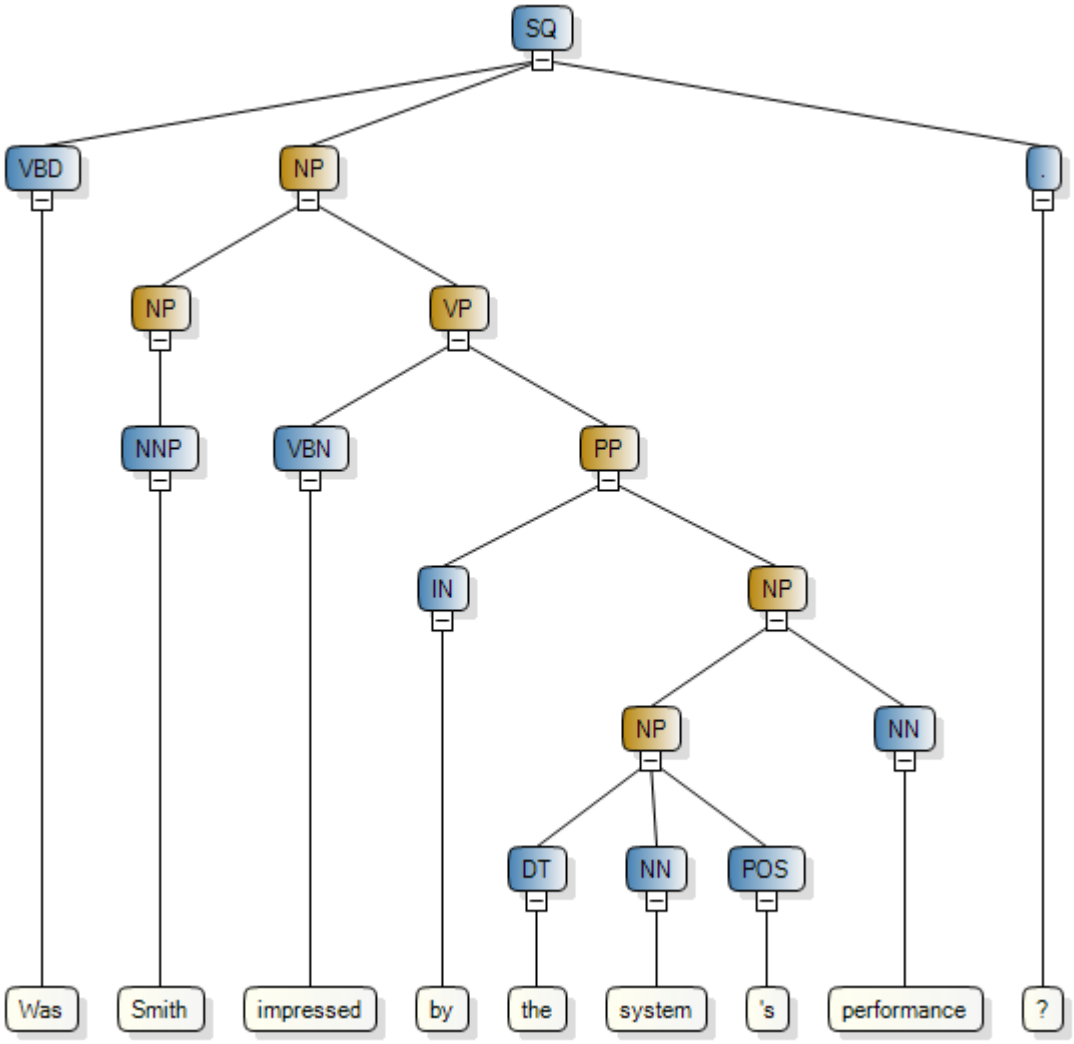
Parse P1 (1 of 1)



Parse P2 (1 of 1)


SQ
VBD
NP
.
NP
ADJP
NNP
JJ
PP
IN
NP
NP
NN
DT
NN
POS
Was
Smith
impressed
by
the
system
's
performance
?

Parse Q (1 of 2)



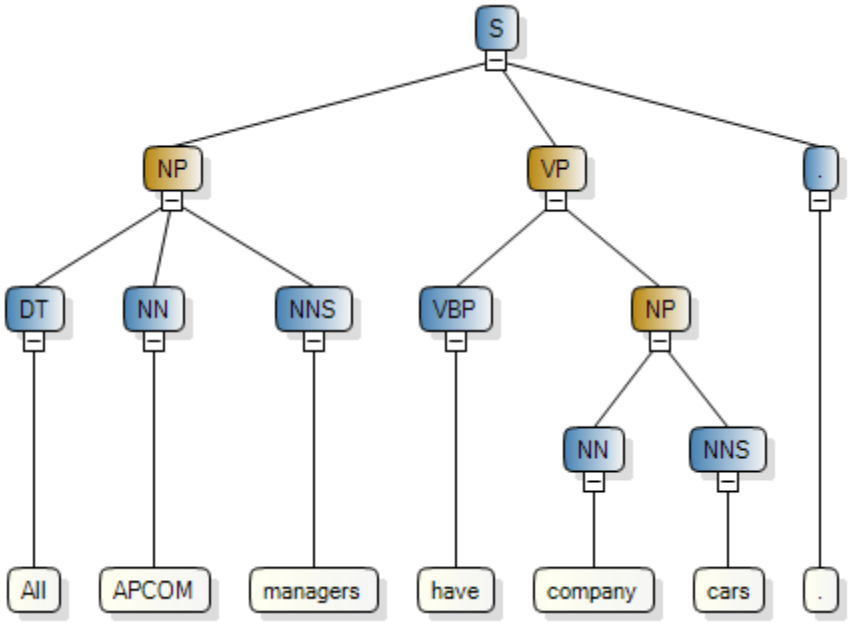
Parse Q (2 of 2)

Section 2.4 - Example 9 – Problem 103 from FraCaS Test Suite

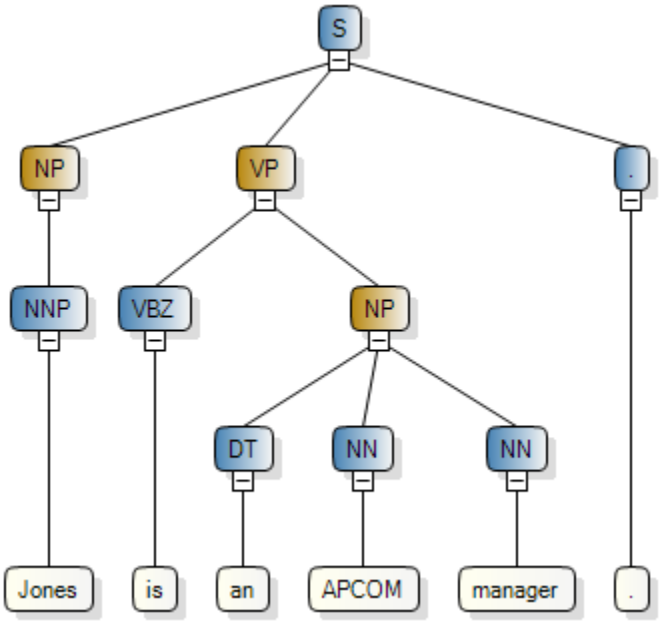
Problem 103 comes from section 2.4 of the FraCaS Test Suite. This particular section focuses on Dependent Plurals.

Table D-4. Problem 103 from Section 2.4 - FraCaS Test Suite

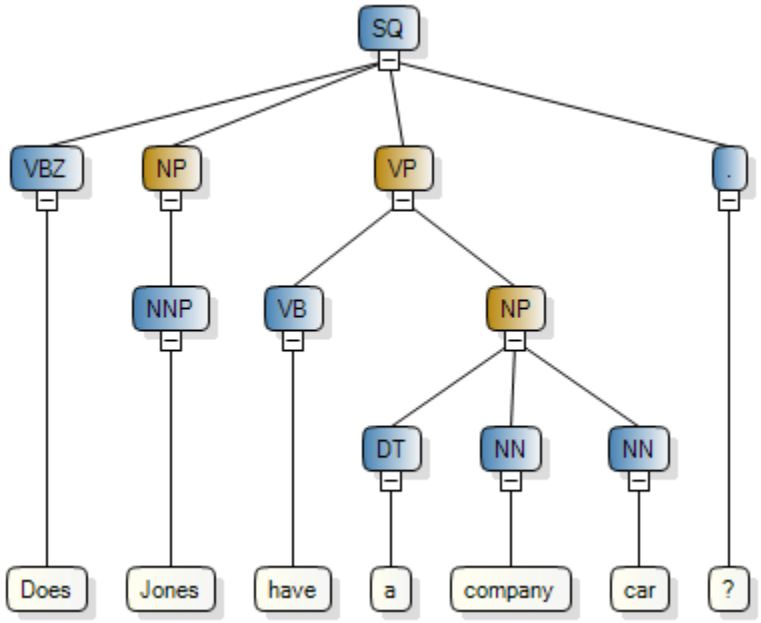
P1	All APCOM managers have company cars.
P2	Jones is an APCOM manager.
Q	Does Jones have a company car?



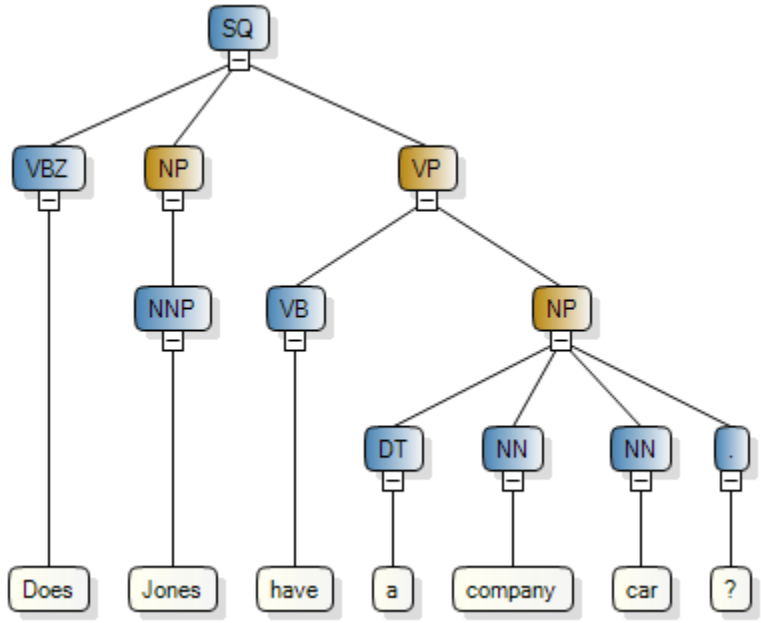
Parse P1 (1 of 1)



Parse P2 (1 of 1)



Parse Q (1 of 2)



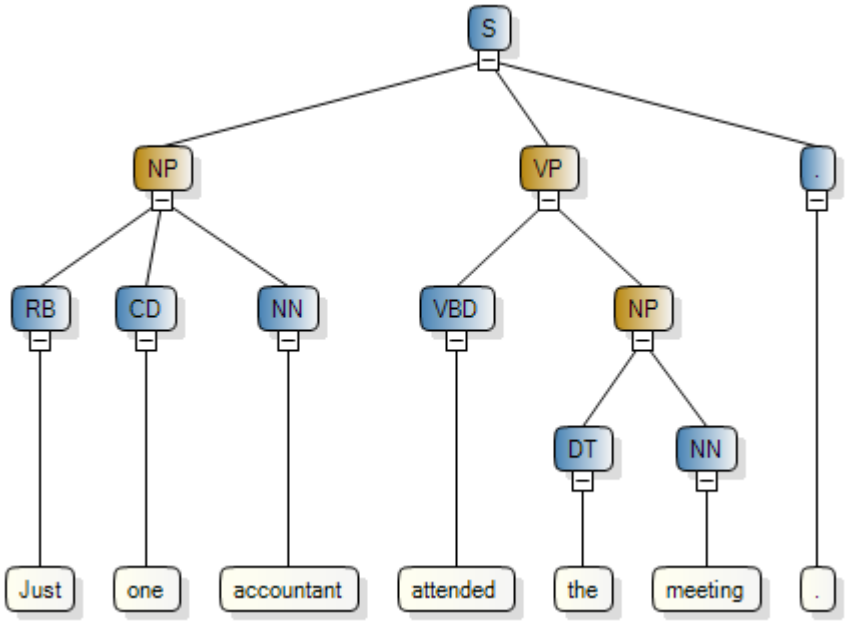
Parse Q (2 of 2)

Section 2.5 - Example 10 – Problem 105 from FraCaS Test Suite

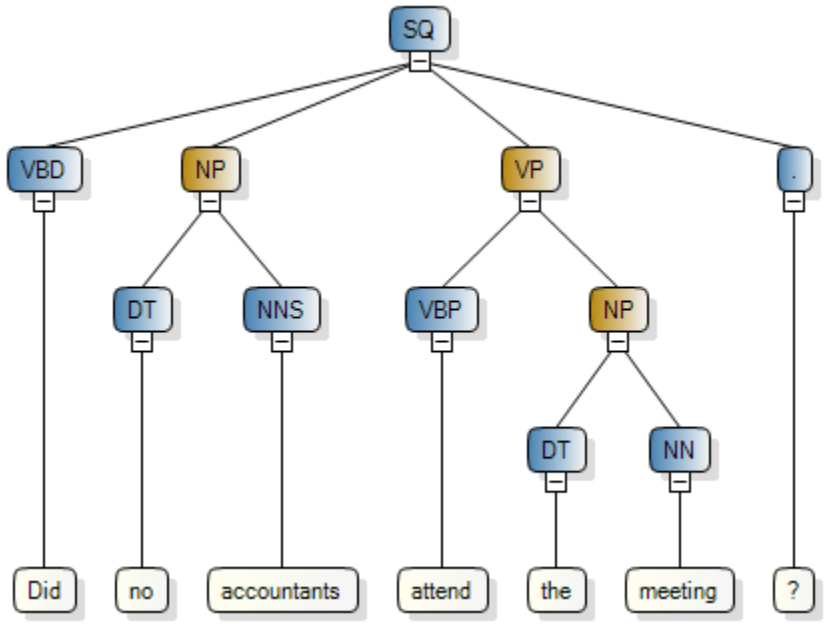
Problem 105 comes from section 2.5 of the FraCaS Test Suite. This particular section focuses on Negated Plurals.

Table D-5. Problem 105 from Section 2.5 - FraCaS Test Suite

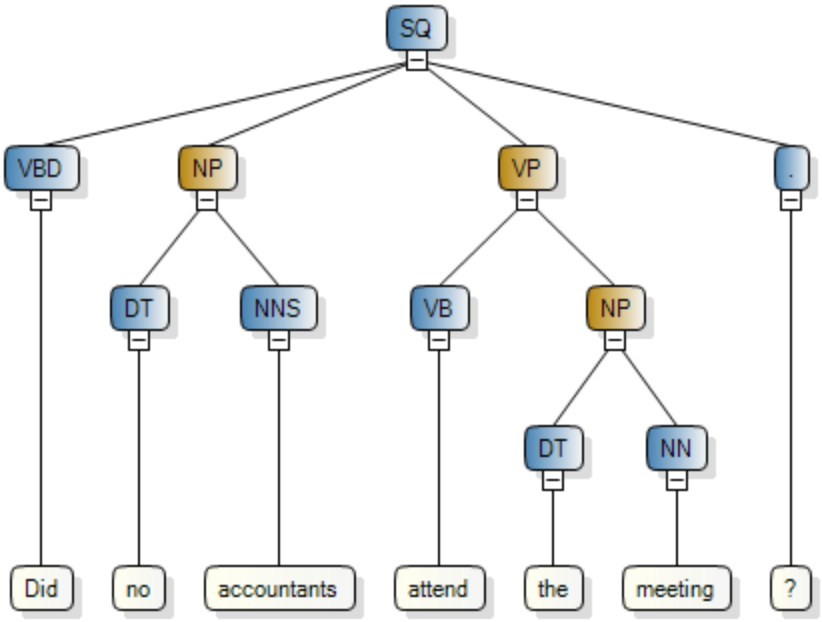
P1	Just one accountant attended the meeting.
Q	Did no accountants attend the meeting?



Parse P1 (1 of 1)



Parse Q (1 of 2)



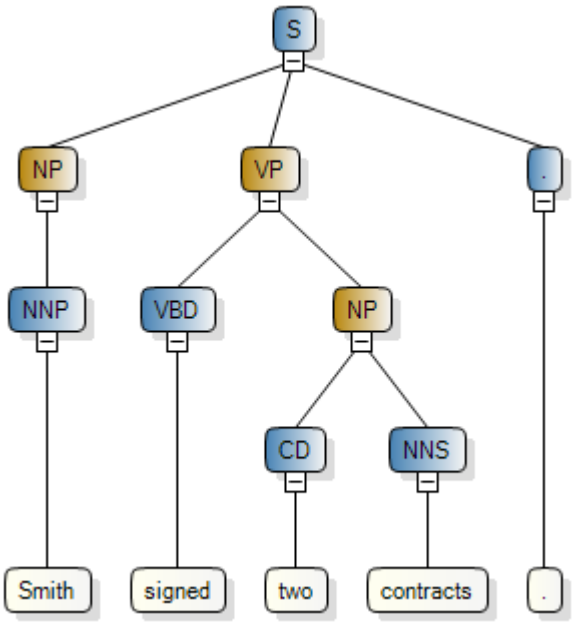
Parse Q (2 of 2)

Section 2.6 - Example 11 – Problem 113 from FraCaS Test Suite

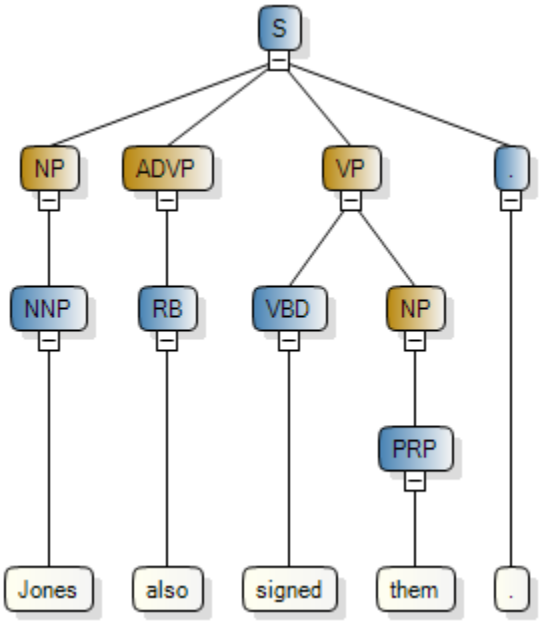
Problem 113 comes from section 2.6 of the FraCaS Test Suite. This particular section focuses on Collective and Distributed Plurals.

Table D-6. Problem 113 from Section 2.6 - FraCaS Test Suite

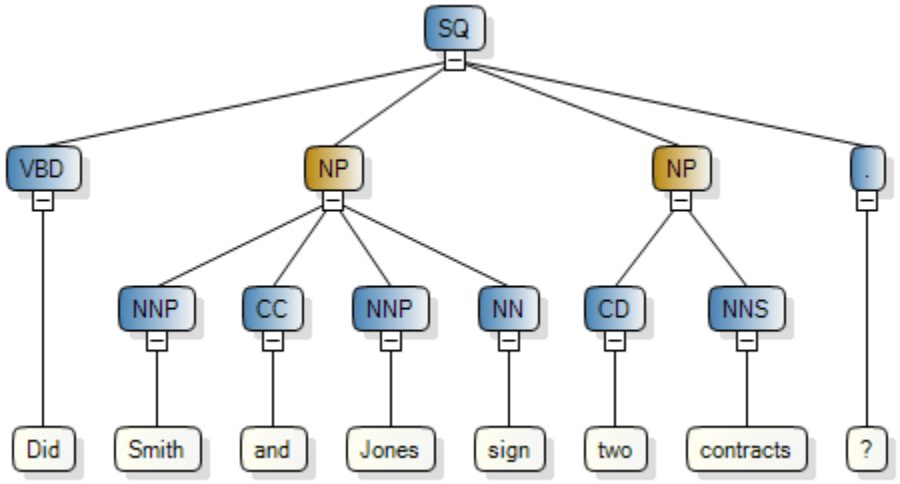
P1	Smith signed two contracts.
P2	Jones also signed them.
Q	Did Smith and Jones sign two contracts?



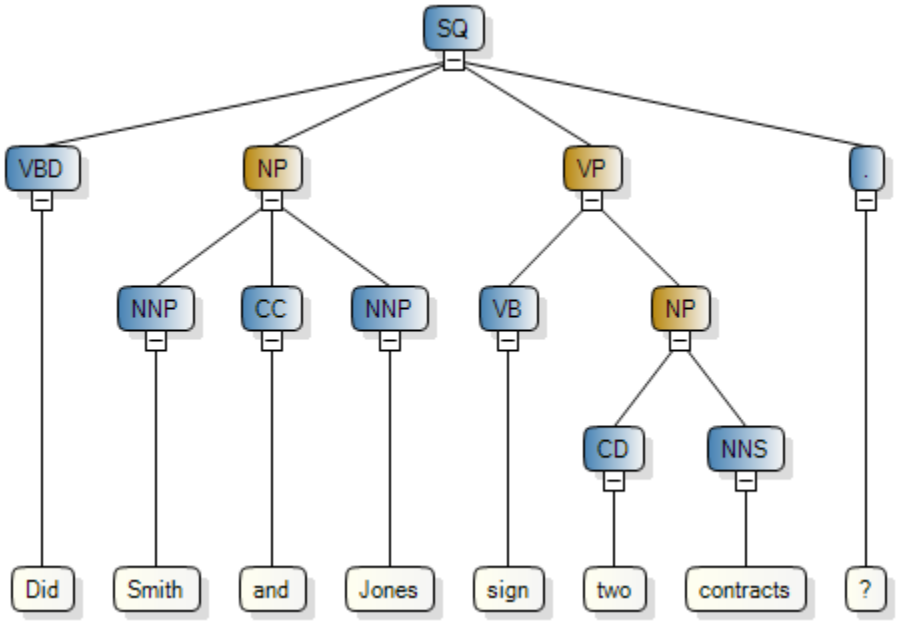
Parse P1 (1 of 1)



Parse P2 (1 of 1)



Parse Q (1 of 2)



Parse Q (2 of 2)

Appendix E - Adjectives

Section 5.1 - Problem 197 from FraCaS Test Suite

Problem 197 comes from section 5.1 of the FraCaS Test Suite. This particular section focuses on Affirmative and Non-Affirmative Adjectives.

Table E-1. Problem 197 from Section 5.1 - FraCaS Test Suite

P1	John has a genuine diamond.
Q	Does John have a diamond?

The system starts at the top of the tree and works its way down left to right looking at each node and analyzing what its children nodes are (if applicable). First, we look at the S node. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object. Now looking at the [NP] node (Noun Phrase), it has one child, [NNP]. Since there is no article we can for now assume the statement is referring to a specific [NNP] (John), which is important for matching purposes later on. The name of this noun object is 'John'. We know that there is only one 'John' that is being talked about as well. At this point, we have placed into the temporary ontology an instance of a 'John', where the quantity is only 1 and that it contains no attributes at this point in time (or at all in this example).

Next, we look at the [VP] node. The [VP] node contains two children, both a [VBZ] node (Verb, third-person singular present), and a [NP] node. We look at the verb 'has' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a present tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurs in present time at time t = 0.

We now have the final noun object for this verb phrase to process. We have a [NP], a noun phrase, to process. In the [NP], there are three children [DT] (article), [JJ] (adjective), and [NN] (generic noun singular). We create a new noun object. The system now looks at [DT], sees that it is 'a'. From that it knows that the noun object is referring to a generic object. We specify that the noun object is going to be a generic object. Next, the system looks at the node [JJ] (adjective). There is only one child underneath this, it is 'genuine'. Adjectives have two phases of special case processing. One, once the object itself is completed and again after the entire statement or question is close to being finished processing. The final node [NN] under [NP] is now examined by the system, which returns a 'diamond' which is a generic singular noun. So we combine the three children together under one noun object, which tells us we have a generic diamond that is genuine. This noun object is also added to the temporary ontology.
Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of 'diamond' to the instance of John via the word 'has'.

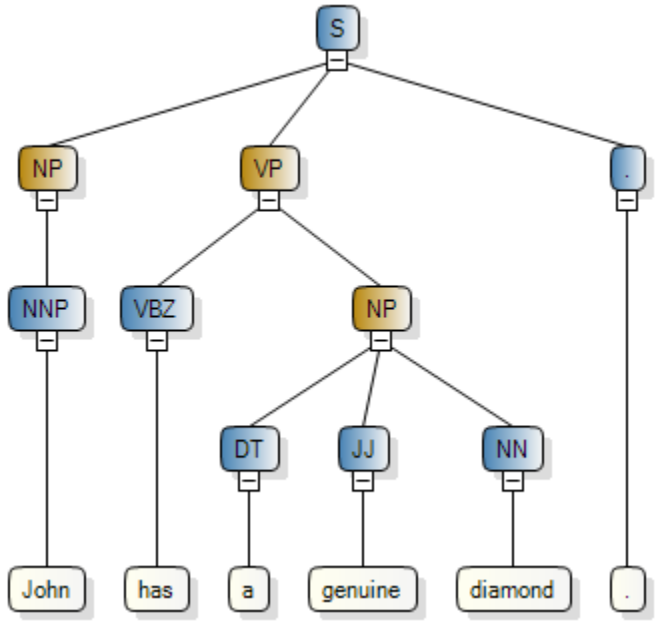


Figure E-1 Parse P1 (1 of 2)

The system has constructed the temp ontology shown in Figure E-2. It looks at every predicate begins to assimilate the data. The only predicate in this particular premise is the 'has'

predicate. Since 'has' is past tense, it is normalized to present tense, notating that it is in the past tense. The rule applied in this case is chosen based on the type of the parameters used in this predicate (other predicates/rules can look at not just the type but also properties/attributes/etc. to identify which rule to use). In this case, they are both instances (Proper Noun, and generic object), which is to say we need to create a relationship between 'John' and 'diamond' via 'has'.

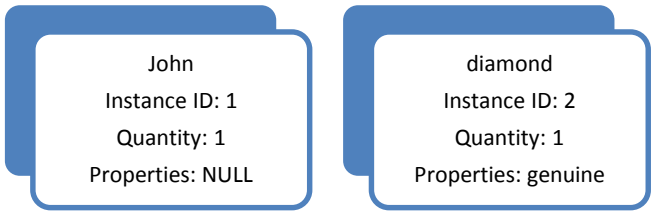


Figure E-2 Temporary Ontology from Parse 1 of P1

Now that all predicates within the first premise have been processed, we merge this temp ontology into the main ontology. This is a trivial process for this particular case as there currently exists nothing in the main ontology. The system simply copies every instance/class/relation over.

Once the ontologies have been merged, the main ontology is analyzed to look for new relationships that can be constructed by this new knowledge. Since an instance can only be matched to another instance if it matches a parent class, aka they are of the same concept, and we must construct a parent class for any instance that currently has no parent class. For this premise, we have two instances both 'John' and 'diamond' neither has a parent class. We construct a parent class for each of these maintaining the exact properties as seen in them. We create a parent class for instance 'John', which maintains the same quantity and same properties which in this case 'John' has no properties. Finally for 'John', the class 'John' that contains all 'John's is created as noted by the quantity set to ALL. This is shown in Figure E-3.

However, 'diamond' has a property, this property 'genuine' is looked at to see what if it has any special meaning, which it does not, so we create a parent class that maintains the

quantity as well as the property ‘genuine’ and make it the parent class of the instance ‘diamond’. We make this class the parent of the instance ID: 2. Following this, we need to generate a parent class for that which is the exact same minus the properties. Then the system builds the all-encompassing class which contains all types of diamonds which has the quantity set to ALL. This is shown in Figure E-3.

Have< = t + 0>(Instance ID: 1, Instance ID: 2)

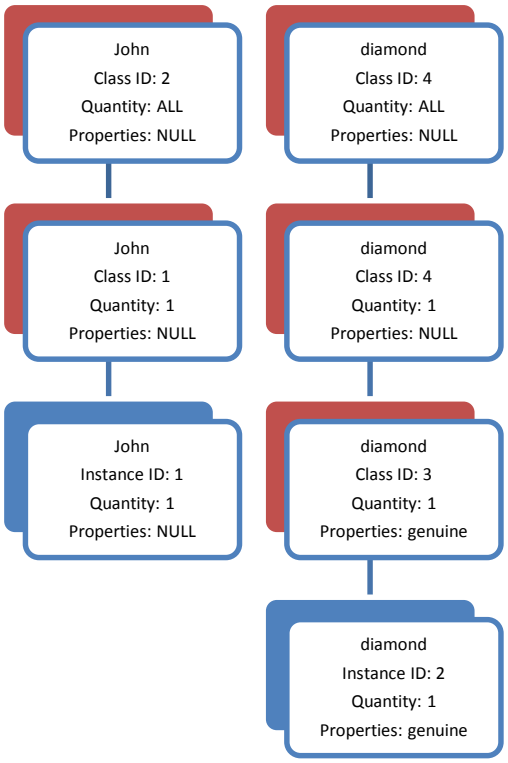


Figure E-3 Main Ontology after P1 - 5.1

The system has already found a valid parse for premise one. If the system was initially presented with the parse shown in Figure E-4, it would reject the parse due to the clear structural problems it has. From the top perspective (root) there is no punctuation. The punctuation is required only at the root level. So due to structural reasons alone we can reject this parse for premise one.

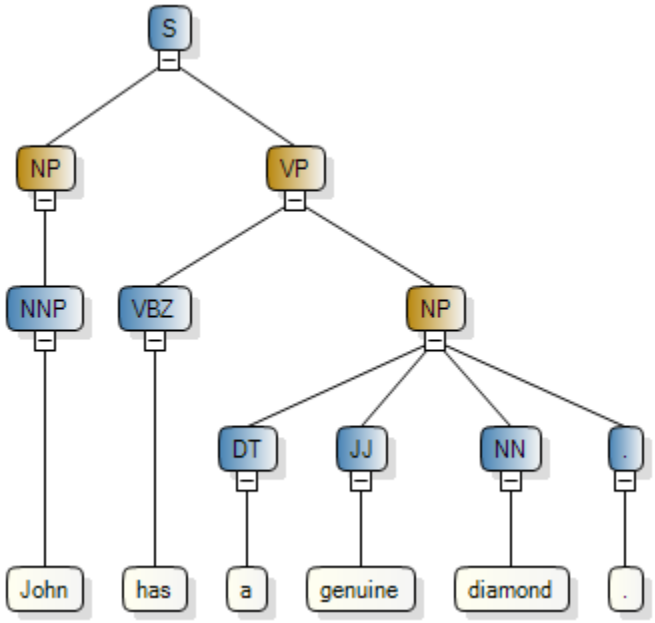


Figure E-4 Parse P1 (2 of 2)

The question is the final input to analyze. The question has two parses shown in Figure E-5, and Figure E-7. The system must go through each parse until it finds a valid parse. The problem in Figure E-7 is that the '?' is in the wrong part of the tree. So we must rule this particular parse out, but since it comes second the system does not even see it as it is shown that the first parse shown in Figure E-5 is valid.

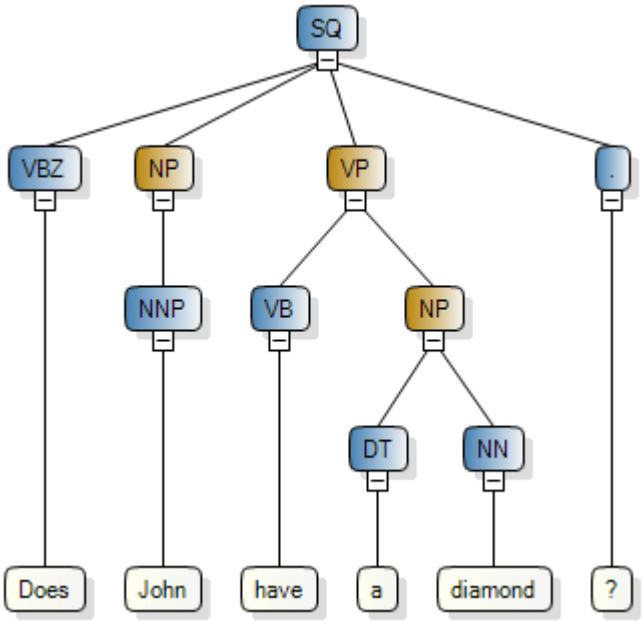


Figure E-5 Parse Q (1 of 2)

It is possible even with a valid parse, that the words within the parse could mean that the parse was in fact not correct. The parent for this particular parse tree [SQ] (Inverted yes/no question, or main clause of a wh-question, following the wh-phrase in SBARQ) has four children, [VBZ], [NP], [VP], and [.]. These four types indicate that it is a yes no type question without further inspection of the nodes. We see the first node is [VBZ] and load the verb into the yes no question. Next, we have [NP], we create a noun object to place into the question yes-no object. We look at the children of [NP] and there is only one child [NNP]. So we simply return 'John' along with the fact that we know it is singular and a proper noun to complete our noun object and from this create an instance of 'John', where we have exactly one 'John'. The temporary ontology has been created and is shown in Figure E-6.

HAVE< <= t + 1>(Instance ID: 3 (John), Instance ID: 4 (diamond))

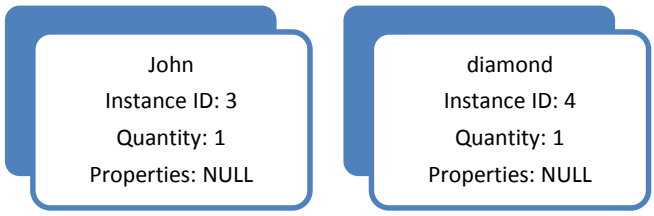


Figure E-6 Temporary Ontology for Questions

To answer this particular question, we see that there is no missing elements or unknowns to fill in; we must first match each particular instance with everything in the ontology to see which items are potential candidates for matching the particular constraints of the temporary ontology (aka the question). We look at all possible combinations (very expensive operation) to see when if we have a match a complete match across all constraints on the temp ontology. If we can find a match for every instance and relationship found in Figure E-6 then we will have a match indicating a yes. However, if we find evidence to the contrary then we know that the answer is 'no'. If there is not sufficient information the answer would be 'unknown'. First, find all possible options for each instance found, 'John' (instance ID: 3) and 'diamond' (instance ID: 4). For 'John' (instance ID: 3), we see that it matches the class 'John' (Class ID: 1) and we test against all possible children of that class. There is only one child, which is 'John' (instance ID: 1). It matches on the name, quantity, and properties exactly so this is a direct match. Next, we test for 'diamond', try to find a class that exactly matches, which we find that 'diamond' (class ID: 4) matches. So we test against all sub children (looking for an instance that satisfies our constraint). Since, a genuine diamond is a diamond, according to the main ontology, it is a match. So we keep 'diamond' (instance ID: 2) as a match, there are no other possible options. So next we test to see if there exists a 'have' relationship that exists between the two of them where 'John' (instance ID: 1) is the one that 'have' a 'diamond' (instance ID: 2), additionally the time in which the system learned John has a diamond must have already occurred prior to this question being asked and there is. So in this case we have a match and can answer that yes John has a diamond.

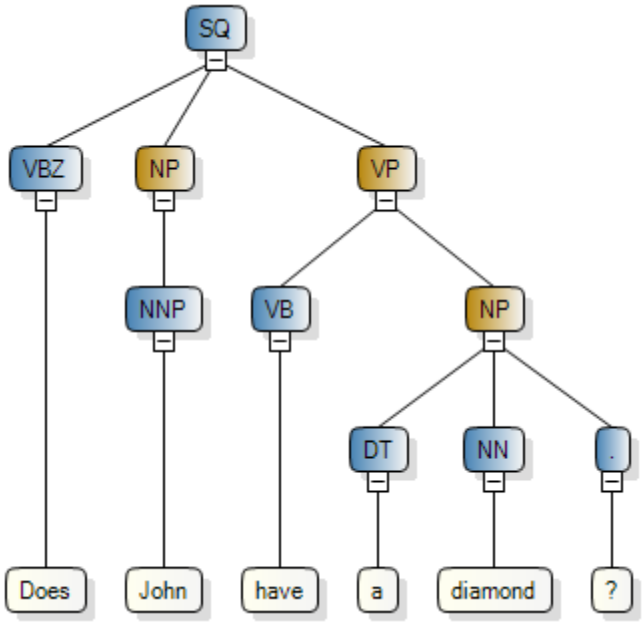


Figure E-7 Parse Q (2 of 2)

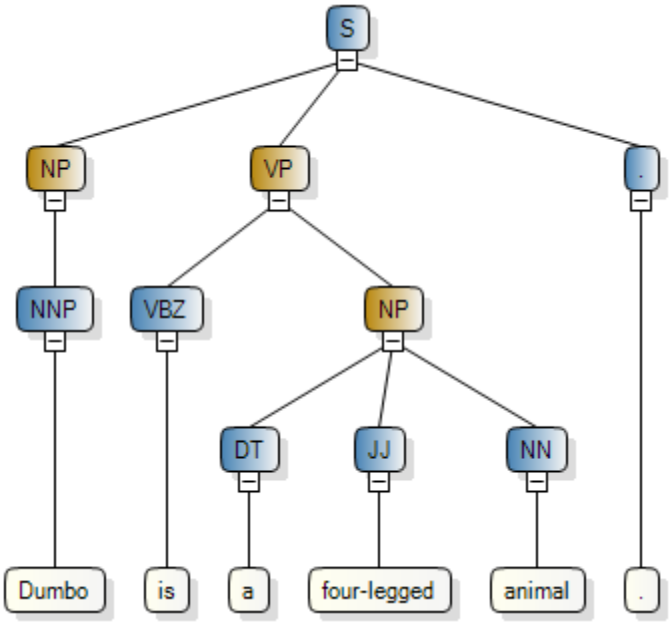
We can see from Figure E-7 that there are blatant structural problems from the lack of a '.' node at the child level from the root SQ. So this would have been discarded had it been seen by the system.

Section 5.2 - Problem 203 from FraCaS Test Suite

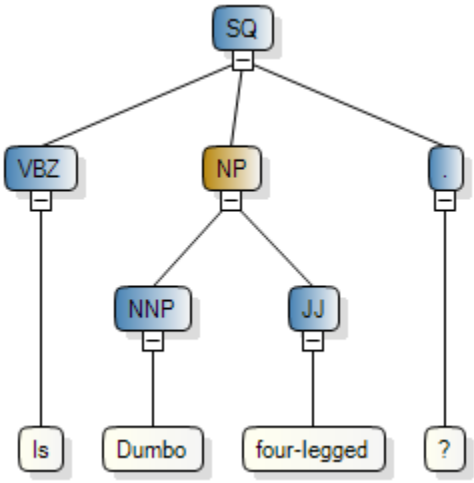
Problem 203 comes from section 5.2 of the FraCaS Test Suite. This particular section focuses on No Comparison Class Adjectives.

Table E-2. Problem 203 from Section 5.2 - FraCaS Test Suite

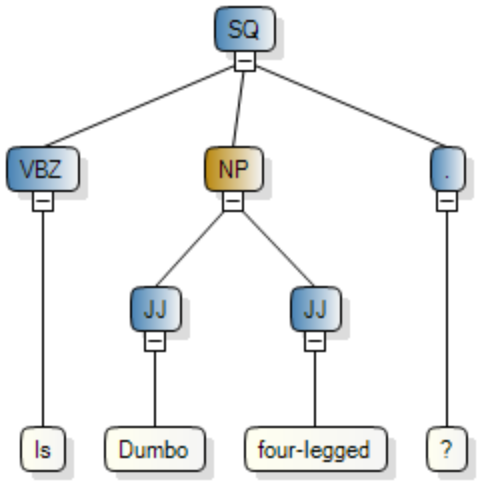
P1	Dumbo is a four-legged animal.
Q	Is Dumbo four-legged?



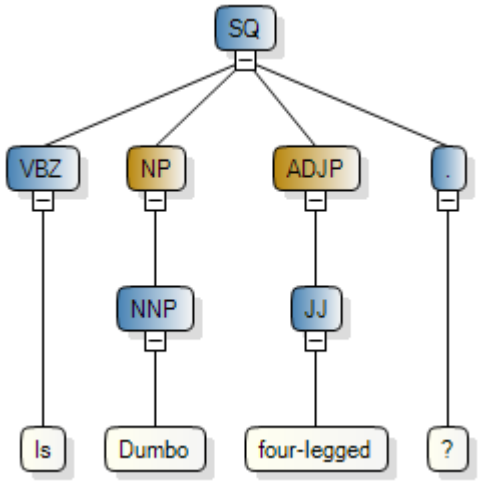
Parse P1 (1 of 1)



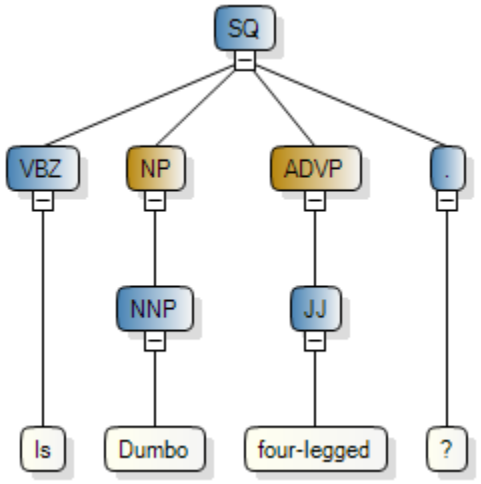
Parse Q (1 of 4)



Parse Q (2 of 4)



Parse Q (3 of 4)



Parse Q (4 of 4)

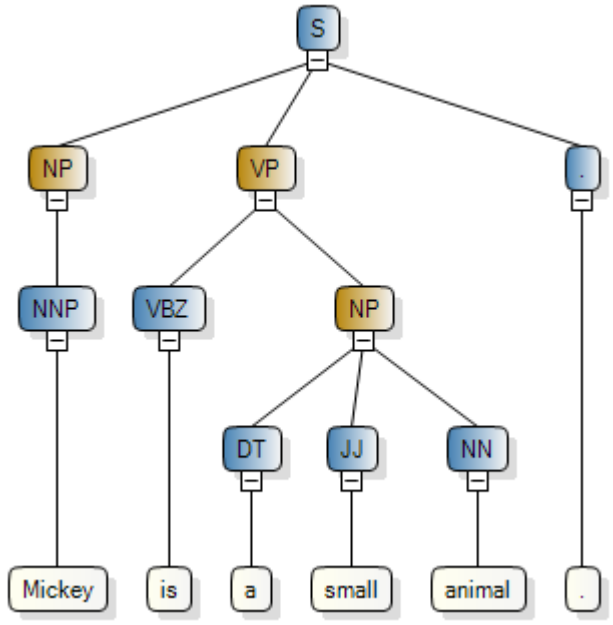
Section 5.3 - Problem 204 from FraCaS Test Suite

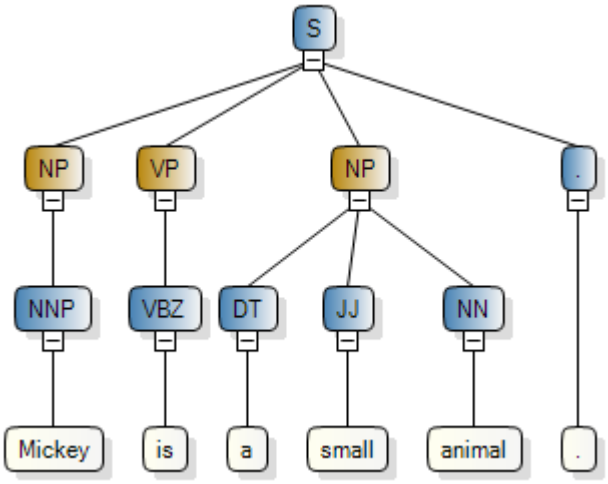
Problem 204 comes from section 5.3 of the FraCaS Test Suite. This particular section focuses on Opposite Adjectives.

Table E-3. Problem 204 from Section 5.3 - FraCaS Test Suite

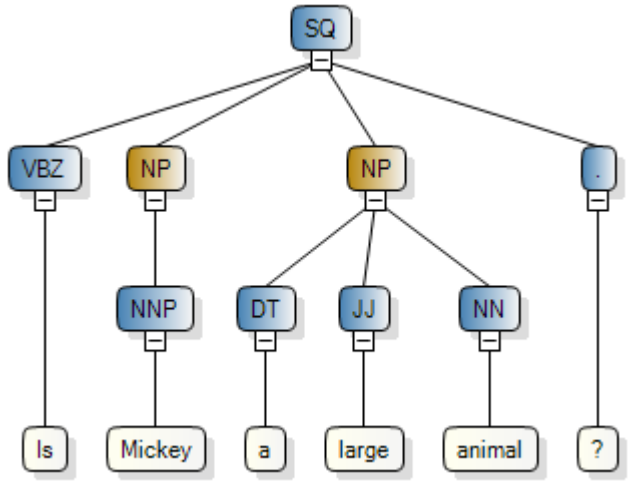
P1	Mickey is a small animal.
Q	Is Mickey a large animal?

Parse P1 (1 of 2)

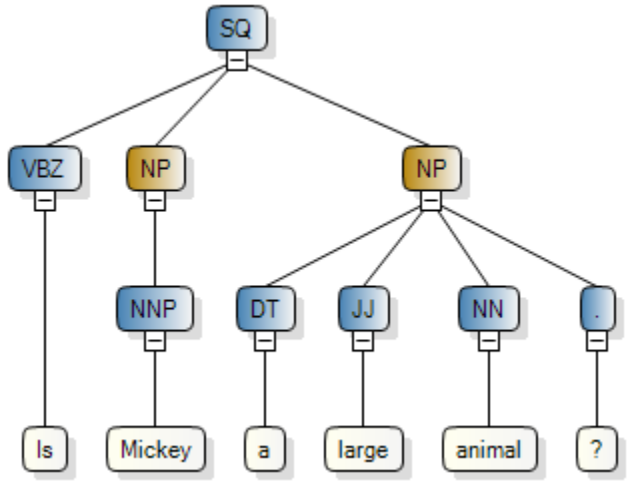




Parse P1 (2 of 2)



Parse Q (1 of 2)



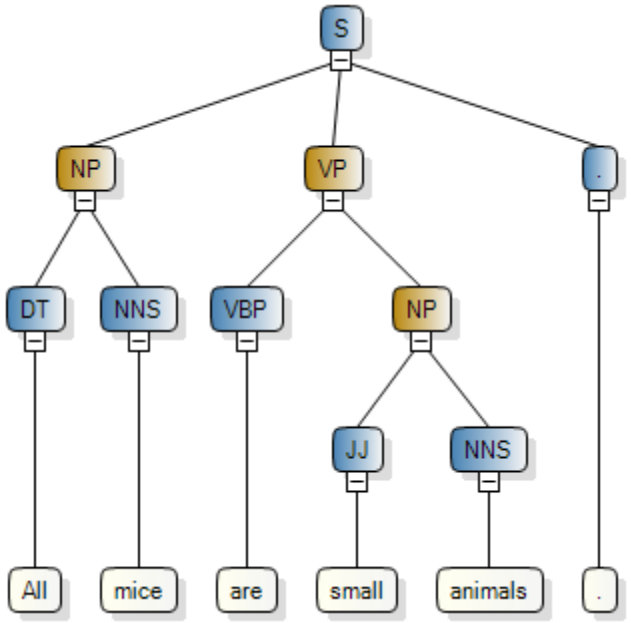
Parse Q (2 of 2)

Section 5.4 - Problem 210 from FraCaS Test Suite

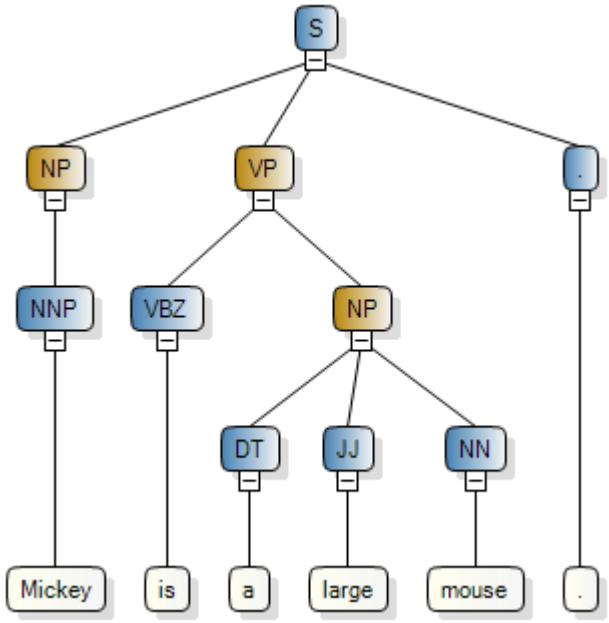
Problem 210 comes from section 5.4 of the FraCaS Test Suite. This particular section focuses on Extensional Comparison Adjectives.

Table E-4. Problem 210 from Section 5.4 - FraCaS Test Suite

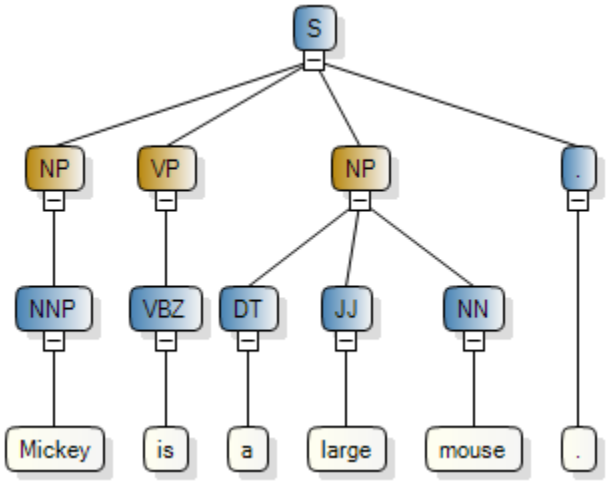
P1	All mice are small animals.
P2	Mickey is a large mouse.
Q	Is Mickey a large animal?



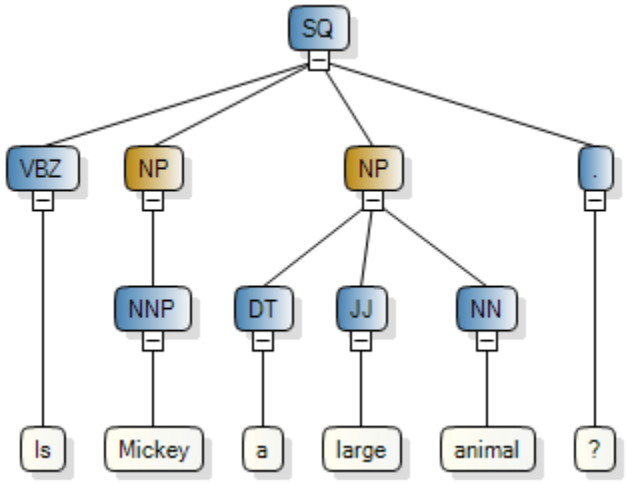
Parse P1 (1 of 1)



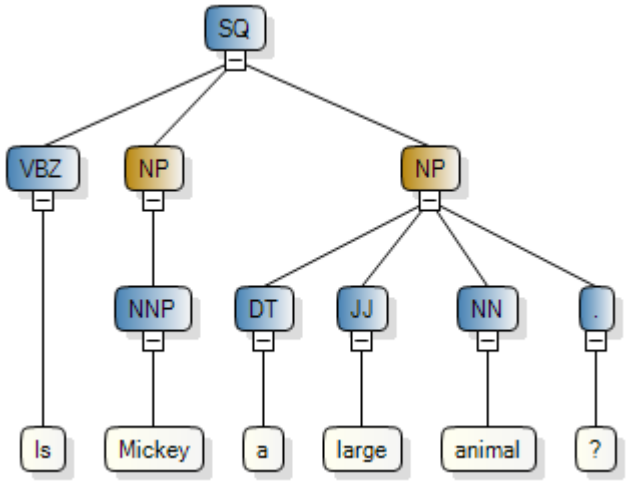
Parse P2 (1 of 2)



Parse P2 (2 of 2)



Parse Q (1 of 2)



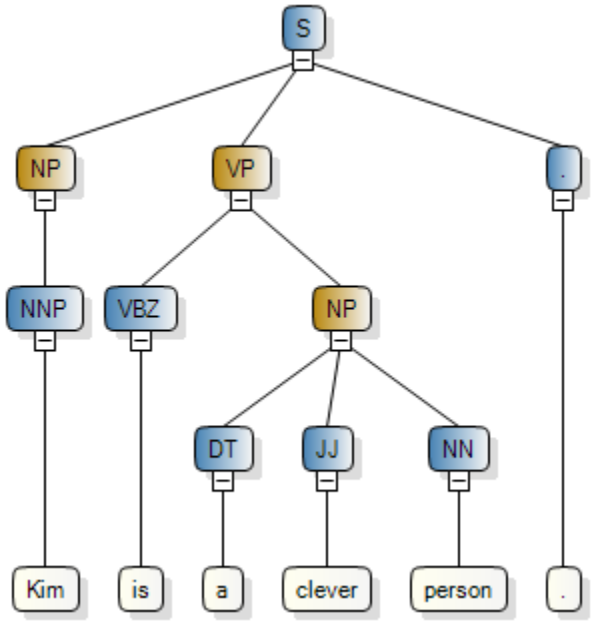
Parse Q (2 of 2)

Section 5.6 - Problem 218 from FraCaS Test Suite

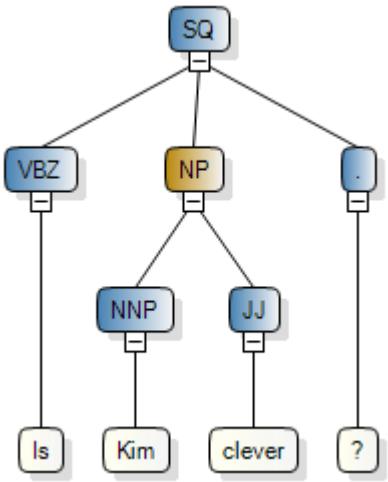
Problem 218 comes from section 5.6 of the FraCaS Test Suite. This particular section focuses on Default Comparison Class Adjectives.

Table E-5. Problem 218 from Section 5.6 - FraCaS Test Suite

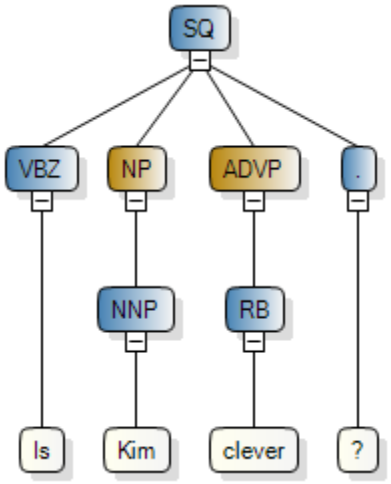
P1	Kim is a clever person.
Q	Is Kim clever?



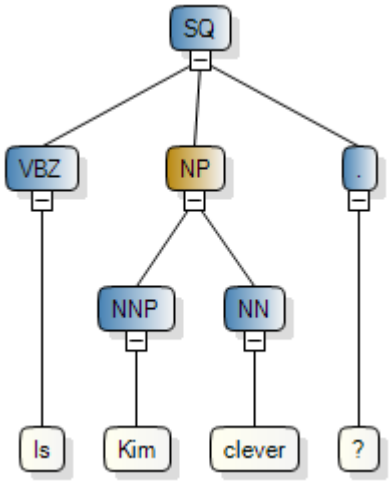
Parse P1 (1 of 1)



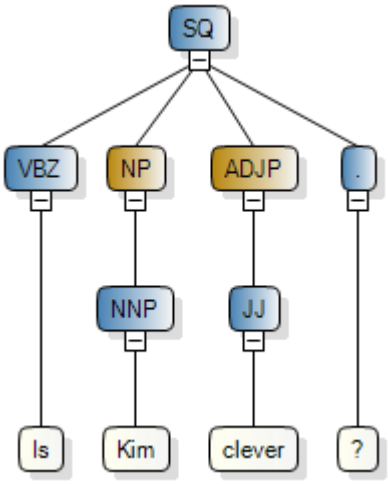
Parse Q (1 of 4)



Parse Q (2 of 4)



Parse Q (3 of 4)



Parse Q (4 of 4)

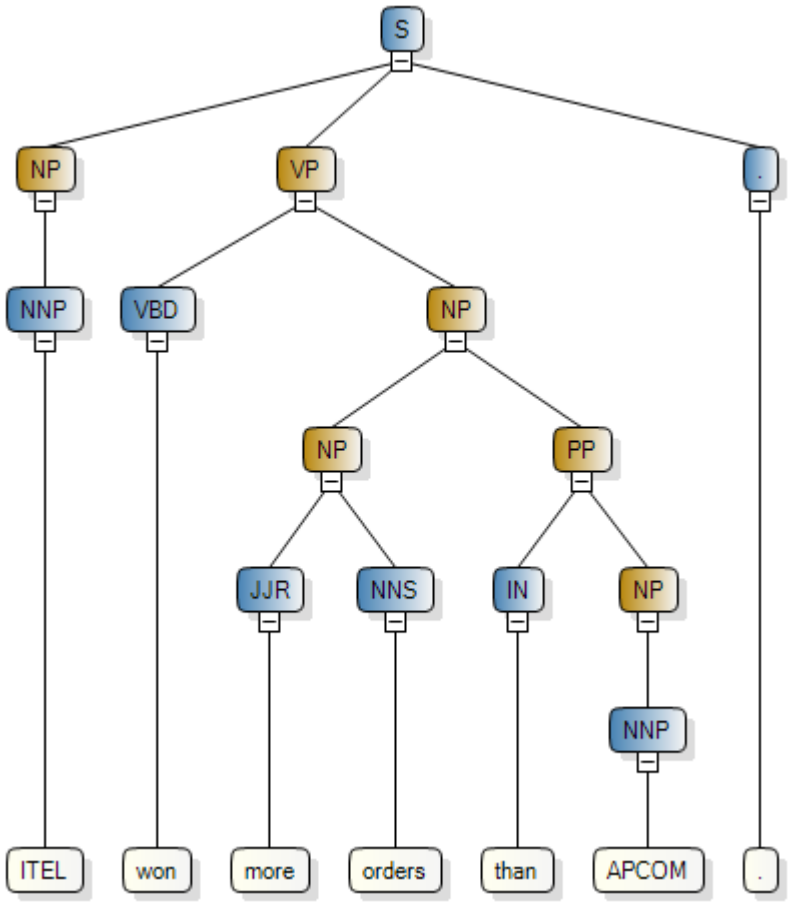
Appendix F - Comparatives

Section 6.1 - Problem 233 from FraCaS Test Suite

Problem 233 comes from section 6.1 of the FraCaS Test Suite. This particular section focuses on Phrasal Comparatives.

Table F-1. Problem 233 from Section 6.1 - FraCaS Test Suite

P1	ITEL won more orders than APCOM.
Q	Did ITEL win some orders?



Parse P1 (1 of 1)

This problem contains only the premise walk through.

First, we look at the S node. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in

terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object.

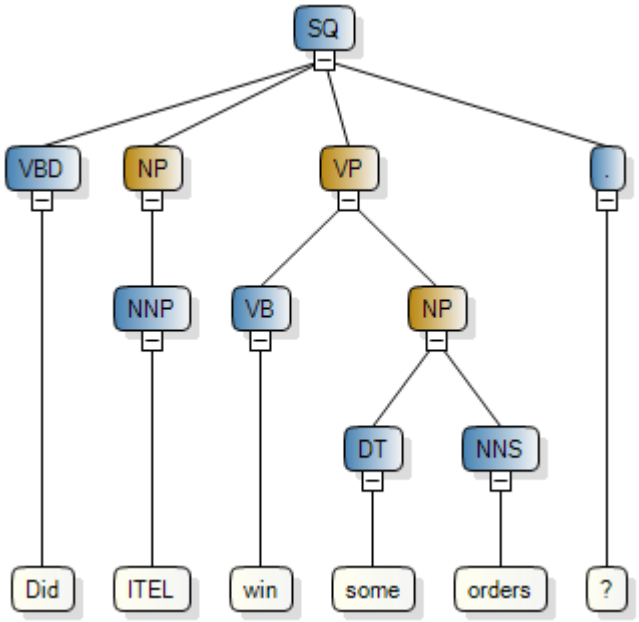
Now looking at the [NP] node (Noun Phrase), it a single child, NNP which tells the system that it is a singular proper noun. We look at NNP, we know that this particular noun is referring to a specific [NNP], as there is no article indicating a generic reference be created, which is important for matching purposes later on. We know that "ITEL" refers only to one "ITEL" from the fact that it is a [NNP]. In addition to that we know we are talking about a specific "ITEL". Since this is the only child of [NP] we are done with the initial noun object. It has no attributes and a quantity of one.

Next, we look at the [VP] node (Verb Phrase). The [VP] node contains two children, both a [VBD] node (past tense), and a [NP] node. We look at the verb 'won' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a past tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurred at some point in the past we can say that it occurred at time $t < 0$.

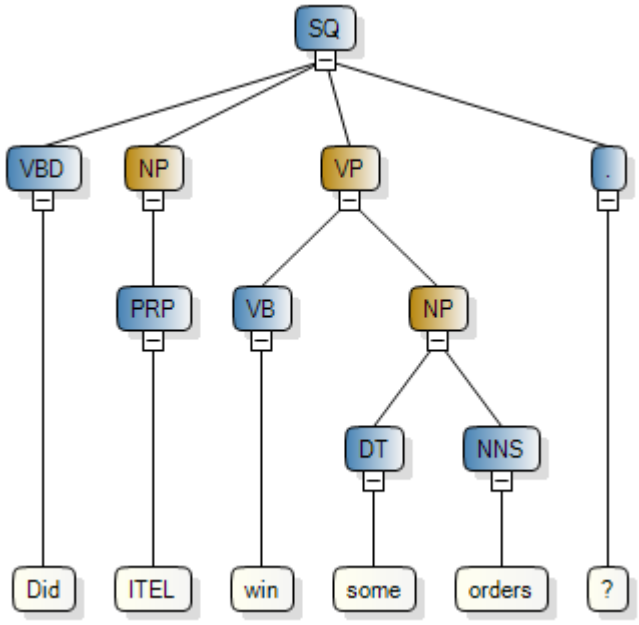
We now have the final noun object for this verb phrase to process. The node [NP] has two children [NP] and [PP] (prepositional phrase). Looking at the first child of [NP], we see it has two children [JJR] (Adjective, comparative) and a [NNS]. The child of [JJR] is a "more", we add this as an attribute to the noun object which is in progress of being created. We now take a look at the second child, [NNS] (plural noun), "orders" and load that into the noun object.

Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of Italian to the instance of contract via the word 'signed'.

At this point, it doesn't matter when constructing the temp ontology what definition 'signed' is referencing, so long as it has valid parameters so as to not reject the statements parse outright. During the inference stage is when the definition is chosen and any additional inferences are inferred at this point.



Parse Q (1 of 2)



Parse Q (2 of 2)

Appendix G - Temporal Reference

Section 7.1 - Problem 251 from FraCaS Test Suite

Table G-1. Problem 251 from Section 7.1 - FraCaS Test Suite

P1	ITEL has a factory in Birmingham.
Q	Does ITEL currently have a factory in Birmingham?

"has" normalized to "have"

Have< = t + 0>(Instance ID: 1, Instance ID: 2)

in (Instance ID: 2, Instance ID: 3)

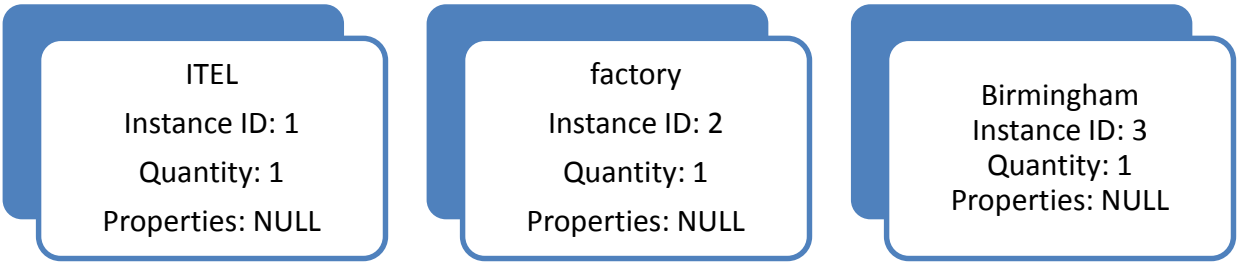


Figure G-1 Temp Ontology for Problem 251 P1

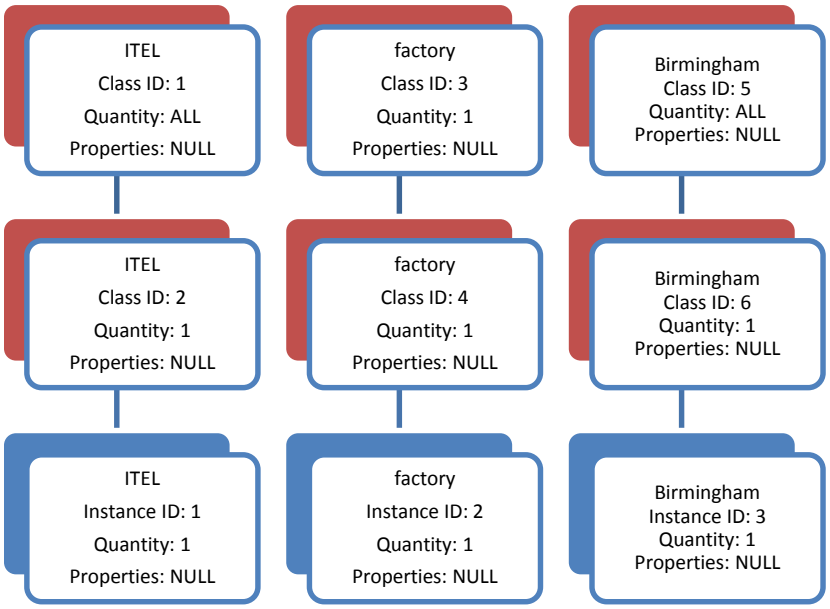


Figure G-2 Merged Ontology with Generated Knowledge for Problem 251 P1

Appendix H - Verbs

This section focuses on verbs. This section emphasizes the importance of verb normalization. This is to take a verb and convert it to its present tense form while maintaining the information of the tense so it is trivial to compare against other verbs without a future lookup (even though it would still be possible to do it this way). For a match to occur when comparing verbs, the verb name itself much match, along with the parameters, and additionally the tense information must align.

Section 8.2 - Example ? – Problem 331 from FraCaS Test Suite

Problem 331 comes from section 8.2 of the FraCaS Test Suite. This particular section focuses on Phrasal Comparatives. This problem follows the example from Section 2.1 problem 81 almost identically except for a different verb. The entire process is listed here for completeness.

Table H-1. Problem 331 from Section 8.2 - FraCaS Test Suite

P1	Smith and Jones left the meeting.
Q	Did Smith leave the meeting?

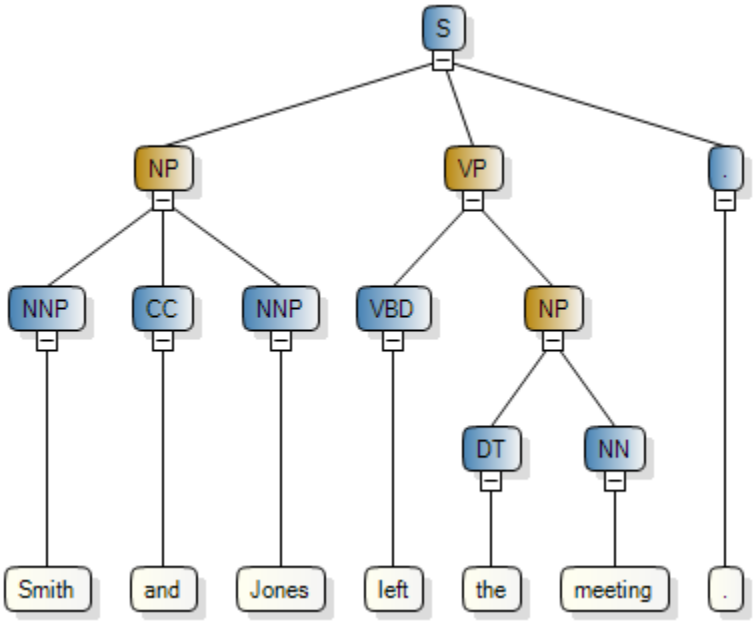


Figure H-1 Parse P1 (1 of 1) for Problem 331

First, we look at the S node. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object.

Now looking at the [NP] node (Noun Phrase), it has three children, NNP, CC, and NNP. This form indicates that there is a list of nouns separated by some conjunction. First we look at the first NNP, we know that this particular noun is referring to a specific [NNP], as there is no article indicating a generic reference be created, which is important for matching purposes later on. Finally, we finish this noun object by placing Smith as the name for this particular object. Observe that there is no article modifying the NNP and we know that the NNP stands for a singular Proper Noun; we know the quantity of this particular object is 1. Since there are no other properties or attributes to attach to the Object Smith at this point we proceed with the processing the next term in the noun list from the [NP]. Next we have [CC] which stands for Coordinating Conjunction. This term, only has 1 child which is 'and'. So we return this new and conjunction to the noun list to tell it how the rest of the sentence can affect the set that is formed from the noun list. Next, we have the second [NNP] which is for Jones and created similarly as Smith. At this point, we have placed into the temporary ontology an instance of a set that contains a reference to each of the follow: Smith, and Jones combined with an and conjunction where the quantity is only 1 and that it contains no attributes at this point in time. Next, we look at the [VP] node (Verb Phrase). The [VP] node contains two children, both a [VBD] node (past tense), and a [NP] node. We look at the verb 'left' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a past tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurred at some point in the past we can say that it occurred at time $t < 0$.

We now have the final noun object for this verb phrase to process. The node [NP] has two children [DT] and [NN]. The [DT] contains 'the', and [NN] contains 'meeting'. We know that given the [DT] 'the' we are referring to a specific item, where the quantity of said item is set to 1. (This is known from the fact that [NN] is a singular noun). The system tries to find an exact match in the main ontology first since it could exist already. Since it does not find a match, it creates an instance of contract. We add this instance of contract to the temp ontology as well.

Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of Italian to the instance of meeting via the word 'left'. At this point, it doesn't matter when constructing the temp ontology what definition 'left' is referencing, so long as it has valid parameters so as to not reject the statements parse outright. During the inference stage is when the definition is chosen and any additional inferences are inferred at this point.

The system now takes a look at everything derived from the first premise as a whole and begins to build the temp ontology shown in Figure H-2. It looks at every predicate and begins to assimilate the data. The only predicate (of interest) in this particular premise is the 'left' predicate. There is no additional information or facts generated from the use of the word 'left'. A dictionary look up of the word 'left', there could be up to 14 possibilities (ignoring the idiom), all of which mean the same base thing. Since 'left' is working with the left hand side of a noun list, we apply the verb to the set itself, and to each item in the set connected by the and conjunction in this case.

LEAVE< < t + 0 >(Instance ID: 1 (Smith), Instance ID: 3 (contract)) and LEAVE< < t + 0 >(Instance ID: 2 (Jones), Instance ID: 3 (meeting))

LEAVE< < t + 0 >(SET_1, Instance ID: 3 (meeting))

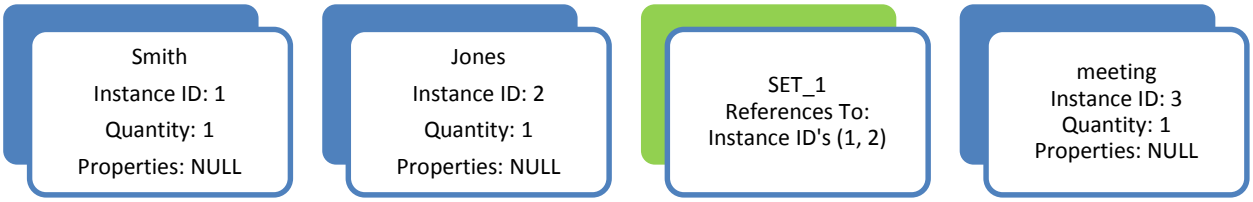
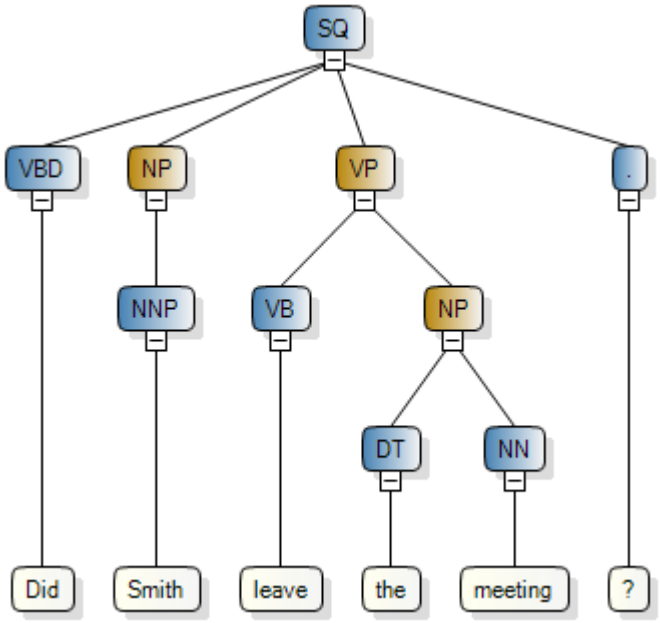


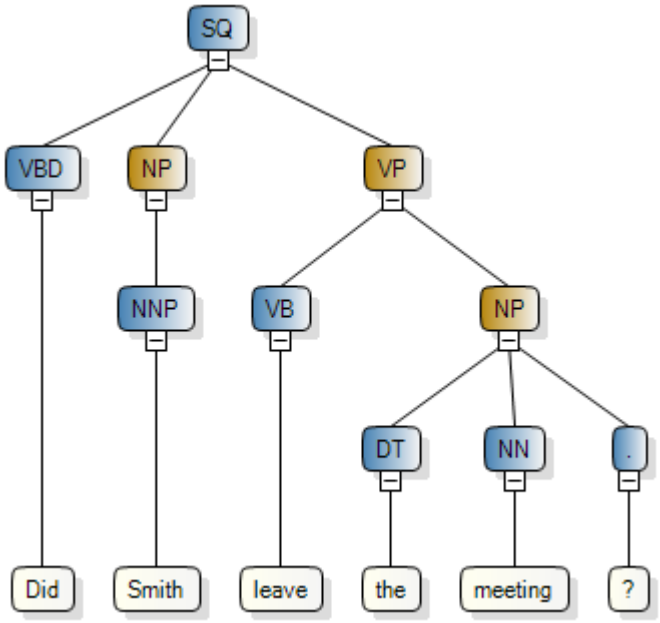
Figure H-2 Temp Ontology for P1 for Problem 331

Now that all predicates within the first premise have been processed, we merge this temp ontology into the main ontology. This is a trivial process for this particular case as there currently exists nothing in the main ontology. The system simply copies every instance/class/relation over.

Once the ontologies have been merged, the main ontology is analyzed to look for new relationships that can be constructed by this new knowledge. Since an instance can only be matched to another instance if it matches a parent class, aka they are of the same concept, and we must construct a parent class for any instance that currently has no parent class(es). For this premise, we have three instances 'Smith', 'Jones', and 'contract' and none of them have a parent class. We construct a parent class for each of these maintaining the exact properties as seen in them. Which in this case, no instance has any properties and note that the quantity is 1, so we create a class 'Smith' that has no properties and has a quantity of 1. We make this class the parent of the instance version of 'Smith'. We also must check to see if we can create or associate with a class 'Smith' with quantity ALL, that represents all Smiths that could exist. Since it originally had none, we know to create class Smith with quantity all (Class ID: 1) and insert a parent child relationship between the Class ID: 1 and class ID: 2. If said class exists we just provide add the parent child relationship between quantity 1 of class Smith (Class ID: 2). We do the same for instance 'Jones', and 'contract'. Taking a look at each relation within the ontology to see if there are any new facts that can be inferred. In this case, there is nothing additional to infer.



Parse Q (1 of 2)



Parse Q (2 of 2)

Appendix I - Attitudes

This section necessitates a statement pointer, which behaves much like a function pointer as defined previously, except that it points to an entire statement. We attach a "fuzzy" measurement to this to define if it is factual, not factual, or unknown. For these problems, there are no scalars or fuzzy modifiers to say that something might or might not be true. One example where you could weight it more towards probably true, is if an "actor" uses the verb think, such as 'Smith thinks Jones signed the contract'. That statement could not be guaranteed to be true, though depending how much you value Smith's opinion, you could say that Jones likely signed the contract. These types of problems are not found in this section.

The values for the confidence are represented by a decimal number in the set [0, 1]. Where 1 represents true or factual, 0 would represent a false or not factual, 0.5 would represent a true unknown, and any value greater than 0.5 would indicate that it is either likely more true or factual than not. Conversely, if the value was less than 0.5, it would be more likely that it was less true or not factual than it was to be true.

Section 9.2.2 - Problem 342 from FraCaS Test Suite

Problem 342 comes from section 9.2.2 of the FraCaS Test Suite. This particular section focuses on Veridicality Attitudes.

Table I-1. Problem 342 from Section 9.2.2 - FraCaS Test Suite

P1	Smith saw Jones sign the contract.
Q	Did Jones sign the contract?

The system starts at the top of the tree and works its way down left to right looking at each node and analyzing what its children nodes are (if applicable). First, we look at the S node. It has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), looks at the [.] first. It notices that it contains a '.'. Therefore the form as already indicated by NP VP is a premise. So we first build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object.

Now looking at the [NP] node (Noun Phrase), it has one child, [NNP]. Since there is no article we can for now assume the statement is referring to a specific [NNP] (Smith), which is important for matching purposes later on. The name of this noun object is 'Smith'. We know that there is only one 'Smith' that is being talked about in this context of this specific noun phrase. At this point, we have placed into the temporary ontology an instance of a 'Smith', where the quantity is only 1 and that it contains no attributes at this point in time (or at all in this example).
Next, we look at the [VP] node. The [VP] node contains two children, both a [VBD] node (Verb, third-person singular present), and a [S] node. This form factor some verb, followed by a statement indicates that we have an attitude detected. We look at the verb 'saw' and we begin to build our first relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a past tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurs in present time at time $t < 0$.

We now have the [S] node to process. This [S] node represents a statement by itself. This node has three children. The form of the children here indicates that this is a rather standard statement. The S node (handled differently than others in terms of processing order), checks if there is a '.', which there isn't so it then checks to make sure it is a child of another phrase, which in this case it is, so it continues. We build a statement object, this contains just a noun object and a verb object. Since the system works left to right, we first look at the [NP] node to construct a noun object.

Now looking at the [NP] node (Noun Phrase), it has one child, [NNP]. Since there is no article we can for now assume the statement is referring to a specific [NNP] (Jones), which is important for matching purposes later on. The name of this noun object is 'Jones'. We know that there is only one 'Jones' that is being talked about as well. At this point, we have placed into the temporary ontology an instance of a 'Jones', where the quantity is only 1 and that it contains no attributes at this point in time (or at all in this example).

Next, we look at the [VP] node. The [VP] node contains two children, both a [VB] node (Verb, base form), and a [NP] node. We look at the verb 'sign' and we begin to build another relationship from this. While we haven't taken a look at what the verb actually is at this current point in time, we do have it stored so we can reference it. We know that it is a present tense verb and uses 'system time' which is as previously mentioned refers to a counter in which statements or key points have been processed. Since this is the systems first premise the counter is 0. So we know this particular verb occurs in present time at time t = 0. While this did occur in the past, as Smith saw it happen at some point in time prior, at that point in time he saw it at the present time. To see if Jones signed the contract, it still must be in the past, as the parent verb still must be satisfied (which includes the time component).

We now have the final noun object for this verb phrase to process. We have a [NP], a noun phrase, to process. In the [NP], there are two children [DT] (article), and [NN] (generic noun singular). We create a new noun object. The system now looks at [DT], sees that it is 'the'. From that it knows that the noun object is referring to a specific object. It checks to see if there is a reference to this object, which must be satisfied by both name and time constraints. It is possible to be discussing several different contracts as they can exist at different points in time. The node [NN] under [NP] is now examined by the system, which returns a 'contract', which is a generic singular noun. So we combine the two children together under one noun object, which tells us we have a single specific contract. This noun object is also added to the temporary ontology.

Now that the verb object has been completed through parsing successfully so far, and there were no problems detected with the information presented, the system finishes the relationship connecting the instance of 'contract' to the instance of Jones via the word 'sign'.

Now that the [S] node has been completed that is a child of the highest level [VP] node, we can return back to that [VP] node to finish processing it. We attach the newly created statement to this verb object and apply an attitude modifier to each child within it. Effectively this is a reference to both when in time and also a confidence value if you will, or maybe another way of saying a truth value (though the later would decidedly depend on how much you trust the

individual or source providing you with the information which is outside the scope of this paper). Now we look at the verb 'saw' and look up its confidence value, which is a value of 1. Figure I-1 depicts how attitudes are stored, effectively as another attribute with a pointer (reference) to the predicate.

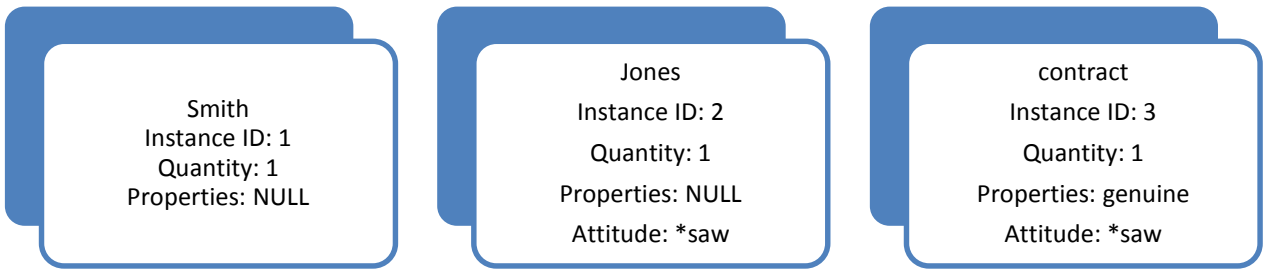
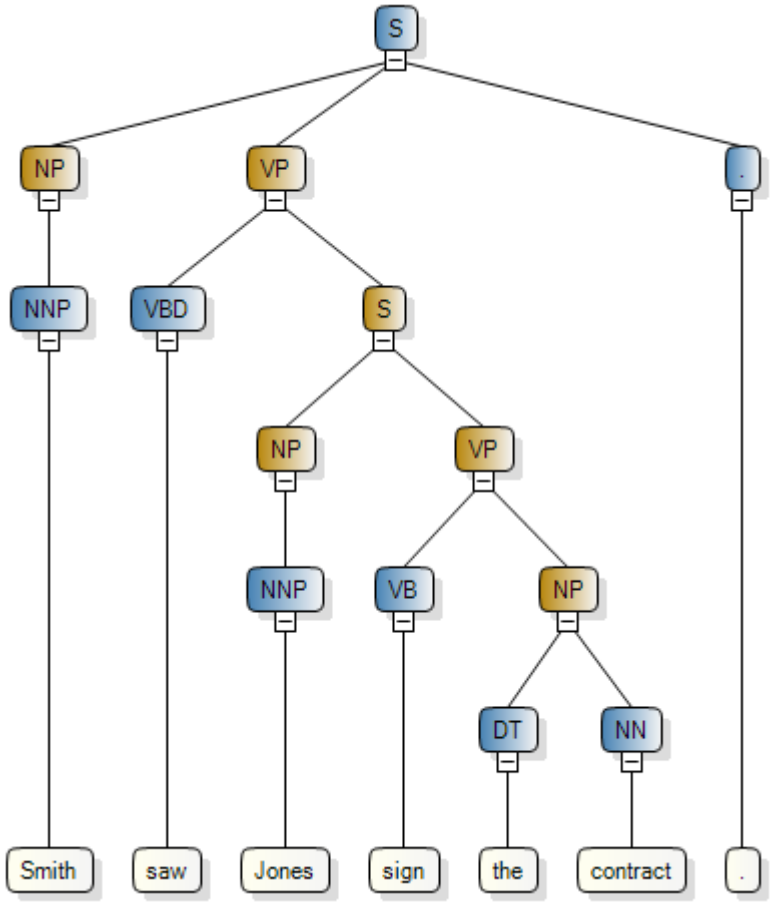
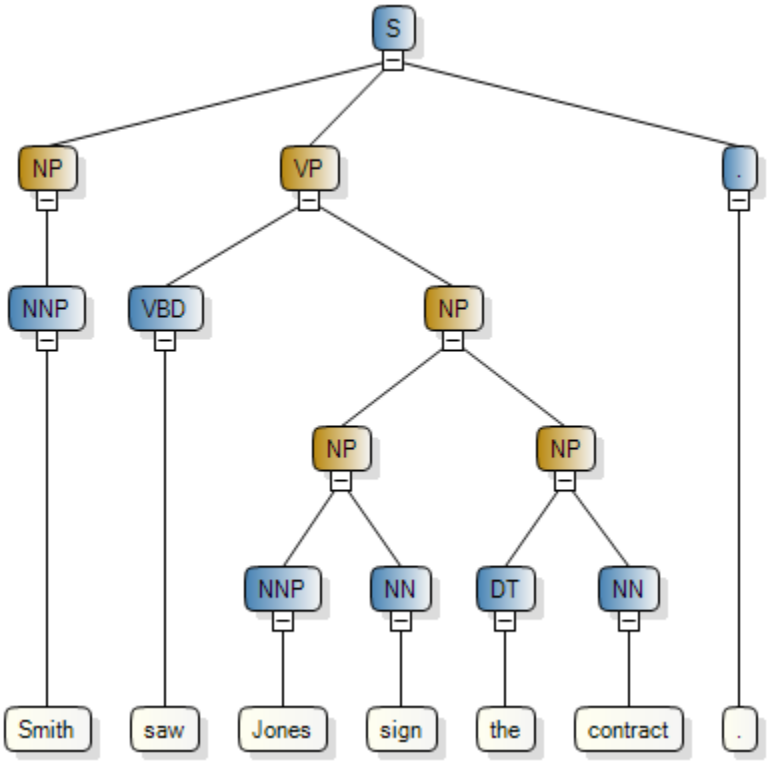


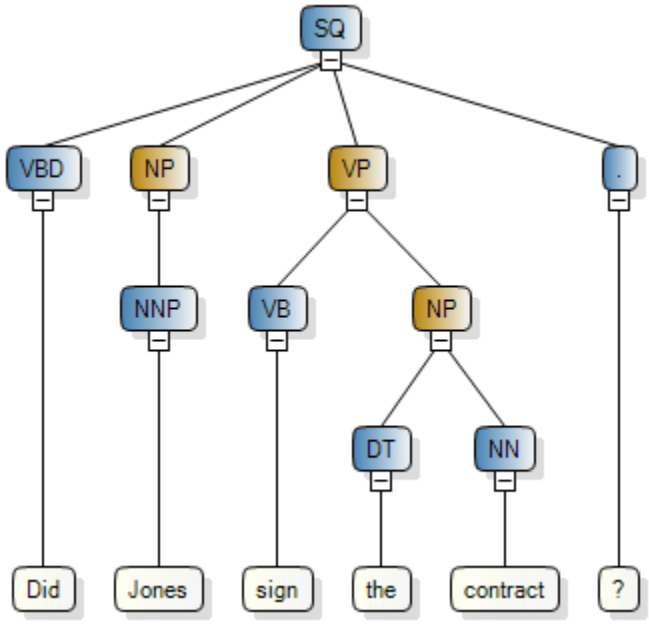
Figure I-1 Temporary Ontology for Problem 342 P1



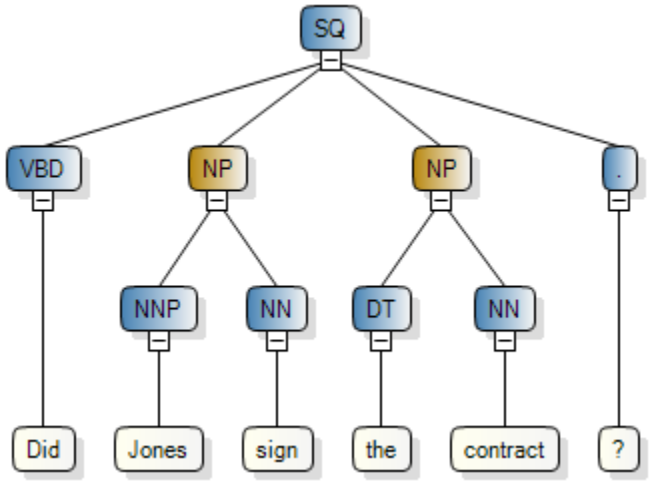
Parse P1 (1 of 2)



Parse P1 (2 of 2)



Parse Q (1 of 2)



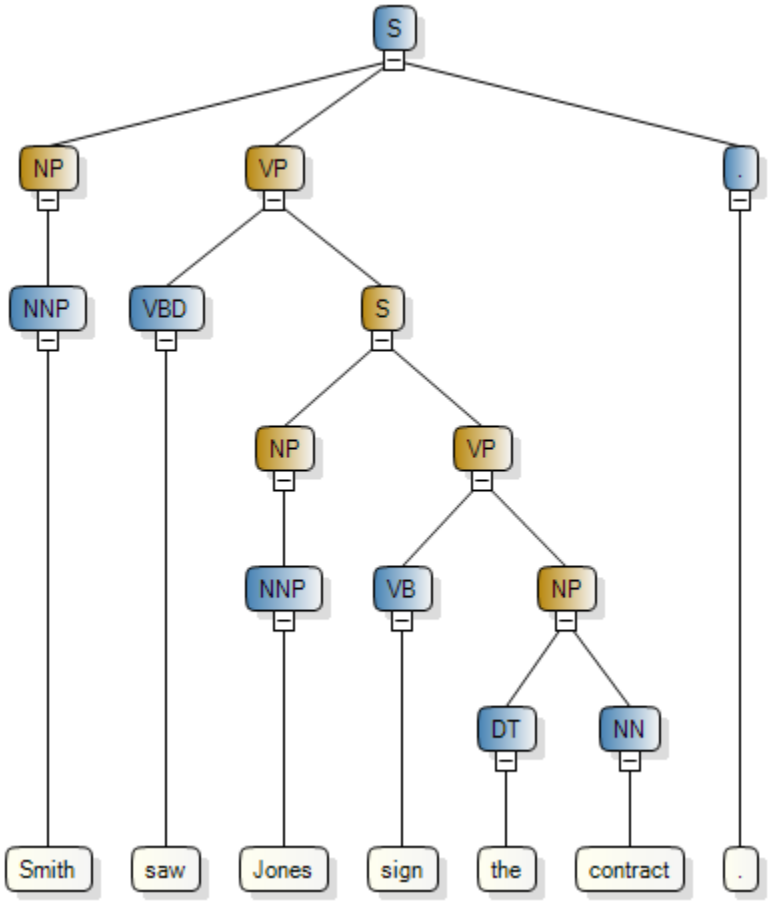
Parse Q (2 of 2)

Section 9.2.3 - Problem 343 from FraCaS Test Suite

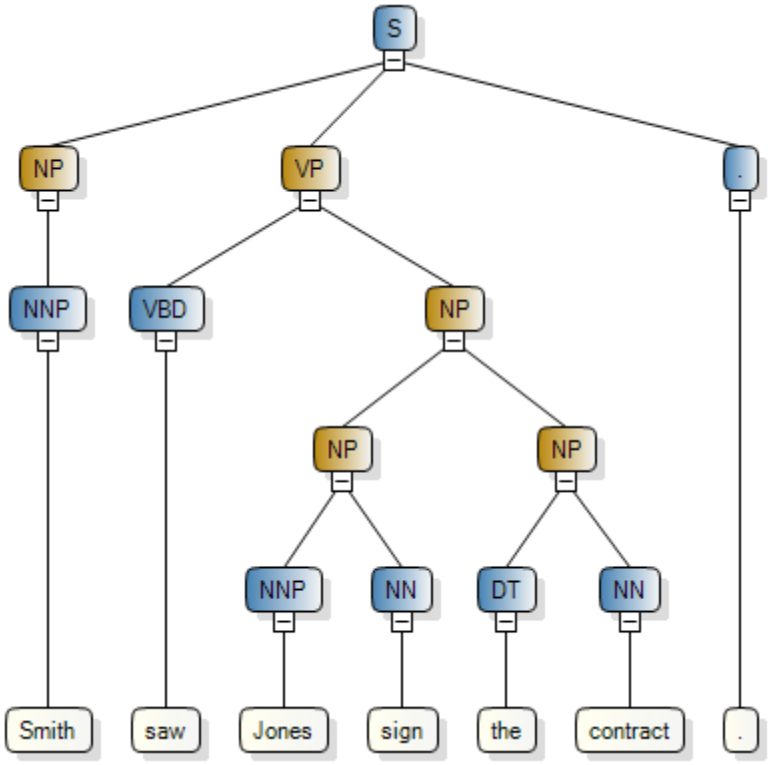
Problem 343 comes from section 9.2.3 of the FraCaS Test Suite. This particular section focuses on Substitution Attitudes.

Table I-2. Problem 343 from Section 9.2.3 - FraCaS Test Suite

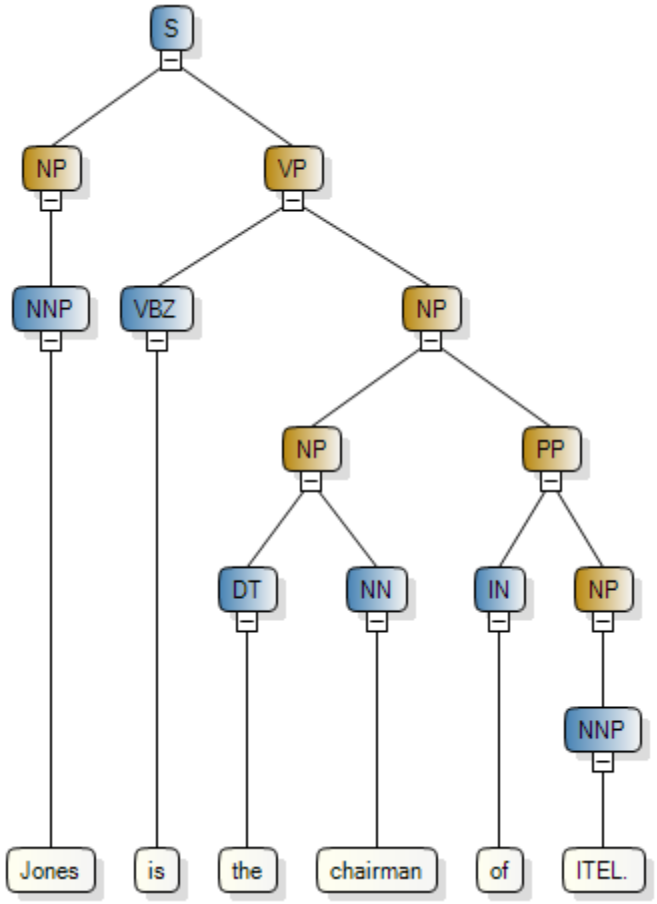
P1	Smith saw Jones sign the contract.
P2	Jones is the chairman of ITEL.
Q	Did Smith see the chairman of ITEL sign the contract?



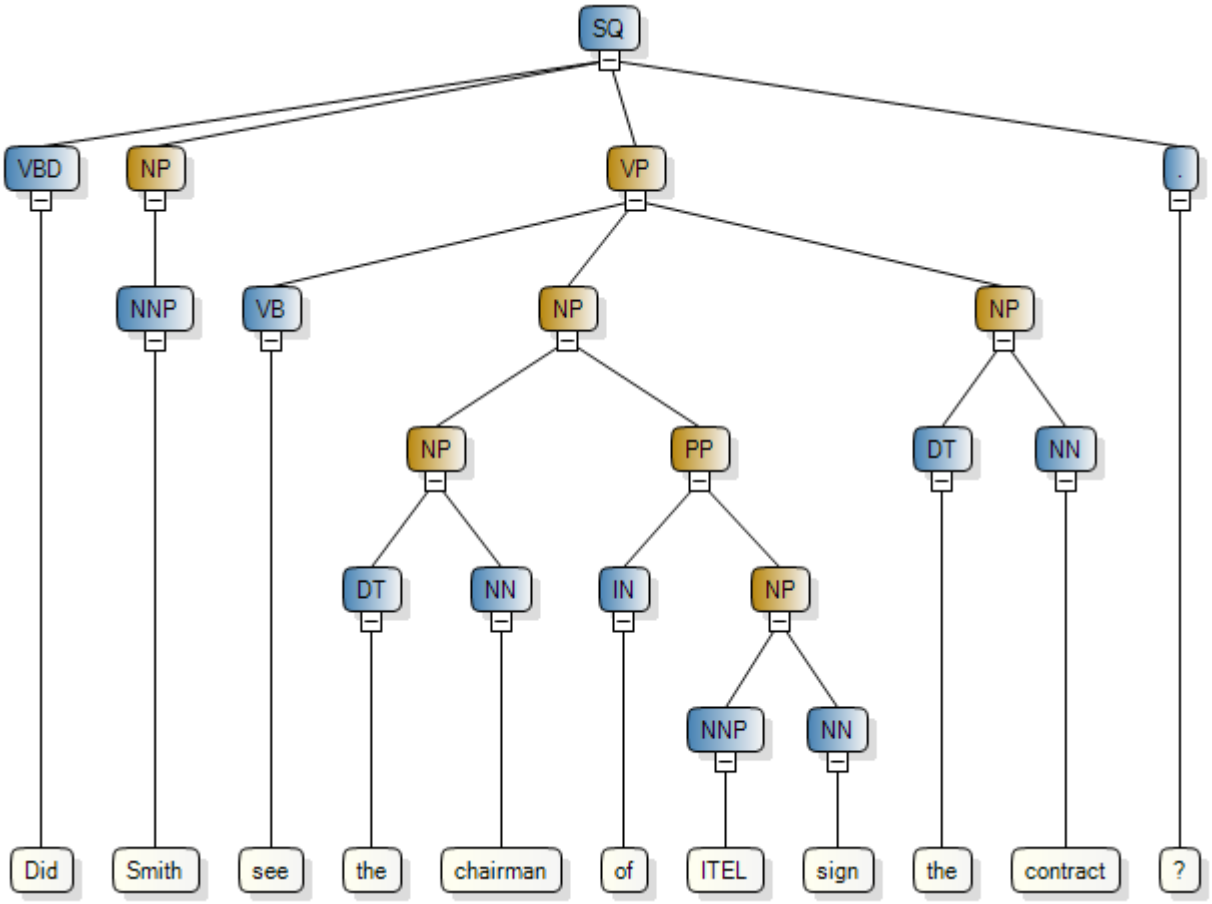
Parse P1 (1 of 2)



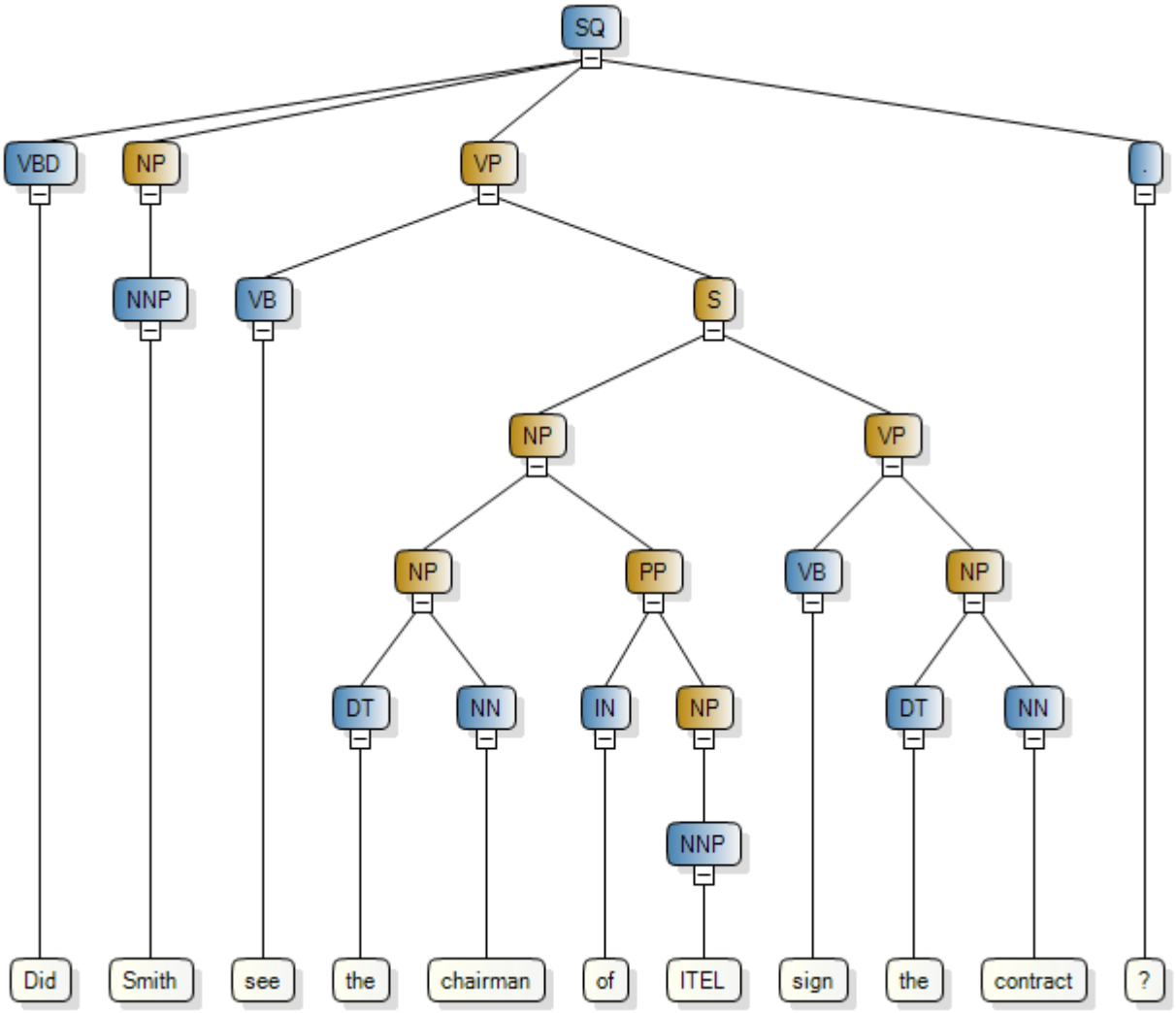
Parse P1 (2 of 2)



Parse P2 (2 of 2)


SQ
VBD
NP
VP
.
NNP
VB
NP
NP
NP
PP
DT
NN
DT
NN
IN
NP
NNP
NN
Did
Smith
see
the
chairman
of
ITEL
sign
the
contract
?

Parse Q (1 of 2)


SQ
VBD
NP
VP
.
NNP
VB
S
NP
VP
NP
PP
VB
NP
DT
NN
IN
NP
DT
NN
NNP
Did
Smith
see
the
chairman
of
ITEL
sign
the
contract
?

Parse Q (2 of 2)