Accommodating Complex Chained Prepositional Phrases in Natural Language Query Interface to an Event-Based Triplestore

Elham Emami
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/5254

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
Accommodating Complex Chained Prepositional Phrases in Natural Language Query Interface to an Event-Based Triplestore

By:

Elham Emami

A Thesis
Submitted to the Faculty of Graduate Studies and Research through the School of Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Science at the University of Windsor

Windsor, Ontario, Canada
2015

© 2015 Elham Emami
Accommodating Complex Chained Prepositional Phrases in Natural Language Query Interface to an Event-Based Triplestore

By:

Elham Emami

APPROVED BY:

______________________________
Dr. D. Martinovic,
Faculty of Education and Academic Development

______________________________
Dr. J. Lu
School of Computer Science

______________________________
Dr. R. Frost, Advisor
School of Computer Science

28-January-2015
DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.
Abstract:

Building Natural language query interfaces (NLI) to databases is one the most interesting and challenging fields of study for computer scientists and researchers. There have been many advancements and achievements in this area that enables NLIs to operate more efficiently and have wide NL coverage. However, there exists some shortcomings in query interface to semantic web triplestores. Some researchers have attempted to extend the range of queries that can be answered. However, only a few techniques can handle queries containing complex chained prepositional phrases. This thesis involves extending an existing method that can accommodate prepositional phrases to also be able to handle “when…,” “where…,” and “with what…” type queries. The approach developed is implemented in the Miranda programing environment.

[Keywords: natural language interface, prepositional phrases, tripestore database]
Acknowledgements

I would like to thank my supervisor Dr. R. A. Frost for his valuable suggestions, remarks, comments, discussion and encouragement.

I would also like to thank Dr. Dragana Martinovic and Dr. Jianguo Lu for reading my thesis report and for their valuable comments.

And special thanks to my lovely parents for their encouragement and support.
Table of Contents

DECLARATION OF ORIGINALITY ........................................................................... iii
Abstract ....................................................................................................................... iv
Acknowledgements ..................................................................................................... v
List of Tables ............................................................................................................... viii
List of Figures ............................................................................................................. viii
Chapter 1: PREFACE ................................................................................................. 1
  1.2 Outline of report.................................................................................................. 1
Chapter 2: INTRODUCTION ....................................................................................... 3
  2.1. Introduction....................................................................................................... 3
  2.2 Montague’s Approach to Natural Language....................................................... 5
  2.3 Problem Addressed ........................................................................................... 6
  2.4 Thesis Statement ............................................................................................... 6
  2.5 Why the thesis is important ............................................................................. 7
  2.6 How the thesis will be proven .......................................................................... 7
Chapter 3: Montague Semantics ................................................................................. 8
  3.1 Subject-Predicate and Determiner-Noun rules: ............................................. 11
  3.2 Conjoined sentences, Verb phrases and Term phrases ................................ 14
  3.3 Anaphoric Pronouns as bound variables: ...................................................... 16
  3.4 Transitive verbs, Meaning Postulates and Non-Specific Readings: ................ 17
  3.5 Adverbs ............................................................................................................ 19
  3.6 Negation ............................................................................................................ 20
Chapter 4: Related work on prepositional phrases and quantifiers ....................... 21
  4.1. Montague’s approach to quantifiers ............................................................... 21
  4.2 Improved Montague Semantic by Event approach .......................................... 22
  4.3 Other researcher’s approaches to accommodate the prepositional phrase ........ 29
Chapter 5: The specific problem addressed ............................................................. 32
Chapter 6: The new idea ............................................................................................ 35
  6.1. Summary of the new idea ............................................................................. 35
  6.2. Statement and Contribution ......................................................................... 36
Chapter 7: Implementation of the new idea ............................................................. 39
Chapter 8: ANALYSIS/EXPERIMENTS CONDUCTED ....................................... 49
Chapter 9: Claim and Conclusion ............................................................................ 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Claim</td>
<td>53</td>
</tr>
<tr>
<td>9.2</td>
<td>What has been achieved?</td>
<td>53</td>
</tr>
<tr>
<td>9.3</td>
<td>Relationships to parsing and access to triplestores</td>
<td>54</td>
</tr>
<tr>
<td>9.4</td>
<td>Contribution to Computational Linguistics and Computer Science</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>REFERENCES:</td>
<td>56</td>
</tr>
<tr>
<td>Appendix I</td>
<td>Program</td>
<td>58</td>
</tr>
<tr>
<td>Appendix II</td>
<td>Survey</td>
<td>65</td>
</tr>
<tr>
<td>Appendix III</td>
<td>Annotated Bibliography</td>
<td>100</td>
</tr>
<tr>
<td>VITA AUCTORIS</td>
<td>139</td>
<td></td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1 ........................................................................................................... 11
Table 2 ........................................................................................................... 30

LIST OF FIGURES

Figure 1 ........................................................................................................... 12
Figure 2 ........................................................................................................... 34
Figure 3 ........................................................................................................... 51
1.1 Introduction
This report describes a new approach for the accommodation of prepositional phrases in natural language interfaces to triplestore database and develops an event-based denotational semantics that can be used for natural-language (NL) query interfaces to triplestores. This topic is significant because the usage of triplestores databases in the semantic web, and natural language interfaces to them, is growing rapidly and are now extensive and popular. A wide range of queries can be answered by a proposed method using a denotational semantics that accommodates prepositional phrases in natural language queries to triplestores.

1.2 Outline of report
This report shows how to accommodate prepositional phrases in natural language interfaces and handle queries that ask for temporal and spatial details and contain “when..?,” “where…?,” and “with what…?” type questions. Chapter 2 of the report contains an introduction to Montague Semantic, the thesis statement and the main motivation of it. Chapter 3 is about the use of Montague Semantics in database query processing. Chapter 4 describes previous work that has studied prepositional phrase
accommodation in natural language queries. The specific problem addressed is described in Chapter 5 and the new idea is introduced in Chapter 6. Chapter 7 describes the implementation of new idea. The analysis and experiments are covered in Chapter 8. Chapter 9 contains the claims and conclusions. A comprehensive survey on accommodating prepositional phrase in natural language queries to triplestore databases is attached as an appendix at the end of the report.
Chapter 2: INTRODUCTION

2.1. Introduction

A Database is a collection of data that is organized to process future accessibility. A database management system is a collection of programs that enables users to create and maintain the database system for access in the future. There are different models to store data, such as the relational database model, in which data is stored as records and fields in rows and columns of tables. Data can be related to each other by defining relationships between tables. Another popular method of storing data that is studied in this thesis is the triplestore method. A Triple consists of a “subject”, a “predicate” and an “object”. The triplestore has some advantages compared with relational stores. Although the relational model is more mature and popular than the triplestore model, it suffers some limitations and deficiencies. A major aspect of the relational model is that it uses a schema for its organization, thus, the designer or database administrator has the responsibility to recognize what type of questions will be asked in advance and builds a schema based on that. Therefore, the relational data has to have schema for its organization. Problems show up when the database schema needs to be changed because the relational database is not flexible and its schema remains static while the stored data type is not known in advance in some applications, like criminal investigations. In order to provide continuous changes to apply to database schema and queries, some researchers believe that it would be easier
if the data is represented in binary relations or graphs and stored in triplestores rather than conventional relational databases. Because triplestores do not have the deficiency of a fixed schema, they are highly flexible. Most data in the Semantic Web is stored in Resource Description Framework (RDF) triplestore format. A triple includes a subject, a predicate and an object. The subject is the resource, the predicate denotes the property and the object is the value of the property that can be another resource. Values of subjects, predicates and the objects are denoted by Uniform Resources Identifiers (URIs), which are similar to URLs used in the web to ensure that their values are unique for the future access. For example the fact “Al Capone was married to Mae Capone” can be represented by the following triple.

<http://dbpedia.org/resource#Al_Capone>
  <http://dbpedia.org/ontology/spouse>
  <http://dbpedia.org/resource#Mae_Capone>

Adding more details regarding the time and place of the marriage, is possible by adding more triples to existing triples. However, it is not always sufficient to do this because it might be that Capone married more than once and therefore adding another triple simply stating the time of the marriage could lead to ambiguity about which marriage took place at that time. The following discusses this problem and provides more detail regarding the solutions of the issue. Triples are considered as one of the basic units of languages like RDF, RDFS and OWL, which can enhance interface capabilities. These languages are used to represent data in the the semantic web.

The role of databases has grown dramatically with advancements in data storage. An example triplestore is DBpedia, which contains 2.5 billion facts extracted from Wikipedia.

Also, technology experts have made successful efforts to provide more convenient interaction for users with computers and machines. This type of interaction can be done by typing or speaking and does not need users to know low level or machine languages. It can also send their requests and information via talking or typing. That facility has been provided by natural language interfaces by which users enter their requests and queries in their own language and the related response will be retrieved for them. There are some
advantages to providing a natural language interface for users to connect with databases. The natural language interfaces to database allow people to communicate with database in the same way they communicate with each other. The main advantages of natural language interface is that users are not required to learn queries languages like SQL because they are difficult for non-experts to learn and use. Moreover, there are some type of questions, specifically those containing negation and quantification, that can be expressed easily in natural language rather than formal query languages such as SQL.

One approach that has made considerable progress in answering natural language queries to triplestore data has been developed by Frost et al. [2014]. The approach is based on an efficient version of Montague Semantics. After doing a survey and study in natural language queries to the triplestore database, it was realized that no existing method can accommodate prepositional phrases, such as “who stole a car in Manhattan in 1980?,” as referred to by Frost et al. [2013] and Frost et al. [2014]. However, though a method suggested approach by Frost et al. [2014] can handle chained complex prepositional phrases, it cannot accommodate “when…,” “where…,” or “with what…,” type questions.

2.2 Montague’s Approach to Natural Language

In the 1970s, Richard Montague developed an approach to the interpretation of natural language in which he claimed that we could precisely define the syntax and semantics for substantial sub-sets of natural languages such as English. He described the aim of his enterprise as follows:

“The basic aim of semantics is to characterize the notion of a true sentence (under a given interpretation) and of entailment” Montague [1970].

The salient points of Montague's approach are a model of theoretical semantics, a systematic relation between syntax and semantics, and a fully explicit description of a fragment of natural language. His approach constituted a revolution after the Chomsky revolution that brought mathematical methods into syntax. Richard Montague was a mathematical logician who had specialized in set theory and modal logic. In Montague's view, the study of natural language belonged to mathematics, not to psychology, Janssen [2012]. Montague [1970] formulated his understanding of semantics as follows:
“There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed I consider it possible to comprehend the syntax and semantics of both kinds of languages with a single natural and mathematically precise theory.”

In Montague’s semantics, all things can be depicted as sets of entities in the universe of discourse. He claimed that each object in a phrase of natural language can be interpreted as a function. Montague claimed that there is a correspondence between each syntactic category of natural language and each semantic type in the universe of discourse, and he also believed that there is correspondence between syntactic rules that explain how complex syntactic phrases can be built from simpler ones. Similarly, the semantic rules that explain how complex semantic functions that are the meanings of complex phrases can be built from the meanings of simpler phrases. Montague was one of the first to develop a compositional semantics for a substantial part of English. The next chapter provides more details and description regarding Montague’s semantics.

2.3 Problem Addressed

To address the main problem of natural language interface to triplestore, there is no method to accommodate chained prepositional phrases in natural language interfaces that can also handle questions type “when..,” “Where…and,” “With what..”.

2.4 Thesis Statement

It is possible to construct natural language query interfaces to triplestores which can accommodate arbitrarily-nested complex chained prepositional queries such as:

- “When did Capone steal a car in Manhattan using a crowbar…?”
- “Where did Capone steal a car in 1918 or 1920…?”
- “With what did Capone steal two cars and a truck that was owned by a gangster who joined every gang in Brooklyn that was joined by Torrio?”
2.5 Why the thesis is important

This thesis is important because proving it will enable the extension of the coverage of queries about spatial and temporal details of stored facts in triplestore databases and provide an approach to deal with queries such as “When did Capone marry?,” “Where was the red car stolen in 1905?” and “With what did Hall discover Phobos in 1187?”.

2.6 How the thesis will be proven

In order to prove this thesis, a new semantics that manipulate events rather than entities was developed. This required the most significant denotations in the method of Frost et al. [2013] to be changed. The new semantics was implemented in Miranda with an in-program triplestore. The advantages of using Miranda as the programing language include:

- It has built-in list operators and a list comprehension construct which correspond to the “relative set notation” that used in denotations of transitive verbs.
- The higher-order functions used in the denotations can be defined directly in Miranda.
- Given the declarative nature of Miranda, the definitions are executable specifications which allow us to test our ideas.

Also the program was tested with a set of queries with different types of language constructs.
Chapter 3: Montague Semantics

To analyze the meaning of natural language expressions, a semantics of them has to be modeled theoretically. In two papers written by Montague in 1971 “Universal Grammar” and “The proper Treatment of Quantifications in ordinary English” PTQ, M0ntague introduced his novel ideas about language. Richard Montague was a logician and philosopher who had done significant and remarkable work on language based on the theory known after his death as Montague grammar. Montague grammar is a new strating point for studying formal semantics, Partee et al. [1930].

Based on paper by Partee et al., Montague was born September 20, 1930 in California and died March 7, 1971 in Los Angeles.

He studied journalism in Junior College, he joined the University of California, Berkeley in 1948, and started studying in mathematics, philosophy, and Semitic languages, graduating with a B.A in Philosophy in 1950. He continued graduate work at Berkeley in all three areas, receiving an M.A. in mathematics in 1953 and his Ph.D. in Philosophy in 1957. Montague quickly turned into a known person in mathematical logic, with contributions to proof theory, model theory, axiomatic set theory, and recursion theory. He could apply logical methods to a different of issues in philosophy, such as the philosophy of language. Montague’s work on semantics was very significant and important for
linguistics. Building on his development of a high order typed intentional logic with a possible-worlds model theoretic semantics and a formal pragmatics incorporating indexical pronouns and tenses, Montague turned in the late 1960’s to the project of “Universal Grammar”.

The following is based on Partee et al. [1997]: Montague had attempted to develop a philosophically satisfactory and logically accurate form of syntax, semantics and pragmatics that exist in both formal and natural language. There were controversies amongst linguistics about Montague’s ideas because some of them believed that natural language such as English is not suitable to accurate formalize. Also, many linguistics were not sure about the appropriateness of logicians’ approaches to the domain of natural language semantics.

“The proper Treatment of Quantifications in ordinary English” had the most impact on linguists and also on the developing of formal semantics which had been used by Montague in 1973.

As the author in the paper referred to by Partee et al., Montague grammar refers to PTQ and its extensions most the time by linguistics and philosophers but it contains an algebraic framework of “Universal Grammar” that constitutes Montague’s theory of grammar. The strength of Montague proposal was truly decisive for the compositional semantic interpretation of syntax structure. That is demonstrated in PTQ in which higher-order logics that accompanied lambda abstraction made it possible to interpret noun phrases such as “every man”, “the man”, “a man” as semantics components. PTQ also contains a pioneering approach for quantifiers and transitive verbs, conjunctions, adverbial modifiers and more.

Montague grammar is a grammar for a natural language which includes three components: syntax, a syntactic analysis of the expressions and translation of natural language into a logical language, and the semantics that interpretation model to express logical language. Lambda calculus is required in order to express Montague semantics. Lambda calculus can be thought of as a compact programming language. The \( \lambda \) calculus consists of transformation rules and variable substitution in the other terms and a function definition scheme. It was introduced by Alonzo Church [1930] as a formalized way for the concept of computability.
The λ calculus is highly powerful method because any computable function can be represented and evaluated by applying this formalism. Some researchers and linguistics believe that it is similar to Turing machines although it emphasizes the use of transformation and does not pay attention to machine implementation.

Church et al. [1941] realized that each expression in natural language which contains …...x…. can be defined as function of x. He proposed λx as name for the function. The rule for λ conversion is as follow:

\[\lambda x \ [\ldots x \ldots] (\alpha) \Rightarrow [\ldots \alpha \ldots]\]

In which, each x is replaced by α, so the result would be [……α……].

Montague Grammar and semantics is described in detail in the book titled “Montague Semantics” by Dowty et al. [1981]. Montague used some examples to explain more about the semantics of NL expressions by using lambda. The following two example a. The translation of “Every man eats” and “Every man sleeps” in predicate logic would be:

\[\forall x(M(x) \rightarrow E(x)) \text{ and } \forall x(M(x) \rightarrow S(x))\]

The general format of the above sentences is the following:

\[\lambda Y[\forall x(M(x) \rightarrow Y(x))]\]

Which for first sentence it would be:

\[\lambda Y[\forall x(M(x) \rightarrow Y(x))](E)\]

And for the second one, the conversion would be:

\[\lambda Y[\forall x(M(x) \rightarrow Y(x))](S)\]

There is difference between sense and reference of expression in Montague theory. The sense or intension of expression relates to the meaning of expression but the reference or extension of expression considers the semantics of the expression. According to the rules of PTQ grammar α^i shows the intension of expression of α and α^e denotes the extension of expression of α. Montague also introduced another rule which is called the cancellation rule. It is denoted by ^i and is important in simplification of expression produced by translation of sentence in natural language. The determinants “every” and “a” denote the following expressions:

Every => \[\lambda P[\lambda Q[\forall x(Pe(x) \rightarrow Qe(x))]]\]

A => \[\lambda P[\lambda Q[\exists x(Pe(x) \wedge Qe(x))]]\]
There are syntactic categories which are used in the PTQ grammar:

<table>
<thead>
<tr>
<th>CATEGORY NAME</th>
<th>DEFINITION OF THE CATEGORY NAME</th>
<th>EQUIVALENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>CN</td>
<td>Common Noun</td>
</tr>
<tr>
<td>TV</td>
<td>IV/T (≡IV/(t/IV))</td>
<td>Transitive Verb</td>
</tr>
<tr>
<td>IV</td>
<td>IV</td>
<td>Intransitive verb</td>
</tr>
<tr>
<td>T</td>
<td>t/IV</td>
<td>Term Phrases and Proper Name</td>
</tr>
<tr>
<td>IAV</td>
<td>IV/IV</td>
<td>Intransitive Adverb</td>
</tr>
<tr>
<td>IAV/T</td>
<td>(IV/IV)/T</td>
<td>Preposition</td>
</tr>
<tr>
<td>T/CN</td>
<td>(t/IV)/CN</td>
<td>Determiner</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>Sentence</td>
</tr>
<tr>
<td>T/T</td>
<td>t/t</td>
<td>Sentence Adverb</td>
</tr>
<tr>
<td>IV/T</td>
<td>IV/t</td>
<td>Sentence Complement Adverb</td>
</tr>
<tr>
<td>IV/IV</td>
<td>IV/IV</td>
<td>Infinitive Complement adverb</td>
</tr>
</tbody>
</table>

Table 1. Categories used in PTQ grammar

3.1 Subject-Predicate and Determiner-Noun rules:

The set of basic expressions of category of A is being stated by $B_A$ that can be specified as simply as “lexical entries” of each category. The set of phrases of category A can be denoted by $P_A$ that includes the basic expressions for each category together and complicated expression of that category that are formed by syntactic rules. The first syntactic rule of PTQ rules is related to basic expressions:

$$S_1. \quad B_A \subseteq P_A \text{ for every category } A.$$
The rest of the rules take care of more complicated expressions which all of them include three major type of information: (1) category or categories of expression that are used as input of the syntactic rule, (2) the category of expression which is denoted as output of syntactic rule and is result of applying function on the input categories, (3) the operational structure of expressions that describe how the input categories turns to output one.

The general form of syntactic rules in PTQ grammar is as follow:

\[ S_n : \text{If } \alpha \in \text{PA and } \beta \in \text{PB then } F_n(\alpha, \beta) \in \text{PC, where } F_n(\alpha, \beta) \text{ is………} \]

A and B state the syntactic categories of input expression and C is syntactic category of output which is the result of function \( F_n \), \( n \) demonstrate the number of the rule and \( F_n \) is the name of rule in the PTQ grammar. Some of the PTQ grammar which is required to understand Montague semantic has been provided as follows:

Consider \( S_4 \) as example of syntactic rule which is called “subject-predicate” rule.

\[ S_4 : \text{If } \alpha \in \text{PT and } \beta \in \text{PIV then } F_4(\alpha, \beta) \in \text{P_I, where } F_4(\alpha, \beta) = \alpha \beta', \text{ and } \beta' \text{ is the result of replacing the first verb in } \beta \text{ by its third person singular present form.} \]

This rule takes the term phrase which is member of \( \text{PT} \) and \( \text{PIV} \) member of verb phrase then forms a sentence by combing them that is member of \( \text{P_I} \). In order to explain more about the operation of this rule take the following example: Jack \( \in \text{B_I} \) so expectedly refer to \( S_1 \) we have Jack \( \in \text{P_I} \), walk \( \in \text{BIV} \) so we have walk \( \in \text{PIV} \) thus based on \( S_4 \) Jack walks \( \in \text{P_I} \).

Sentence or phrases which are constructed syntactically can be demonstrated through an analysis tree. The analysis tree for “Jack walks” is as shown below.

```
Jack walks
   /   \
  Jack  walk
```

Figure 1

All the terminal nodes of the analysis tree are basic expressions. The translation rule related to \( S_4 \) is as follow:
T4: If $\alpha \in \Pr$ and $\beta \in \Pv$ and $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, then $F_4(\alpha, \beta)$ translates into $\alpha'(\beta')$. The translation of “Jack talks” is as follow:

**Jack** translates into $\lambda P [P_e(j)]$

**Jack talks**

1. Jack $\Rightarrow \lambda P [P_e(j)]$
2. talks $\Rightarrow$ talk'
3. $\lambda P [P_e(j)](\text{talk')}$ [From 1,2 by T4]
4. talk'(j) [Lambda conversion]
5. talk'(j) [Cancellation rule]

It should be mentioned that j is an individual constant corresponding to the person called Jack.

The S2 rule relates the “Determiner-Noun rule” which combines determiner such as “every” and “a” to produce partial sentences like: “every color” or “a car”. The format of S2 rule is as follow:

S2: If $\alpha \in \PrCN$ and $\beta \in \Pn$ then $F_2(\alpha, \beta) \in \Pr$, where $F_2(\alpha, \beta) = \alpha \cdot \beta$ and $\alpha'$ is $\alpha$ except in the case where $\alpha$ is a and the first word in $\beta$ begins with a vowel, here $\alpha'$ is an. The translation for rule S2 is:

T2: If $\alpha \in \PrCN$ and $\beta \in \Pn$ and $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, then $F_2(\alpha, \beta)$ translates into $\alpha'(\beta')$.

The words **every**, **the** and **a** can be translated as follows according to Montague:

**every** translates into: $\lambda P \ [\lambda Q \ \forall x \ [P_e(x) \rightarrow Q_e(x)]]$

**the** translates into: $\lambda P \ [\lambda Q \ \exists x \ [\forall x \ [P_e(x) \leftrightarrow x = y] \ \wedge \ Q_e(y)]]$

**a** translates into: $\lambda P \ [\lambda Q \ \exists x \ [P_e(x) \wedge \ Q_e(x)]]$

The translation of the sentence “Every dog runs” is like below by applying T2 and T4.

1. every $\Rightarrow \lambda P [\lambda Q \ \forall x [P_e(x) \rightarrow Q_e(x)]]$
2. dog $\Rightarrow$ dog'
3. every dog $\Rightarrow \lambda P [\lambda Q \ \forall x [P_e(x) \rightarrow Q_e(x)](d o g')]$

[From 1,2 by T2]
4. $\lambda Q \ \forall x [\text{dog}^{\sim, i}(x) \rightarrow Q_e(x)]$ [Lambda conversion]
5. $\lambda Q \ \forall x [\text{dog}^{\sim, i}(x) \rightarrow Q_e(x)]$ [Cancellation Rule]
6. runs $\Rightarrow$ runs'
7. every dog runs $\Rightarrow \lambda Q \ \forall x [\text{dog}'(x) \rightarrow Q_e(x)]$(runs')
Based on T2 and T4 the following translation can be as:

“**The sun shines**”
\[ \exists x \left[ \forall x \left[ \text{sun'}(x) \iff x = y \right] \land \text{shines'}(y) \right] \]

“**A dog barks**”
\[ \exists x \left[ \text{dog'}(x) \land \text{barks'}(x) \right] \]

### 3.2 Conjoined sentences, Verb phrases and Term phrases

The first conjunction rule in PTQ grammar which combines two sentences and combine them is called S11a and produce a new sentences by assigning rule S11. Take two following sentences as example for considering rule S11. “Jack talks” and “every dog runs” are two sentences which can be conjoined by the rule S11a and the new sentence “Jack talks and every dog runs” can be produced by applying S11. The rule is as follows:

**S11a:** If \( \alpha, \beta \in \text{Pt} \), then \( F_{11a}(\alpha, \beta) \in \text{Pt} \) where \( F_{11a}(\alpha, \beta) = \alpha \land \beta \)

The associated translation rule of S11a is T11a which is:

**T11a:** If \( \alpha, \beta \in \text{Pt} \) and \( \alpha, \beta \) translate into \( \alpha' \), \( \beta' \) respectively, then \( F_{11a}(\alpha, \beta) \) translates into \( \left[ \alpha' \land \beta' \right] \).

By helping T11a, the sentence “Jack runs and every dog runs” can be translated to \( \left[ \text{talk'}(j) \land \forall x \left[ \text{dog'}(x) \rightarrow \text{runs'}(x) \right] \right] \) which **Jack talks** can be translated into \( \text{talk'}(j) \) and **every dog runs** can be translated into \( \forall x \left[ \text{dog'}(x) \rightarrow \text{runs'}(x) \right] \).

The rule S11b define as follow:

**S11b:** If \( \alpha, \beta \in \text{Pt} \), then \( F_{11b}(\alpha, \beta) \in \text{Pt} \), where \( F_{11b}(\alpha, \beta) = \alpha \lor \beta \)

The translation rule for S11b would be T11b, which is:

**T11b:** If \( \alpha, \beta \in \text{Pt} \) and \( \alpha, \beta \) translate into \( \alpha' \), \( \beta' \) respectively, then \( F_{11b}(\alpha, \beta) \) translates into \( \left[ \alpha' \lor \beta' \right] \)

There are three more conjunction rule which are S12a, S12b, S13 that S12a and S12b take two verb phrases and produce a sentence but S13 take two term phrase which is
member of Pt and combine together to make a meaningful sentence. Here the rules S12a, S12b and S13 are defined:

**S12a:** If $\alpha, \beta \in P_{IV}$, then $F_{12a}(\alpha, \beta) \in P_{IV}$, where $F_{12a}(\alpha, \beta) = \alpha$ and $\beta$

**S12b:** If $\alpha, \beta \in P_{IV}$, then $F_{12b}(\alpha, \beta) \in P_{IV}$, where $F_{12b}(\alpha, \beta) = \alpha$ or $\beta$

**S13:** If $\alpha, \beta \in P_{IV}$, then $F_{13}(\alpha, \beta) \in P_{IV}$, where $F_{13}(\alpha, \beta) = \alpha$ or $\beta$

The associated translation with above rule are:

**T12a:** If $\alpha, \beta \in P_{IV}$ and $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, then $F_{12a}(\alpha, \beta)$ translates into $\lambda x [(\alpha')(x) \land (\beta')(x)]$

**T12b:** If $\alpha, \beta \in P_{IV}$ and $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, then $F_{12b}(\alpha, \beta)$ translates into $\lambda x [(\alpha')(x) \lor (\beta')(x)]$

**T13:** If $\alpha, \beta \in P_{IV}$ and $\alpha, \beta$ translate into $\alpha', \beta'$ respectively, then $F_{13}(\alpha, \beta)$ translates into $\lambda P [(\alpha')(P) \lor (\beta')(P)]$. The translation of sentence “Every human walks or talks” by using T12b can be:

Every human walks or talks.

1. every human $\Rightarrow \lambda Q \forall x [human'(x) \rightarrow Qe(x)]$
   [Derived in section 2]

2. walk $\Rightarrow$ walk'

3. talk $\Rightarrow$ talk'

4. walk or talk $\Rightarrow \lambda x [walk'(x) \lor talk'(x)]$ [From 2,3 by T12b]

5. every human walks or talks $\Rightarrow$
   $\lambda Q \forall y [human(y) \rightarrow Qe(y)] (\lambda x [walk'(x) \lor talk'(x)]^i)$
   [From 1,4 by T4]

6. $\forall y [human'(y) \rightarrow \lambda x [walk'(x) \lor talk'(x)]^e(j)]$
   [Lambda conversion]

7. $\forall y [human'(y) \rightarrow \lambda x [walk'(x) \lor talk'(x)](y)]$
   [Cancellation Rule]

8. $\forall y [human'(y) \rightarrow [walk'(y) \lor talk'(y)]]$
   [Lambda conversion]
3.3 Anaphoric Pronouns as bound variables:

The rule associated with pronoun is called S14 which is as follow:

S14: If \( \alpha \in PT \), \( \beta \in Pt \), then \( F_{14,n}(\alpha, \beta) \in Pt \)

The translation rule of S14 is:

T14: If \( \alpha \in PT \), \( \beta \in Pt \) and \( \alpha, \beta \) translate into \( \alpha´, \beta´ \) respectively, then \( F_{14,n}(\alpha, \beta) \) translates into \( \alpha´(\lambda x \beta´) \).

Now the sentence “A woman eats and enjoys” by using T14 rule will be as below:

As example pronoun “She” is translated as follow:

\[
\text{She}_n \Rightarrow \lambda P [ P_e(x_n)]
\]

“A woman eats and enjoys.”

1. \( \text{She}_n \Rightarrow \lambda P [ P_e(x_n)] \)
2. \( \text{eat} \Rightarrow \text{eat} \)
3. \( \text{enjoy} \Rightarrow \text{enjoy} \)
4. \( \text{She}_2 \text{ eats} \Rightarrow \lambda P [ P_e(x_2)](\text{eat}´) \) [From 1,2 by T4]
5. \( \text{eat}´(x_2) \) [Lambda conversion]
6. \( \text{enjoy}´(x_2) \) [Cancellation Rule]
7. \( \text{She}_2 \text{ enjoys} \Rightarrow \lambda P [ P_e(x_2)](\text{enjoy}´) \) [From 1,3 by T4]
8. \( \text{a woman} \Rightarrow \lambda P \exists x [ \text{woman}´(x) \land P_e(x)] \)
9. \( \text{a woman eats} \) and \( \text{she enjoys} \Rightarrow \lambda P \exists x [ \text{woman}´(x) \land P_e(x)](\lambda x_2[ \text{eat}´(x_2) \land \text{enjoy}´(x_2)]) \)
10. From 10,11 by T14
11. \( \exists x [ \text{woman}´(x) \land \lambda x_2[ \text{eat}´(x_2) \land \text{enjoy}´(x_2)] \] [Lambda convention]
12. \( \exists x [ \text{woman}´(x) \land \lambda x_2[ \text{enjoy}´(x_2)] ) \) [Cancellation Rule]
13. \( \exists x [ \text{woman}´(x) \land \lambda x_2[ \text{enjoy}´(x_2)] ] \) [Lambda conversion]
3.4 Transitive verbs, Meaning Postulates and Non-Specific Readings:

The rule S5 combine two discussions of transitive verb and term phrase that the result is IV-phrase. The rule S5 is as follow:

**S5**: If α ∈ PTV, β ∈ PT, then F5(α, β) ∈ PIV where F5(α, β) = αβ

The translation of the S5 rule is as follow:

**T5**: If α ∈ PTV, β ∈ PT and α, β translate into α’, β’ respectively, then F5(α, β) translates into α’(β’).

Tom seeks a dog” can be translated as follow by applying T5:

1. seek ⇒ seek´
2. a dog ⇒ λQ∃x[dog´(x) ∧ Q^c(x)] [Previously derived]
3. seek a dog ⇒ seek´(λQ ∃x[dog´(x) ∧ Q^c(x)]^i)
   [From 1, 2 by T5]
4. John seeks a dog ⇒ λP[P^c(t)] (seek´(λQ ∃x[dog´(x) ∧ Q^c(x)]^i))(t)[By T4]
5. seek´(λQ ∃x [dog´(x) ∧ Q^c(x)]^i)(t) [Lambda conversion]
6. seek´(λQ ∃x [dog´(x) ∧ Q^c(x)]^i)(t) [Cancellation Rule]
7. seek´(t, λQ ∃x [dog´(x) ∧ Q^c(x)]^i) [Relational notation]

The other way for translation is:

**“Tom seeks a dog.”**

1. he0 ⇒ λP[P^c(x0)] [Basic expression]
2. seek ⇒ seek´
3. seek him0 ⇒ seek´(λP[P^c(x0)]^i) [By T5]
4. Tom seeks him0 ⇒ λP[P^c(t)](seek´(λP[P^c(x0)]^i)^j)[By T4]
5. seek´(λP[P^c(x0)]^i)^j(t) [Lambda conversion]
6. seek´(t, λP[P^c(x0)]^j)[ Cancellation Rule and Relational Notation]
7. a dog ⇒ λQ ∃x [ dog´(x) ∧ Q^c(x)] [Previously derived]
8. Tom seeks a dog ⇒ λQ ∃x [ dog´(x) ∧ Q^c(x)](λx0 [ seek´(t, λP[P^c(x0)]^i)]^j)[By T14]
9. ∃x [ dog´(x) ∧ λx0 [ seek´(t, λP[P^c(x0)]^i)]^j] (x)] [Lambda conversion]
10. ∃x [ dog´(x) ∧ λx0 [ seek´(t, λP[P^c(x0)]^i)] (x)] [Cancellation rule]
There is another special notation which has been introduced by Montague, the definition is as follow:

\[ \delta = \lambda y \lambda x [ \delta (\lambda P [P_c(y)](x))], \text{where } \delta \in ME_{f(TV)} \]

So based on the new notation, we have:

12. \[ \exists x [\text{dog}(x) \land \text{seek}(\lambda t[\text{P}_c(x)](x))] \]
[Relational notation]

13. \[ \exists x [\text{dog}(x) \land \lambda z [\text{seek}(\lambda P [P_c(x)](z))(t)]] \]
[\lambda -conversion]

14. \[ \exists x [\text{dog}(x) \land \lambda y [\lambda z [\text{seek}(\lambda P [P_c(y)](z))(x)](x)(t)]] \]
[\lambda -conversion]

15. \[ \exists x [\text{dog}(x) \land \text{seek}(x)(t)] \]
[\delta - notation]

16. \[ \exists x [\text{dog}(x) \land \text{seek}(x)(t)] \]
[Relation notation]

The translation for “be” is as follow:

\[ \lambda \Phi \lambda x \Phi (\lambda y [x = y])/e \]

This translation is considered as the most complicated expression assigned as a translation of any English word in Montague semantic based on Dowty et al, 198. In order to describe this notation, the following example shed more light on the definition:
“Maloos is a cat”.

1. a cat \( \Rightarrow \lambda Q \exists x [\text{cat}'(x) \land Q(x)] \) [Previously derived]
2. be a cat \( \Rightarrow \lambda \Phi \lambda x \Phi(\lambda y[x = y])^{\#}(\lambda Q \exists x [\text{cat}(x) \land Q(x)]^{\ lambda}) \) [By T5]
3. \( \lambda \Phi \lambda x \Phi(\lambda y[x = y])^{\#}(\lambda Q \exists x [\text{cat}(z) \land Q(z)]^{\ lambda}) \)
   [Alphabetic variant of 2]
4. \( \lambda x \lambda Q \exists [\text{cat}(z) \land Q(z)]^{\ lambda}(\lambda y[x = y])^{\#} \)
   [Lambda conversion]
5. \( \lambda x \lambda Q \exists [\text{cat}(z) \land Q(z)](\lambda y[x = y])^{\#} \)
   [Cancellation Rule]
6. \( \lambda x \exists [\text{cat}(z) \land Q(z)]^{\ lambda}(\lambda y[x = y])^{\#}(z) \)
   [Lambda conversion]
7. \( \lambda x \exists [\text{cat}(z) \land Q(z)](\lambda y[x = y])^{\#}(z) \)
   [Cancellation Rule]
8. \( \lambda x \exists [\text{cat}(z) \land Q(z)]^{\ lambda}(z) \)
   [Lambda conversion]
9. Maloos is a cat \( \Rightarrow \lambda P[P^e(m)](\lambda x \exists [\text{cat}(z) \land x = z]) \) [By T4]
10. \( \lambda x \exists [\text{cat}'(z) \land x = z](m) \)
    [Lambda conversion]
11. \( \lambda x \exists [\text{human}'(z) \land x = z](m) \)
    [Cancellation Rule]
12. \( \exists [\text{human}'(z) \land m = z] \)
    [Lambda conversion]
13. \( \text{human}'(m) \) [By principle of first-order logic with identity]

### 3.5 Adverbs

Two type of adverbs are categorized in the PTQ grammar that includes sentence adverb like necessarily and verb phrase adverb like slowly. Adverb like necessarily is noted by the symbol □ and the verb phrase adverb is shown as:

\[ \lambda_p \ [ \ p \ ]^{\#} \]

#### Necessarily Mary eats

1. She eats \( \Rightarrow \text{eat}'(x_2) \) [previously derived]
2. necessarily \( \Rightarrow \lambda p [ \ p \ ]^{\#} \) [Basic Expression]
3. necessarily she eats \( \Rightarrow \lambda p [ \ p \ ]^{\#}([\text{eat}'(x_2)]^{\ lambda}) \) [By T2]
4. [\text{eat}'(x_2)] [Lambda conversion and Cancellation rule]
5. necessarily Mary eats \( \Rightarrow \lambda P[P^e(m)](\lambda x_2 [\text{eat}'(x_2)]^{\ lambda}) \) [By T14]
6. \( \lambda x_2 [\text{eat}'(x_2)](m) \) [Lambda conversion and Cancellation Rule]
7. [\text{eat}'(m)] [Lambda conversion]

And sentence like Mary eats slowly is translated as:

Slowly’(‘eat’)(m)
3.6 Negation

The rule S17 of PTQ is related to negation which is as follow:

S17: If $\alpha \in P_T$ and $\beta \in P_{IV}$, then $F_{17a}(\alpha, \beta)$, $F_{17b}(\alpha, \beta)$, $F_{17c}(\alpha, \beta)$, $F_{17d}(\alpha, \beta)$, $F_{17e}(\alpha, \beta) \in P_t$, where:

$F_{17a}(\alpha, \beta) = \alpha \beta$ and $\beta$ is the result of replacing the first verb in $\beta$ by its negative third person singular present;

$F_{17b}(\alpha, \beta) = \alpha \beta''$ and $\beta''$ is the result of replacing the first verb in $\beta$ by its third person singular future;

$F_{17c}(\alpha, \beta) = \alpha \beta'''$ and $\beta'''$ is the result of replacing the first verb in $\beta$ by its negative third person singular present perfect; and

$F_{17d}(\alpha, \beta) = \alpha \beta''''$ and $\beta''''$ is the result of replacing the first verb in $\beta$ by its third person singular present perfect;

The translation of S17 is as follows:

T17: If $\alpha \in P_T$ and $\beta \in P_{IV}$ and $\alpha$, $\beta$ translate into $\alpha'$, $\beta'$ respectively, then: $\neg(\beta')$

- $F_{17a}(\alpha, \beta)$ translates into $\neg\alpha'$ (\beta')
- $F_{17b}(\alpha, \beta)$ translates into $F\alpha'$ (\beta')
- $F_{17c}(\alpha, \beta)$ translates into $\neg F\alpha'$ (\beta')
- $F_{17d}(\alpha, \beta)$ translates into $P\alpha'$ (\beta')
- $F_{17e}(\alpha, \beta)$ translates into $\neg P\alpha'$ (\beta')

Where the negation and tense operators (P and F) are given wider scope than the translation of the subject term phrase.

For example the sentence *every animal doesn’t run* can be translated to:

$\neg \forall x[\text{animal'}(x) \rightarrow \text{run'}(x)]$

or $\forall x[\text{animal}(x) \rightarrow \neg \text{run'}(x)]$
4.1. Montague’s approach to quantifiers

As the previous chapter mentioned, Montague attempted to denote some elements of natural language by set of in entities in the “universe of discourse” such as common nouns and transitive verbs. On the other hand, proper nouns can be denoted as functions defined in terms of entity. It means proper nouns denote functions that take set of entities as an argument and return true if that proper noun is member of function otherwise set the result of function false. In the Frost [2007] version of Montague Semantics:

\[
\begin{align*}
\text{Capone setofents} & \Rightarrow (\text{ENT"capone"}) \in \text{setofents} \\
\text{Torrio setofents} & \Rightarrow (\text{ENT"capone"}) \in \text{setofents} \\
\text{fpg setofents} & \Rightarrow (\text{ENT"fpg"}) \in \text{setofents}
\end{align*}
\]

Frost et al.’s [2007] version of Montague Semantics also defines some denotation related to quantifiers:

\[
\begin{align*}
a \ nph \ vph & \Rightarrow \#(nph \cap vph) \sim 0 \\
two \ nph \ vph & \Rightarrow \#(nph \cap vph) = 2 \\
every \ nph \ vph & \Rightarrow nph \subseteq vph \text{ etc.}
\end{align*}
\]
4.2 Improved Montague Semantic by Event approach

A more efficient form of Montague semantics that can be used as a basis for natural-language query processors to relational databases was developed by Frost [2004] that called FLMS. In paper written by Frost et al. [2004], authors introduced sets and relations instead of entities for the denotations of proper nouns and transitive verbs. For example:

\[
\begin{align*}
||\text{thief}|| &= \{\text{capone, torrio, moran }.. \\
||\text{gang}|| &= \{\text{bowery, fpg }.. \\
||\text{smoke}|| &= \{\text{capone, torrio, moran }.. \\
||\text{Capone}|| &= \lambda p \text{ capone } \in p \\
||\text{Moran}|| &= \lambda p \text{ moran } \in p \\
||\text{Torrio}|| &= \lambda p \text{ torrio } \in p \\
||\text{Five Points Gang}|| &= \lambda p \text{ fpg } \in p \\
||\text{every}|| &= \lambda s \lambda t s \rightarrow t \\
||\text{a}|| &= \lambda s \lambda t s \cap t \neq \{ \} \\
||\text{and}|| &= \lambda f \lambda g \lambda s ((f s) \& (g s)) \\
||\text{join}|| &= \lambda q \{ x(x, image_x) \in \text{collect(join_rel)} \\
& & \& q(image_x) \\
\text{join_rel} &= \{(\text{capone, bowery}, (\text{capone, fpg}), \\
& & (\text{torrio, fpg}), \text{etc.}) \}
\end{align*}
\]

Frost, Agboola, Matthews and Donais [2014] believe that using the entity-based approach that converts natural language to SPARQL queries makes obstacle for development of expressive NL query processors. The reason is that the entity-based approach cannot handle complex prepositional phrases and the development of such theories is considerably more complex because of the translation to SPARQL. The fact that “Capone joined Five Points Gang” is presented as follows in the entity-based approach:

\[
(\ldots/\text{Capone}, \ldots/joined/ \ldots/fpg)
\]

Where “…” represents a Uniform Resource Identifier “URI” for name and “…/capone” is a URI for a person. One major problem of this method is that adding specific date to the fact becomes to difficult:

“Capone joined Five Points Gang in 1914”.

\[
(\ldots/\text{Capone}, \ldots/year\_join\_gang, \ldots/1914)
\]

This does not solve the problem because Capone can join several gangs during years. The alternative approach proposed by Frost et al. [2014] uses events instead of entities as subjects in triples. For example:
{(EV 1001, REL "type", TYPE "join_ev"),
(EV 1001, REL "subject", ENT "capone"),
(EV 1001, REL "object", ENT "fpg"))

The following fact “Capone joined fpg in 1914” can be represented by adding below triple in the event-based approach.

(EV 1001, REL "year", ENTNUM 1914)

The event-based approach is a result of two revisions which have been made to renowned and well known formal semantics of English, called Montague semantic. Montague semantic has been modified to create computationally tractable form which is called FLMS, Frost et al. [1989] that can be useful and suitable as a basis for natural language query interfaces to relational databases. FLMS is then modified to a form which is called EV-FLMS, Frost et al. [2013] which can be suitable as basis for querying event-based triplestores. Here is an example event-based triplestore:

```
data =
[(EV 1000, REL "type", TYPE "born_ev"),
(EV 1000, REL "subject", ENT "capone"),
(EV 1000, REL "year", ENTNUM 1899),
(EV 1000, REL "location", ENT "brooklyn"),
(EV 1001, REL "type", TYPE "join_ev"),
(EV 1001, REL "subject", ENT "capone"),
(EV 1001, REL "object", ENT "fpg"),
(EV 1002, REL "type", TYPE "membership"),
(EV 1002, REL "subject", ENT "capone"),
(EV 1002, REL "year", ENTNUM 1908),
(EV 1003, REL "type", TYPE "join_ev"),
(EV 1003, REL "subject", ENT "capone"),
(EV 1003, REL "object", ENT "fpg"),
(EV 1004, REL "type", TYPE "steal_ev"),
(EV 1004, REL "subject", ENT "capone"),
(EV 1004, REL "object", ENT "car_1"),
(EV 1004, REL "year", ENTNUM 1918),
(EV 1004, REL "location", ENT "manhattan"),
(EV 1005, REL "type", TYPE "smoke_ev"),
(EV 1005, REL "subject", ENT "capone"),
(EV 1006, REL "type", TYPE "membership"),
(EV 1006, REL "subject", ENT "car_1"),
(EV 1006, REL "object", ENT "capone"),
(EV 1007, REL "type", TYPE "membership"),
(EV 1007, REL "subject", ENT "fpg"),
(EV 1007, REL "object", ENT "gang"),
(EV 1008, REL "type", TYPE "membership"),
(EV 1008, REL "subject", ENT "brower"),
(EV 1008, REL "object", ENT "gang"),
(EV 1009, REL "type", TYPE "join_ev"),
(EV 1009, REL "subject", ENT "torrio"),
(EV 1009, REL "object", ENT "fpg"),
(EV 1010, REL "type", TYPE "membership"),
(EV 1010, REL "subject", ENT "capone"),
(EV 1010, REL "object", ENT "person"),
(EV 1011, REL "type", TYPE "membership"),
(EV 1011, REL "subject", ENT "torrio"),
(EV 1011, REL "object", ENT "person")]
```

The fact “Capone smokes” which has an intransitive verb is represented by

```
{(EV 1005, REL "type", TYPE "smoke_ev"),
```
Set membership is the result of an action, for example “Capone became a thief”, can be represented by modeling set membership as an event. The modeling is brought below:

{(EV 1002, REL "type", TYPE "membership"),
 (EV 1002, REL "subject", ENT "capone"),
 (EV 1002, REL "object", ENT "thief")}

The fact that “he became a thief in 1908” can be represented just by adding the following triple:

(EV 1002, REL "year", ENTNUM 1908)

If set membership is considered a consequence of an intrinsic property of an entity, e.g. the triples representing following fact “Capone stole a car in 1918 in Manhattan” are:

{(EV 1004, REL "type", TYPE "steal_ev"),
 (EV 1004, REL "subject", ENT "capone"),
 (EV 1004, REL "object", ENT "car1"),
 (EV 1004, REL "year", ENTNUM 1908),
 (EV 1004, REL "location", ENT "Manhattan")}

The providing explanation is derived from Frost et al. [2014]. It is required to notify that “car1” is member of “car” set. Thus, the following triples should exist in the triplestore:

{(EV 1006, REL "type", TYPE "membership"),
 (EV 1006, REL "subject", ENT "car1"),
 (EV 1006, REL "object", ENT "car")}

The basic retrieval functions that have been defined in event-based approach are provided.

The “gets” function returns triples from data that match to taken field value(s):

gets (a,ANY,ANY) = [(x,y,z) | (x,y,z) <- data; x = a]
gets (ANY,ANY,c) = [(x,y,z) | (x,y,z) <- data; z = c]
etc.

Example:

gets (ANY, "subject", "torrio") => [(1009, "subject", "torrio"), (1011, "subject", "torrio"), etc.]
gets (1009, "type", ANY) => [(1009, "type", join_ev)]
Operators to extract one or more fields from a triple include:

\[
\begin{align*}
\text{first} & (a,b,c) = a \\
\text{second} & (a,b,c) = b \\
\text{third} & (a,b,c) = c \\
\text{thirdwithfirst} & (a,b,c) = (c, a) \text{ etc.}
\end{align*}
\]

There are some operators that can return sets of fields from sets of triples by using the functions:

\[
\begin{align*}
\text{firsts trips} &= \text{map} \ \text{first} \ \text{trips} \\
\text{thirds trips} &= \text{map} \ \text{third} \ \text{trips} \\
\text{thirdswithfirsts trips} &= \text{map} \ \text{thirdwithfirst} \ \text{trips} \ \text{etc.}
\end{align*}
\]

More difficult operators can be defined such as:

\[
\begin{align*}
\text{get_subj_for_event} \ ev &= \text{thirds} \ (\text{getts} \ (ev, \ \text{REL} \ "subject", \ \text{ANY})) \\
\text{get_subjs_for_events} \ evs &= \text{concat} \ (\text{map} \ \text{get_subj_for_event} \ evs)
\end{align*}
\]

Example:

\[
\text{get_subjs_for_events} \ [\text{EV 1000}, \ \text{EV 1009}] \Rightarrow \{\text{ENT "capone"}, \ \text{ENT "torrio"}\}
\]

The function “get_members” returns all entities that are members of a taken set argument:

\[
\begin{align*}
\text{get_members} \ set &= \text{get_subjs_for_events} \ evs \ \text{where} \ evs = \text{firsts} \ (\text{getts} \ (\text{ANY}, \ \text{REL} \ "type", \ \text{TYPE "membership"})) \\
&= \text{firsts} \ (\text{getts} \ (\text{ANY}, \ \text{REL} \ "object", \ \text{ENT set})) \\
&= \text{intersect} \ evs \\
\text{events} &= \text{intersect} \ \text{events_for_type_membership} \ \text{events_for_set_as_object}
\end{align*}
\]

An example of operator usage is:

\[
\text{get_members} \ "person" \Rightarrow \{\text{ENT "capone"}, \ \text{ENT "torrio"}\}
\]

Another useful operator is one which returns all of the subjects of an event of a given type:

\[
\begin{align*}
\text{get_subjs_of_event_type} \ event_type &= \text{get_subjs_for_events} \ evs \\
&= \text{firsts} \ (\text{getts} \ (\text{ANY}, \ \text{REL} \ "type", \ \text{TYPE event_type}))
\end{align*}
\]
As instance we have

\[
\text{get_subjs_of_event_type "smoke" => [ENT "capone"]}
\]

There is denotation of noun in the event based approach that is the set of entities. The entities are members of the set associated with that noun. The “get_members” function returns a set as a list. These denotations can then be used by other functions in the program.

\[
\begin{align*}
\text{person} &= \text{get_members "person" } \\
\text{gang} &= \text{get_members "gang" } \\
\text{car} &= \text{get_members "car" } \\
\text{thief} &= \text{get_members "thief" } \\
\text{e.g. } \text{gang} &= \text{[ENT "fpg", ENT "bowery"]}
\end{align*}
\]

The denotation of an intransitive verbs can be defined as the set of entities that are subjects of an event associated with verb:

\[
\begin{align*}
\text{smoke} &= \text{get_subjs_of_event_type "smoke\_ev" } \\
\text{e.g. } \text{smoke} &= \text{[ENT "capone"]}
\end{align*}
\]

Intransitive use of transitive verbs are similar:

\[
\begin{align*}
\text{steal\_intrans} &= \text{get_subjs_of_event_type "steal\_ev" } \\
\text{steal\_intrans} &= \text{[ENT"capone"]}
\end{align*}
\]

In event-based approach proper nouns denote functions that take a set of entities as argument and return “True” if a particular entity is a member of set otherwise result is “False”:

\[
\begin{align*}
\text{capone setofents} &= \text{member setofents (ENT "capone") } \\
\text{torrio setofents} &= \text{member setofents (ENT "torrio") } \\
\text{car\_1 setofents} &= \text{member setofents (ENT "car\_1") } \\
\text{fpg setofents} &= \text{member setofents (ENT "fpg") } \\
\text{year\_1908 setofents} &= \text{member setofents (ENTNUM 1908)} \\
\text{etc.}
\end{align*}
\]

Example:

\[
\text{capone smoke} \Rightarrow \text{True}
\]
The quantifiers such as “a”, “one”, “two”, “every”, etc. and conjoiners are defined in the same way described in FLMS:

- \( a \) \( nph \) \( vbph \) = \#(intersect \( nph \) \( vbph \)) \( \approx \) 0
- \( one \) \( nph \) \( vbph \) = \#(intersect \( nph \) \( vbph \)) = 1
- \( two \) \( nph \) \( vbph \) = \#(intersect \( nph \) \( vbph \)) = 2
- \( every \) \( nph \) \( vbph \) = subset \( nph \) \( vbph \)
- \( nounand \) \( s \) \( t \) = intersect \( s \) \( t \)
- \( nounor \) \( s \) \( t \) = mkset (\( s ++ t \))
- \( that \) = \( nounor \)

\[
\text{termand } tmph1 tmph2 = f \text{ where } f \text{ setofevs} = (tmph1 \text{ setofevs}) \& (tmph2 \text{ setofevs})
\]

\[
\text{termor } tmph1 tmph2 = f \text{ where } f \text{ setofevs} = (tmph1 \text{ setofevs})\text{/}(tmph2 \text{ setofevs})
\]

In order to depict the transitive verb denotation, the “image” has been defined:

\[
\text{make} \text{ image } et
\]
\[
= \text{collect} (\text{concat})\left((\text{thirds} \text{with firsts} \text{ . } \text{getts})\right)
\]
\[
\left((ev, \text{REL } "subject", \text{ANY})| ev \leftarrow \text{ events}\right))
\]

where
\[
\text{events} = (\text{firsts} \text{ . } \text{getts})\left(\text{ANY, REL } "type", \text{TYPE } et\right)
\]

Example:

\[
\text{make} \text{ image } "join\_ev"
\]
\[
\Rightarrow [\text{(ENT } "capone", [\text{EV 1001, EV 1003}]\text{)},
\text{(ENT } "torrio", [\text{EV 1009}]\text{)}]
\]

The transitive verb denotation is as follow:

\[
\text{Join} = f \text{ where } f \text{ tmp}
\]
\[
= [\text{subj}](\text{subj.evs})\leftarrow \text{ make} \text{ image } "join\_ev"\text{;}
\text{tmp}(\text{concat}((\text{thirds.getts})
\text{ (ev, REL } "object", \text{ANY})| ev \leftarrow \text{ evs}))]
\]

Example:

\[
\text{join (a gang)} \Rightarrow [\text{ENT } "capone", \text{ENT } "torrio"]
\]
If user asks question such as:

“Did Capone steal a car?” make_image output will be as following:

\[
\text{make_image} \text{ “steal}_\text{ev}} \Rightarrow \{(\text{ENT "capone”, [EV 1004, EV 1012]}, \\
(\text{ENT "torrio”, [EV 1013]})\}
\]

Afterward “a car” is applied to the output set and it returns whole subjects of the triple that are associated with “a car”.

\{“capone”, “torrio”\}

In the next step, subject “Capone” will apply on the result set and then returns “True” if the “Capone” is member of the result set, otherwise the returning value is “False”.

\text{Capone (steal (a car)) ⇒ True}

The problem of the accommodating prepositional phrases is being highlighted and remarkable in this step because the return value is Boolean type that it is not possible to ask questions like “when…?” or “where….?”. In order to query “when” or “where” type of questions that search for preposition phrases, the expecting result value should be event type, not Boolean. Usage of event type is required to figure out what time or which place is associated with the returned value. With existing event based approach, it is possible to answer the “who ….?” or “what…?” type of questions but not “when” and “where” questions type.

Frost, Agboola, Matthews, & Donais [2014] pointed that their proposed approach is able to locate multiple prepositional phrase easily by parsers that convert the list of prepositional phrases to list of prepositional phrase list that each pair includes REL value and termphrase.

Two following phrase that contains prepositional phrase are converted as:

“in 1908 or 1918, in Manhattan”

\[(\text{REL "year", year}_\text{1908 $\&termor year}_\text{1918}), \\
(\text{REL "location", "manhattan")}]\]
4.3 Other researcher’s approaches to accommodate the prepositional phrase

In order to attain the formal knowledge in ontologies which is used to represent the taxonomy of domain and plays one of the most important role in semantic web, users are required to be familiar with some formats like the ontology syntax such as RDF and OWL, query languages such as RDQL2 and SPARQL3, the structure of the target ontology. In paper referred by Wang, Xiong, Zhou & Yu [2007] state the previous research which has been done by Bernstein, Kaufmann, Gohring & Kiefer [2007] expressed there is undeniable gap between the logic of the semantic web and real-world users. That would be too difficult for users to remember the mentioned format. On the other hand, semantic web requires ontology syntax to have expected efficiency. In paper referred by Wang, Xiong, Zhou & Yu [2007] the authors considered related paper referred by Kaufmann, Bernstein, Zumstein [2006] which they have following shortcoming and defects, the authors stated it is too tough for machines to understand the ordinary natural language because of its ambiguity and complexity. Although NLP community have been attempting to solve and they reach up to 90% in precision and recall, it has so far to recognize words and resolve it.

On the other hand, even parsing natural languages would have been done successfully, there should be some obstacles to translate them to appropriate formal queries. For example the stored vocabularies in knowledge base is totally different with user’s entered words so one of the challenges is to map the user’s words to existing vocabulary in the database. The authors mention that the other challenges in this area is how to utilize semantic information in knowledge base which includes lexical and syntactic information of parsed queries to acquire the formal query. One more thing is that various representations need different methods to interpret the user queries. Although SPARQL is introduced as standard query language for semantic web community, novel approaches are attempting to translate natural language to them.

The new idea which Wang et al. [2007] propose includes following major steps: the nominal phrases are extracted from parse tree and they will be shown in intermediate representation which is called QueryTriples and then they are mapped to OntoTriples which are depicted with entities in ontology by PANTO and then OntoTriples will be translated to SPARQL by targets and modifiers extracted from parse tree. The authors state
that they investigate some problems with translating natural language queries to SPARQL queries and some complicated and advanced features of the semantic like negation, comparative and superlative modification are considered in their new idea.

The authors implemented PANTO as stand-alone web application, it uses StanfordParse7 which produces one parse tree for each input. Test data which has been used in this experiment are based on Mooney which is being used to evaluate the natural language interface. Geography data in United States, restaurant information and job announcement are three domains in this dataset. Original Prolog database is translated into OWL as ontologies in current experiment.

The goal of the experiments is to quantitatively assess the performance and effectiveness of our approach in the current implementation.

The authors state to assess the correct rate that how many of the translated queries correctly represent the semantics of the original natural language queries,

The metrics precision and recall are being used to compare the output with the manually generated SPARQL queries. For each domain, precision means the percentage of correctly translated queries in the queries that PANTO produced an output; recall refers to the percentage of queries that PANTO produced an output in the total query set. Since the queries are prepared for evaluating natural language interface to database, some features in the queries are not supported by the current version of SPARQL.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Geography</th>
<th>Restaurant</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Mooney Queries#</td>
<td>880</td>
<td>250</td>
<td>641</td>
</tr>
<tr>
<td>Selected Testing Queries#</td>
<td>877</td>
<td>238</td>
<td>517</td>
</tr>
<tr>
<td>Precision</td>
<td>88.05%</td>
<td>90.87%</td>
<td>86.12%</td>
</tr>
<tr>
<td>Recall</td>
<td>85.86%</td>
<td>96.64%</td>
<td>89.17%</td>
</tr>
</tbody>
</table>

Table2. Wang, Xiong, Zhou & Yu [2007]

The authors state that the total processing time of query processing was less than one second (0.878) and running time is based on the scale of the ontology and also complexity of the query which is the length and number of the clauses. The authors claim that precision PANTO provides acceptable result for recall and precision in compared with Querix which achieved a precision 86.08 and recall 87.11. The authors experimented on the Mooney data
and they claim that queries show that all the correctly parsed natural language queries can be correctly translated into QueryTriples and then be mapped to OntoTriples. They claim that their experiments on three different ontologies have shown that the PANTO approach produces promising results also their novel approach helped bridge the gap between the logic-based semantic web and real-world users.

Also the authors claim PANTO can cover the whole query scope which are supported by AquaLog since Aqualog can parse queries and PANTO can generate them too.
The event based approach that described in Chapter 4 is a modified version of Montague Semantics that can be useful as basis for natural language interface to relational databases. The improved version of Montague semantics that is described in previous chapter is called EV-FLMS proposed by Frost et al. [2013] that uses events to represent facts. The event-based approach can handle cover and provide answers for questions such as “Who…?” “What…?” but it has some shortcoming that is main motivation of doing current thesis. Take the following fact as example “Capone stole a car in 1918”. The approach was suggested by Frost et al. [1989] can answer queries like “Did Capone steal a car in 1918”. Here is part of data in database related to above fact:

```python
data = {(EV 1004, REL "type", TYPE "steal_ev"),
        (EV 1004, REL "subject", ENT "capone"),
        (EV 1004, REL "object", ENT "car_1"),
        (EV 1004, REL "year", ENTNUM 1918),
        (EV 1006, REL "type", TYPE "membership"),
        (EV 1006, REL "subject", ENT "car_1"),
        (EV 1006, REL "object", ENT "car")}
```
In order to answer “Did Capone steal a car?”, it requires to apply functions like “make_image eve” that has been defined by Frost et al. [2013]. This function take a event name that is “steal_eve” in this example. The operation of the function is as follow:

\[
\text{make_image} \text{ "steal\_eve"} \Rightarrow \\
\{(\text{ENT "capone"}, [EV 1004, EV 1012]), (\text{ENT "torrio"}, [EV 1013])\}
\]

Afterward the defined function called “steal_eve tmph” function that takes “tmph” as argument and its output is set of entities that are subjects of steal event type and their object are tmph in triplestore. The denotation for transitive verb like steal is as follow:

\[
\text{steal\_ev} \text{ tmph} = [\text{subj} | (\text{subj, evs}) \leftarrow \text{make\_image} \text{ "steal\_ev"}; \text{tmph(\text{concat}[(\text{thirds, gets}) (\text{ev, REL "object"}, \text{ANY}) | \text{ev} \leftarrow \text{evs})])}
\]

So we have:

\[
\text{Steal(car) \Rightarrow [\text{ENT "capone"}, \text{ENT "torrio"} ]}
\]

The result is a set containing subject of steal event that the object is a “car”. To provide answer to query “Did Capone steal a car?”, it needs to apply the function of “Capone” to result of “steal\_ev tmph” function. The “Capone” function takes a set of entities like the set result of “steal\_ev tmph” and proveds “True” if “Capone” is member of the set otherwise “False”.

\[
\text{Capone (steal (a car)) \Rightarrow Capone ([\text{ENT "capone"}, \text{ENT "torrio"}]) \Rightarrow True}
\]

Based on Boolean result, the “Did Capone steal a car?” type of question can be answered. In the above example the output is true, so the answer for this query is “Yes”.

According paper referred by Frost et al. [2013], event-based semantics can accommodate queries such as:

“Who stole a car in Manhattan in 1918 or 1920?”
But not

“When did Capone steal a car in Manhattan?”

This thesis modifies the event-based semantics to accommodate queries of the form:

“When did…?”
“Where did …?”
“With what did..?”

The approach proposed by Frost et al. [2013] cannot answer the “when did..?,” “where did..?, and”, “with what did..?” type of questions because the output after applying functions like “Capone” is Boolean type of data. Thus, if “When” function takes Boolean type of data as an argument, it cannot provide the answer. It needs to take a set of steal events that “Capone” is subject and “car” is object of that. Then, it returns years of the events.

Because of the mentioned shortcoming, paper referred by Frost et al. [2013] cannot provide answer for the “when did..?”. In the next chapter, the new idea that improves event-based approach and also coverage of queries can be answered is provided, questions that looks for prepositional phrases in the fact.
6.1. Summary of the new idea

One of shortcomings of event-based approach which has been introduced by Frost et al. [2013] and is inspired by Montague Semantic, is related to how to accommodate the prepositional phrase and answer the queries that concerns about time and place of facts. The detail of shortcomings locating of prepositional phrases in EV-FLMS (Frost et al. [2013]) has been provided in previous chapter. The problem of prepositional phrase accommodation in event-based approach is related to functions definition. The defined functions in paper referred by Frost et al. [2013] should be modified in way that return events rather than entities. Some defined functions in event-based approach such as “Capone” do not return event type of data that is main problem of prepositional phrase locating. The main intention of this thesis is to modify the functions definition and the other notations in way to return events instead of entities or Boolean. The event type of data can be used for further processing and be helpful for prepositional phrase accommodation to answer “when did..,” “where did…and,” ”with what did…” type of questions. On the other hand, quantifiers such as “a”, “two” and “every” definitions needs to be modified in way
to returns set that contains pair of subject and event, in the current version of the EV-FLMS approach the definition for the quantifier “a” is as:

\[ a \text{nph vbph} = \#(\text{intersect nph vbph}) \sim= 0 \]

That notation is based on intersection of nph and vbph that return value should not be zero. In the new definition of “a” quantifier that is event-based should return pair of subjects and events that can be used in next steps. It has been attempted to alter the all defined notations in way to reach the events easily and can be covered whole type of questions.

\[ \text{new}_a \text{nph vph} = [(\text{subj2, evs2}) | (\text{subj1, evs1}) \leftarrow \text{nph}; \text{subj2, evs2} \leftarrow \text{vph}; \text{subj1} = \text{subj2}] \]

**6.2. Statement and Contribution**

It is possible to construct natural language query interfaces to triplestores that can accommodate arbitrarily-nested complex chained prepositional queries such as:

- “When did Capone steal a car in Manhattan using a crowbar…?”
- “Where did Capone steal a car in 1918 or 1920…?”
- “With what did Capone steal two cars and a truck that was owned by a gangster who joined every gang in Brooklyn that was joined by Torrio?”

The notation that have been added to EV-FLMS approach, proposed by Frost et al. [2013] are as follow that are related to recognizing the time of fact or stored data as event.

\[ \text{year}_\text{of ev} = \text{thirds} (\text{getts (ev, REL "year", ANY)}) \]

The above function takes an event and returns the year associated with the year. In order to get set of years, it requires to denote another function that collects all related years. The definition of this function is as follow:

\[ \text{years}_\text{of evs} = \text{concat (map year}_\text{of evs)} \]

Almost all defined notations needs to be modified to recognize and locate the prepositional phrase. For more clarification, all the modified functions and notations start with “new_”.
For example, here is defined notation for the “Capone” function that returns pairs of events and subjects instead of Boolean value:

\[
\text{new\_capone\ ev\_image} = [(\text{subj}, \text{evs}) \mid (\text{subj}, \text{evs}) \leftarrow \text{ev\_image}; \text{subj} = \text{ENT "capone"}]
\]

Moreover, the notation for the image function of “steal” can be explained as below, the image function attempts to collect pair of (subj, obj, [eve]) that events have to be “steal” and subjects and objects can be anything.

\[
\text{new\_image\_steal} = \text{collect} [(\text{subj}, (\text{obj}, [\text{ev}])) \mid \text{ev} \leftarrow \text{events}; \text{subj} \leftarrow \text{thirds (getts (ev,REL "subject", ANY))}; \text{obj} \leftarrow \text{thirds (getts (ev, REL "object", ANY))}]
\]

where \text{events} = \text{firsts (getts (ANY, REL "type", TYPE "steal\_ev")).}

The other function that has to be changed to provide answer for “When did Capone steal a car?” is “steal tmph” that gets the tmph, in this case tmph is “a car”, and returns a set of pairs which each of them contains subject and all the events that are type of “steal” and their object is taken tmph input.

\[
\text{new\_steal\ tmph} = [(\text{subj}, \text{concat (map snd (tmph objs\_evs))}) \mid (\text{subj}, \text{objs\_evs}) \leftarrow \text{new\_image\_steal}; \text{tmph objs\_evs} \leftarrow [\text{[]}]}
\]

In addition to above modification, notation for “car” and “a” also are required to be changed as follow:

\[
\text{new\_get\_members\ set} = [(\text{get\_subj\_for\_event ev})!0, \text{[ev]}) \mid \text{ev} \leftarrow \text{events}]
\]

where

\[
\text{events\_for\_type\_membership} = \text{firsts (getts (ANY, REL "type", TYPE "membership"))}
\]
events_for_set_as_object
= firsts (getts (ANY, REL "object", set))
events = intersect events_for_type_membership
events_for_set_as_object
new_car = new_get_members (ENT "car")
new_a nph vph
= [(subj2, evs2) | (subj1, evs1) < nph;
     (subj2, evs2) < vph;
     subj1 = subj2]

The result for the “new_steal(new_a new_car)” is:
[(ENT "capone",[EV 1004,EV 1012]),(ENT "torrio",[EV 1014])]

The above set contains pairs of subjects with events that type of each event is “steal” and object of them is “a car”. The result of applying new definition of “Capone” function to result of above set is as follow:
new_capone(new_steal(new_a new_car)) = [(ENT "capone",[EV 1004,EV 1012])]

In order to obtain the time attribute of fact or event, definition of other function is necessary, which is called “when”. The “when” function takes the image of events as input and returns set of years correspondent to taken input events.
when ev_image
= [(subj, years_of evs) | (subj, evs) < ev_image; years_of evs~=[[]]

If “when” function applies to result of “new_capone(new_steal(new_a new_car))”, the result is as following:
[(ENT "capone",[ENTNUM 1908,ENTNUM 1928])]

The year 1908 is related to “EV 1004” and year 1908 is related to “EV 1012".
Chapter 7: Implementation of the new idea

The implementation of new idea is provided in this chapter. The prototype of the Semantics have been implemented by Miranda programming language. However, other programming languages that can support high level functions can implement these semantics such as Lisp, Haskell, Python and Scheme. The data which has been used for testing is:

data = [(EV 1000, REL "type", TYPE "born_ev"),
        (EV 1000, REL "subject", ENT "capone"),
        (EV 1000, REL "year", ENTNUM 1899),
        (EV 1000, REL "location", ENT "brooklyn"),
        (EV 1001, REL "type", TYPE "join_ev"),
        (EV 1001, REL "subject", ENT "capone"),
        (EV 1001, REL "object", ENT "fpg"),
        (EV 1002, REL "type", TYPE "membership"),
        (EV 1002, REL "subject", ENT "capone"),
        (EV 1002, REL "object", ENT "thief"),
        (EV 1002, REL "year", ENTNUM 1908 ),
        (EV 1003, REL "type", TYPE "join_ev"),]
(EV 1013, REL "object", ENT "car"),
(EV 1014, REL "type", TYPE "steal_ev"),
(EV 1014, REL "subject", ENT "torrio"),
(EV 1014, REL "object", ENT "car_3"),
(EV 1015, REL "type", TYPE "membership"),
(EV 1015, REL "subject", ENT "car_3"),
(EV 1015, REL "object", ENT "car")

Data comprises set of triples which these triples shows part of fact. Thus, each triple of fact relates to the other triple by the unique identification of the fact or EV that is “EV number”.

Take “EV 1012” as example:

(EV 1012, REL "type", TYPE "steal_ev"),
(EV 1012, REL "subject", ENT "capone"),
(EV 1012, REL "object", ENT "car_2"),
(EV 1012, REL "year", ENTNUM 1928),
(EV 1012, REL "location", ENT "brooklyn")

The above triples with identification “EV 1012” denotes “Capone steal car_2 in 1928 at Brooklyn”. The function “get_members” is defined by EV-FLMS but I have modified as follow:

new_get_members set
= \{([\{(get_subj_for_event ev)!0, [ev]\} | ev <- events] where
  events_for_type_membership
  = firsts (getts (ANY, REL "type", TYPE "membership"))
  events_for_set_as_object
  = firsts (getts (ANY, REL "object", set))
  events = intersect events_for_type_membership
             events_for_set_as_object

This function is being used for recognizing the appropriate names like “person”, “gang”, etc... The modified notation of the appropriate name is:

new_gang = new_get_members (ENT "gang")
new_person = new_get_members (ENT "person")
new_car = new_get_members (ENT "car")
new_thief = new_get_members (ENT "thief")
new_gangster = new_get_members (ENT "gangster")

As example:
new_gang => [(ENT "fpg",[EV 1007]),(ENT "bowery",[EV 1008])]
new_person => [(ENT "capone",[EV 1010]),(ENT "torrio",[EV 1011])]
new_car => [(ENT "car_1",[EV 1006]),(ENT "car_2",[EV 1013]),(ENT "car_3", [EV 1015])]
new_thief => [(ENT "capone",[EV 1002])]

The next modification is regarding intransitive verbs such as “smoke”. The result of intransitive verb is set containing pair of subjects and events that are associated with intransitive verb. The modified notation of the “smoke” is as below:

new_smoke = make_intrans (TYPE "smoke_ev")
make_intrans et = collect bin_rel
where
bin_rel = [(thirds (getts(ev, REL "subject", ANY))!0,ev)
           | ev <- events]
           events = firsts (getts (ANY, REL "type", et))

The example for intransitive verb is as follow:
new_smoke => [(ENT "capone",[ENTNUM 1908,ENTNUM 1908])]
pair of subjects and events if subject is “Capone”. To shed more light on it, take following as example:

\[
\text{new\_capone ev\_image} \\
= [(\text{subj, evs}) \\
\quad | (\text{subj, evs}) \leftarrow \text{ev\_image}; \text{subj = ENT } \text{"capone"}]
\]

The verb “steal” is being considered as example, because “steal” is transitive verb so it needs object that “a car” is used for this example.

“Capone steals a car”

\[
\text{new\_capone (new\_steal (new\_a new\_car))} \Rightarrow [(\text{ENT } \text{"capone"}, [\text{EV 1004, EV 1012}])]
\]

The “new\_steal” function takes the output of “new\_image\_steal” and filter the set based on taken argument. In the other word, “new\_steal” function provides set of subjects and events from “new\_image\_steal” if the argument is “obj” in “new\_image\_steal” return value. Afterward, “new\_capon” applies on result of “new\_steal” and take pair provided on subject is “Capone”.

Here is definition for other proper noun:

\[
\text{new\_torrio ev\_image} \\
= [(\text{subj, evs}) \\
\quad | (\text{subj, evs}) \leftarrow \text{ev\_image}; \text{subj = ENT } \text{"torrio"}]
\]

“Torrio steals a car”

\[
\text{new\_torrio (new\_steal (new\_a new\_car))} \Rightarrow [(\text{ENT } \text{"torrio"}, [\text{EV 1014}])]
\]

“Torrio steals every car”

\[
\text{new\_torrio (new\_steal (new\_every new\_car))} \Rightarrow []
\]

Since it has been pointed to role of quantifiers in previous examples, the improved notations of them are provided here that is event-based version, quantifier notations in FLMS has been provided in chapter 4.

Here is definition for “a”, “new\_a” function takes two argument nph (noun phrase) and vph (verb phrase) and returns (subj, evs) if the subjects of noun phrase and verb phrase is equal.
new_a nph vph
= [(subj2, evs2) | (subj1, evs1) <- nph;
    (subj2, evs2) <- vph;
    subj1 = subj2]

The event-based notation for “every” is as follow. This function takes noun phrase and verb phrase as arguments, subjects of noun phrase should be subset of verb phrase subject set. The result is set containing pairs of noun phrase subjects with events of verb phrase when their subject is equal, otherwise the return set is null.

new_every nph vph
= result, if subset (map fst nph) (map fst vph)
  = [], otherwise
  where result = [(subj2, evs2) | (subj1, evs1) <- nph;
                  (subj2, evs2) <- vph;
                  subj1 = subj2]

The notation of conjoiners have been modified as follow:

new_termand tmph1 tmph2
= f  where
  f  ev_image
  = res1 ++ res2, if (res1 ~=[[]]) & (res2 ~=[[]])
  = [], otherwise
  where
    res1 = tmph1 ev_image
    res2 = tmph2 ev_image

new_termor tmph1 tmph2
= f  where
  f  ev_image = tmph1 ev_image ++ tmph2 ev_image
One of main complicated part of semantics can be transitive verbs. In order to modify the transitive verbs the “make_image” function which is used in EV-FLMS approach also requires to be changed. The modified version of “image” function for “steal” verb is in this way:

\[
\text{new_image}\_\text{steal} = \text{collect}
\]

\[
[(\text{subj}, (\text{obj}, [\text{ev}])) | \text{ev} \leftarrow \text{events};
\text{subj} \leftarrow \text{thirds (getts (ev,REL "subject", ANY))};
\text{obj} \leftarrow \text{thirds (getts (ev, REL "object", ANY))}]
\]

where

\[
\text{events}
\]

\[
= \text{firsts (getts (ANY, REL "type", TYPE "steal\_ev"))}
\]

The “image\_steal” function in above definition should collect objects and events associated with “steal” verb based on subjects of considered events.

\[
\text{new_image}\_\text{steal} = \text{collect}
\]

\[
[(\text{subj}, (\text{obj}, [\text{ev}])) | \text{ev} \leftarrow \text{events};
\text{subj} \leftarrow \text{thirds (getts (ev,REL "subject", ANY))};
\text{obj} \leftarrow \text{thirds (getts (ev, REL "object", ANY))}]
\]

In the above example that considers image of steal, the result are set of subjects that steals anything (object) with correspondent event type. According provided data, “Capone” steals “car\_1” and “car\_2” in the event of [EV 1004] and [EV 1012] respectively. However, entity “Torrio” steal “car\_3” in event of [EV 1014].

The other transitive verb that has been used as experiment is “join” that requires “image” functions. The modified notation for “image\_join” is as follow:

\[
\text{new_image}\_\text{join} = \text{collect}
\]

\[
[(\text{subj}, (\text{obj}, [\text{ev}])) | \text{ev} \leftarrow \text{events};
\text{subj} \leftarrow \text{thirds (getts (ev,REL "subject", ANY))};
\text{obj} \leftarrow \text{thirds (getts (ev, REL "object", ANY))}]
\]

where

\[
\text{events}
\]
The performance and definition of “image_join” is same as “image_steal”. The only difference is that this function considers events that are “join” type. We can generalize the image function in way that accepts the verb or “eve” and searches data for that specific transitive verb so it does not required to denote a function for image each of transitive verb.

\[
\text{new_image_join} \rightarrow [(\text{ENT "capone"}, [(\text{ENT "fpg"}, [\text{EV 1001}]), (\text{ENT "bowery"}, [\text{EV 1003}])), (\text{ENT "torrio"}, [(\text{ENT "fpg"}, [\text{EV 1009}])])]
\]

Based on above example, Capone joins fpg [EV 1001], Capone joins bowery [EV1003] and Torrio joins fpg [EV1009].

The notation with usage of image function for transitive verb is as below:

\[
\text{new_steal tmph} = \[(\text{subj}, \text{concat(map snd (tmph objs_evs)))} | \text{(subj, objs_evs)} \leftarrow \text{new_image_steal; tmph objs_evs} \neq \[] \]
\]

In the above function, “new_steal” takes term phrase as input since “steal” is transitive verb and needs object and then check whether taken term phrase as same as object which has been extracted in “image_steal” function or not. If the term phrase and object of “image_steal” are same, associated subject and events are considered as output and then the result will be collected based on subjects.

For example:

\[
\text{new_steal(new_a new_car)} \rightarrow [(\text{ENT "capone"}, [\text{EV 1004, EV 1012}]), (\text{ENT "torrio"}, [\text{EV 1014}])])
\]

\[
\text{new_steal(new_every new_car)} = [\]
\]

The result for “new_steal(new_every new_car)” is null set expectedly since there is no any subjects that steal all the cars defined in “new_car"function.
The definition for “new_join tmph” is as follow which is same as “new_steal tmph” but looking for join verb in type of events.

\[
\text{new}_\text{join tmph} = ((\text{subj, concat(map snd (tmph objs_evs)))) | \\
\text{(subj, objs_evs) } \text{<- new_image_join; } \\
\text{tmph objs_evs } \sim = \text{[]} )
\]

Example for “new join tmph” is in this way:

\[
\text{new}_\text{join(new_a new_gang)} \Rightarrow [(\text{ENT "capone",[EV 1001,EV 1003]}),(\text{ENT "torrio", [EV 1009]})]
\]

The above example relates Capone joins two gangs [EV1001, EV 1003] and torrio joins just one gang based on [EV 1009].

Example:

\[
\text{new}_\text{join(new_every new_gang)} \Rightarrow [(\text{ENT "capone",[EV 1001,EV 1003]})]
\]

The above function notes that Capone joins all the gangs defined in data, fpg and bowery, that details can be found in [EV 1001] and [EV 1003].

Based on base and simple functions which have described above, more complex queries can be answered. In the following complex queries and accurate result have been provided:

1. “A person steals a car”. This query can be answered with set pairs includes subjects that are member of person set and events that are type of steal and have “car” as object which is member of “new_car” set.

\[
\text{new_a new_person (new_steal (new_a new_car)) } \Rightarrow [(\text{ENT "capone",[EV 1004, EV 1012]}),(\text{ENT "torrio",[EV 1014]})]
\]

2. “Every person joins a gang”. The result of this query should be all the member set of person who joins at least one of the defined gangs in data.
new_every new_person (new_join (new_a new_gang)) => [(ENT "capone", [EV 1001, EV 1003]), (ENT "torrio", [EV 1009])]

3. “When did Torrio steal a car?”. This queries asks for prepositional phrase so the answer of it should be set of years that “Torrio stole a car” which is empty set.
new_when (new_torrio (new_steal (new_a new_car))) => []

4. “When did Torrio join a gang?”. This query returns empty set since there is no data which shows Torrio joined a gang in specific time because of that “new_when” function returns null.
The “born” verb which is intransitive verb also added that the notation is as follow:
new_image_born
= collect
    [(subj, [ev]) | ev <- events;
     subj <- thirds (getts (ev, REL "subject", ANY))]

where events
    = firsts (getts (ANY, REL "type", TYPE "born_ev"))
new_born = make_intrans (TYPE "born_ev")

Based on above definition the answer for queries like “when was Capone born?” is in this way:
new_when (new_capone (new_born)) => [(ENT "capone", [ENTNUM 1899])]
8.1 Termination

According to paper referred by Bruynooghe, et al [2007], the main goal of termination analysis is that program terminates for specified and infinite input. The proof of termination is based on size of functions that map program states to well-founded domain elements. The proof of termination is really essential especially when the program goes through loops. Termination is guaranteed that the program can be finished if the size of input is bounded. The program executes in finite numbers because size number of input decrease in each computation under condition that size of input is bounded. This implies that the computation for any of the specified inputs terminates.

For logic programs, loops occur through recursion and the size of a term is bounded if it is rigid and the size of term does not change under any instantiation. Because of that, analyzers consider two factors: size of terms to detect decrease and degree of instantiation to detect rigidity. Proving termination is based on suitable norm and size of instantiation provided that all loop are decreasing and bounded. For practical reasons, most termination
analyzers choose the natural numbers as the well-founded domain and measure size using so-called linear or semi linear norms.

We discuss termination analysis of the algorithm by applying the well-practiced technique for ‘termination analysis of recursive functions’. We attempt to demonstrate that there is a

We discuss termination analysis of the algorithm by adopting a well-practiced technique for ‘termination analysis of recursive functions’ - where the central idea is to ensure that there exists a well-founded ordering so that the argument of each recursive call is ‘smaller’ (or ‘greater’) than the corresponding inputs. This comparison is done in terms of a ‘measure’ (an element of the well-founded set), which decreases (or increases) after each recursive-procedure execution. A ‘measure-function’ needs to be defined so that it can map a data-object (which is related to the corresponding recursive-function’s input) to a member of a well-founded ordered set.

8.2 Termination proof

The manipulated program contains the recursive functions that have all the conditions for termination. The defined function is in below format:

\[
\text{rec[]} = []
\]
\[
\text{rec}(a:as) = \text{rec as} ++ [a]
\]

The length of “rec” list decreases by one element is well-formed ordinary founded limit. According the provided definition, the size of “rec as” is smaller by one than “rec(a:as)” so we can claim that the size of list decreases by one in each computation. The base case is also defined in the recursive function that shows when the size of list is empty or the list does not have any members, it is equal to empty set. The above function meets the base case definitely because the decreased number of size of list is one. It happens the size of list is zero and the list is equal to empty set and that it is terminate point of function.
8.3 Time complexity

In this section, it is shown that the worst case time complexity for accommodation of prepositional phrase in natural language queries to triplestore database is $O(n^2)$. In the developed programs that locate the prepositional phrases, there is recursive function:

$$
\text{rec} \; [] = []
$$

$$
\text{rec} \; (a:as) = \text{rec} \; \text{as} ++ a
$$

The time complexity for the above recursive function is $O(n^2)$. In order to determine worst case time complexity of above function, assume that $n$ is the size of list. Thus, $\text{rec}(a:as)$ has to be called $n$ times to be empty and its size turn to zero because the list size decrease by one in each call so the $\text{rec}(a:as)$ is called $n$ times.

![Figure 3](image.png)
On the other hand, appending the member “a” to the list needs “n” times complexity in worst case. Thus, the worst case time complexity for recursive function is $O(n^2)$ totally.
Chapter 9: Claim and Conclusion

9.1 Claim

We can claim that is the first and only approach which is able to accommodate prepositional phrases based on Montague Semantic over triple store database. Based on recommended approach the coverage of queries which can be answered over triple store will be increased.

9.2 What has been achieved?

It is possible to construct natural language query interfaces to triplestores which can accommodate arbitrarily-nested complex chained prepositional queries such as:

- “When did Capone steal a car in Manhattan using a crowbar…?”
- “Where did Capone steal a car in 1918 or 1920…?”
“With what did Capone steal two cars and a truck that was owned by a gangster who joined every gang in Brooklyn that was joined by Torrio?”

9.3. Relationships to parsing and access to triplestores

Our proposed approach cannot provide answers if the query is in the following format:

“When did Capone steal a car?”

The above query should be converted to the below format:

When (Capone (steal (a car)))

In order to convert the natural language to form that can be acceptable to our system and provides appropriate answer, the format needs to be modified by the mentioned way with the time of complexity of \(O(n^3)\).

9.4 Contribution to Computational Linguistics and Computer Science

Our research has contribution to natural language interface to event-based triplestore. Our proposed approach increase the coverage of queries can be queried over database by natural language. None of researchers have studied how accommodate prepositional phrases by Montague’s semantic over database contains triples of data. We changed the semantics to events that can handle the studied issue and can answer questions which are looking for subject, object, and prepositional phrase etc.

9.5. Limitation

The proposed approach is able to find the prepositional phrases only, it cannot answer all type of questions of “When” and “Where”. If the fact is like “Capone married after John was born”, and query asks for “When did Capone marry?” cannot be provided with appropriate answer since it is looking for the prepositional phrases and then try to provide the answers based on that. The other limitation is that all the facts have to model to triplestore and needs a software convert the fact to triplestore that can be used as storage data. Converting to triplestores which is based on event requires software otherwise our proposed approach which is event-based cannot accommodate the prepositional phrases.
9.6. Conclusion and future work

In this thesis, the event based approach that has been proposed by Frost et al. [2014] has been completed by converting all the definitions that are based on entities to events. Accommodation of prepositional phrases in natural language queries over database to triplestore database is provided in this thesis by changing and modifying whole definitions to event based definitions. It is the first proposed approach that can find prepositional phrases in natural language queries over triplestore database. Our research group plan to extend the semantics to accommodate aggregations and negations and also integrate the semantics with a parser using the SAIGA attribute grammar programming environment [16].
REFERENCES:


[23] Partee, B. Montague, Richard (1930-71)


APPENDIX I - PROGRAM

point ::= EV num | ENT [char]
         | ENTNUM num | TYPE [char]
         | REL [char] | ANY

data = [(EV 1000, REL "type", TYPE "born_ev"),
        (EV 1000, REL "subject", ENT "capone"),
        (EV 1000, REL "year", ENTNUM 1899),
        (EV 1000, REL "location", ENT "brooklyn"),
        (EV 1001, REL "type", TYPE "join_ev"),
        (EV 1001, REL "subject", ENT "capone"),
        (EV 1001, REL "object", ENT "fpg"),
        (EV 1002, REL "type", TYPE "membership"),
        (EV 1002, REL "subject", ENT "capone"),
        (EV 1002, REL "object", ENT "thief"),
        (EV 1002, REL "year", ENTNUM 1908 ),
        (EV 1003, REL "type", TYPE "join_ev"),
        (EV 1003, REL "subject", ENT "capone"),
        (EV 1003, REL "object", ENT "bowery"),
        (EV 1004, REL "type", TYPE "steal_ev"),
        (EV 1004, REL "subject", ENT "capone"),
        (EV 1004, REL "object", ENT "car_1"),
        (EV 1004, REL "year", ENTNUM 1908),
        (EV 1004, REL "location", ENT "brooklyn"),
        (EV 1005, REL "type", TYPE "smoke_ev"),
        (EV 1005, REL "subject", ENT "capone"),
        (EV 1006, REL "type", TYPE "membership"),
        (EV 1006, REL "subject", ENT "car_1"),
(EV 1006, REL "object",  ENT "car"),
(EV 1007, REL "type",  TYPE "membership"),
(EV 1007, REL "subject",  ENT "fpg"),
(EV 1007, REL "object",  ENT "gang"),
(EV 1008, REL "type",  TYPE "membership"),
(EV 1008, REL "subject",  ENT "bowery"),
(EV 1008, REL "object",  ENT "gang"),
(EV 1009, REL "type",  TYPE "join_ev"),
(EV 1009, REL "subject",  ENT "torrio"),
(EV 1009, REL "object",  ENT "fpg"),
(EV 1010, REL "type",  TYPE "membership"),
(EV 1010, REL "subject",  ENT "capone"),
(EV 1010, REL "object",  ENT "person"),
(EV 1011, REL "type",  TYPE "membership"),
(EV 1011, REL "subject",  ENT "torrio"),
(EV 1011, REL "object",  ENT "person"),
(EV 1012, REL "type",  TYPE "steal_ev"),
(EV 1012, REL "subject",  ENT "capone"),
(EV 1012, REL "object",  ENT "car_2"),
(EV 1012, REL "year",  ENTNUM 1908),
(EV 1012, REL "location",  ENT "brooklyn"),
(EV 1013, REL "type",  TYPE "membership"),
(EV 1013, REL "subject",  ENT "car_2"),
(EV 1013, REL "object",  ENT "car"),
(EV 1014, REL "type",  TYPE "steal_ev"),
(EV 1014, REL "subject",  ENT "torrio"),
(EV 1014, REL "object",  ENT "car_3"),
(EV 1015, REL "type",  TYPE "membership"),
(EV 1015, REL "subject",  ENT "car_3"),

59
(EV 1015, REL "object", ENT "car")]

firsts = map first
first (a,b,c) = a
second (a,b,c) = b

thirdwithfirsts = map thirdwithfirst
thirdwithfirst (a,b,c) = (c,a)
thirds = map third
third (a,b,c) = c

year_of ev = thirds (getts (ev, REL "year", ANY))
years_of evs = concat (map year_of evs)

new_when ev_image
= [(subj, years_of evs) | (subj, evs) <- ev_image; years_of evs~=[[]]

new_capone ev_image
= [(subj, evs)
   | (subj, evs) <- ev_image; subj = ENT "capone"]

new_get_members set
= [[[get_subj_for_event ev)!0, [ev]] | ev <- events]  
  where
  events_for_type_membership
  = firsts (getts (ANY, REL "type", TYPE "membership"))
  events_for_set_as_object
  = firsts (getts (ANY, REL "object", set))
events = intersect events_for_type_membership
  events_for_set_as_object

new_smoke = make_intrans (TYPE "smoke_ev")

make_intrans et
  = collect bin_rel
where
  bin_rel
    = [(thirds (getts(ev, REL "subject", ANY))!0,ev)
        | ev <- events]
  events = firsts (getts (ANY, REL "type", et))

new_steal_intrans = make_intrans (TYPE "steal_ev")

new_join_intrans = make_intrans (TYPE "join_ev")
new_gang = new_get_members (ENT "gang")
new_person = new_get_members (ENT "person")
new_car = new_get_members (ENT "car")
new_thief = new_get_members (ENT "thief")
new_gangster = new_get_members (ENT "gangster")

new_termand tmph1 tmph2
  = f  where
    f ev_image  = res1 ++ res2, if (res1 ~= []) & (res2 ~= [])
              = [], otherwise
where
  res1 = tmph1 ev_image
  res2 = tmph2 ev_image
new_termor tmph1 tmph2 = f  where
    f ev_image = tmph1 ev_image ++ tmph2 ev_image

new_a nph vph = [(subj2, evs2) | (subj1, evs1) <- nph;
                     (subj2, evs2) <- vph;
                     subj1 = subj2]

new_every nph vph = result, if subset (map fst nph) (map fst vph)
                    = [], otherwise  
    where   result = [(subj2, evs2) | (subj1, evs1) <- nph;
                        (subj2, evs2) <- vph;
                        subj1 = subj2]

new_image_steal = collect
    [(subj, (obj, [ev])) | ev <- events;
     subj <- thirds (getts (ev, REL "subject", ANY));
     obj  <- thirds (getts (ev, REL "object", ANY))]
    where events = firsts (getts (ANY, REL "type", TYPE "steal_ev"))

new_image_join = collect
    [(subj, (obj, [ev])) | ev <- events;
     subj <- thirds (getts (ev, REL "subject", ANY));
     obj  <- thirds (getts (ev, REL "object", ANY))]
    where events = firsts (getts (ANY, REL "type", TYPE "join_ev"))

new_steal tmph
= [[subj, concat(map snd (tmph objs_evs)))] |
  (subj, objs_evs) <- new_image_steal;
  tmph objs_evs ~= [] ]

test1 = new_steal (new_a new_car)

new_join tmph = [[subj, concat(map snd (tmph objs_evs)))] |
  (subj, objs_evs) <- new_image_join;
  tmph objs_evs ~= [] ]

test2 = new_capone (new_steal (new_a new_car))

test3 = new_capone (new_steal (new_every new_car))

new_torrio ev_image t
= [[subj, evs]
  | (subj, evs) <- ev_image; subj = ENT "torrio"]

new_torrio ev_image t

10 = new_when (new_torrio (new_steal (new_a new_car)))

11 = new_join (new_a new_gang)

17 = new_join (new_every new_gang)

test22 = new_capone (new_join (new_a new_gang))

test24 = new_when (new_capone (new_join (new_a new_gang)))

test25 = new_a new_person (new_join (new_a new_gang))

test26 = new_every new_person (new_join (new_a new_gang))
\[\text{test27} = \text{new\_torrio} (\text{new\_join} (\text{new\_a new\_gang}))\]
\[\text{test28} = \text{new\_torrio} (\text{new\_join} (\text{new\_a new\_gang}))\]
\[\text{test29} = \text{new\_when} (\text{new\_torrio} (\text{new\_join} (\text{new\_a new\_gang})))\]

\[\text{new\_image\_born}\]
\[= \text{collect}\]
\[
[(\text{subj},[\text{ev}]) \mid \text{ev} \leftarrow \text{events};
\text{subj} \leftarrow \text{thirds (getts (ev,REL "subject", ANY))}]\]
\[\text{where}\]
\[\text{events}\]
\[= \text{firsts (getts (ANY, REL "type", TYPE "born\_ev"))}\]
\[\text{new\_born} = \text{make\_intrans} (\text{TYPE "born\_ev"})\]
\[\text{test30} = \text{new\_capone} (\text{new\_born})\]
\[\text{test31} = \text{new\_when} (\text{new\_capone} (\text{new\_born}))\]
\[\text{test32} = \text{new\_torrio} (\text{new\_born})\]
\[\text{test33} = \text{new\_when} (\text{new\_torrio} (\text{new\_born}))\]
Appendix II – Survey

Accommodating Prepositional Phrases in Natural Language Query Interfaces to the Semantic Web

Elham Emami, University of Windsor

Using natural language interfaces has been most popular and widespread nowadays because of its ease of use and speed. There are some of them query over the semantic web data which data is represented and stored in the triplestore format like DBpedia. One of the challenges addressed by this type of data interfaces is how to accommodate prepositional phrases in natural language query interface to triplestore data. Event-based triplestore approach proposes a solution to cover and solve the problem but it cannot provide wideness of query coverage with prepositional phrases content for users. There are not a lot and diverse research which have attempted to solve the locating prepositional phrases in NLQI to semantic web problem and improve it. This survey is about the related research and work.

Additional Key Words and Phrases: semantic web, natural-language query, preposition or prepositional phrase.
1. INTRODUCTION

The Resource Description Framework (RDF) is a specifications originally designed as a metadata data model, which is growing semantic web standard for the ontologies specification. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats.

The current survey attempts to look through processor’s methods to retrieve accurately result to users query to semantic web which contains prepositional phrase. Much researches attempts have been done to retrieved much more appropriated and accurate result to user’s queries but in majority of them the importance of the prepositional phrases have been ignored and instead concentrate on the subjects, verbs, etc. which decrease the accuracy and precision of their proposed approach.

If the mentioned issue which is accommodating prepositional phrases in natural language queries to semantic dataset is resolved, natural language interfaces can retrieved much more related result to queries and boost their performance. Thus, it can help to construct more powerful natural language interfaces to databases.

Relevant research papers were found by searching Google scholar with provided keyword "semantic web" "natural-language query" preposition OR prepositional” and guidance and help of Dr. Frost. Various keywords and authors names have been used to find and search much more related papers on ACM publication library, IEEE, LNCS, Scopus, etc.

Three journal papers, eight conference papers, one doctoral theses, and one survey which are closely related to this survey were identified. They are listed in the bibliography.

Thirteen papers have been selected as the basis of this survey. The current survey look through at these papers which have worked directly to how locate prepositional phrases in the natural language queries to semantic web. They introduced exactly the importance of the preposition phrases accommodation in triplestore data and how can be located by their proposed approaches. On the other hand, on the other hand there are some
papers which have attempted to answer “When …? ” and “Where …?” type questions. Their method can be studied in current survey in the aspect of locating spatial and temporal data and being used to inspire for accommodate prepositional phrases.

Papers referred Wang, Xiong, Zhou & Yu [2007] have been selected since it gets the pure natural language and then in the some steps of processing it deals with triplestore so it can be related to the survey topic, Frost, Amour and Fortier [2013] and Frost, Agboola, Matthews & Donais [2014] have been chosen since they consider prepositional phrases issues in natural language to semantic web directly which is exactly on the survey topic. The papers referred by Bernstein, Kaufmann, Fuchs, and Bonin [2004] introduced a new approach to convert English which can be used for query an ontology and can be inspired to accommodate the prepositional phrases. The paper referred by Stoermer[2006] attempted to explain that RDF which is being used for describing the triplestore has some lacks and try to cover its problem by proposing a solution which is extension of RDF knowledge bases with context features. This paper can be used for knowing what the shortcomings of the RDF is. Kolas & Self [2007] paper has been selected since it looks through the spatial attributes and describes a prototype system for storing spatial data and Semantic Web data together in a SPatially-AUgmented Knowledgebase (SPAUK) without sacrificing query efficiency. Paper referred by Cimiano & Minock [2009] has been selected since it tries to solve the quantitative problems in natural language and also in some parts it tries to detects the different types of prepositional phrase like light preposition and spatial prepositions. Paper referred by M. Dylla, M. Sozio, and M. Theobald [2011] has been chosen because it proposed a declarative and reasoning framework to express and process temporal constraints and queries via first-order logical predicate. So the problem of finding a consistent subset of time-annotated facts to a scheduling problem can be reduced. Paper referred by Analyti, & Pachoulakis [2012] also has been chosen since it is itself a survey and can be organized and knowledgeable source for getting essential information about how to locate prepositional phrases.

The remainder of this survey is structured as follows: The first part of section 2 focuses on papers which have attempted to look at the temporal and spatial attributes in the
statements which have not exactly on the prepositional problems in the natural language query to semantic web but can be well-defined solution which can be inspired in future to provide proficient solution to prepositional phrases issue.

The second part of section 2 contains reviews of three papers which have researched exactly on topic and proposed novel approach which can be helpful for improving the performance and accuracy of the natural language query processors to semantic web by accommodating prepositional phrases properly. Section three contains conclusion.

Although using natural language for querying to semantic web is growing and the semantic web is popular, not that much papers and researches work on this area to increase the precision and accuracy of results.

2. Approaches:

In this survey the papers can be grouped in two categories, the first one is related to Dr. Frost papers which are exactly on topic and survey target and the second category contains the published papers which are related to spatial and temporal problems which their result and introduced approaches can be used and shed more light for accommodating prepositional phrases in natural language query interfaces to the Semantic Web. The major reason of these categorization can be their view to data and metadata of data base and methods which are totally different but can be helpful for this survey concern.

2.1 Accommodation of prepositional phrases

In this section papers have been considered which have tried to accommodate prepositional phrases in natural languages to Semantic Web.

2.2. Spatial and Temporal Attribute

In this section the papers have been studied which attempted to locate to address some problems regarding spatial and temporal attributes in set of statements which have been referenced by context.
2.1.1. A Controlled English Query Interface for Ontologies

According to Bernstein et al. [2004], the semantic web contains the dramatically growing knowledge base which should allow users to query over this huge amount of data with formal logic and query which can be also easy for users leaning of that query language. There is huge gap between the knowledge of the users and available query languages so far so because of that reason it is being difficult for them to query over them. The current paper tried to addressed this problem and bridge the gap between the logic-based semantic web and real-world users. They refer as Popescu et al. [2003] which the authors have not pointed to any shortcoming of previous work but they stated that they could not find any other application of controlled natural language querying of semantic web content. Furthermore, they stated that work on natural language interfaces to data bases has largely tapered off since the 80’s.

The authors proposed their new approach by introducing ACE which is small subset of the English meaning which each of the ACE sentence is correct, also ACE contains English grammar which is contains set of constructions and interpretations rule. For being simple semantics and syntactical complexity of the English is removed from the ACE. APE, Attempto Parsing Engine, which has been implemented in Prolog as a Define Clause Grammar translate the ACE to DRS- discourse representation structure that logically represent the text. A DRS consists of discourse referents such as quantified variables which represents the objects of a discourse, and of conditions for the discourse referents. The conditions can be logical atoms or complex conditions built from other DRSs and logical connectors like negation, disjunction, and implication.

Afterward the DRS is translated to semantic web query language PQL which can be used for query an ontology. PQL’s two major statement types are ATTRIBUTE and RELATION.

To validate of the proposed approach Bernstein et al. [2004] combined Prolog and Java components, as APE and the rewriting framework are programmed in SICStus Prolog, and the user interface and the query engine are also programmed in Java. They executed number of real-world queries and then compared its retrieval performance with two keyword-based retrieval approaches, both of those approaches have proven that the output
is suitable for end-user. Also the authors claim that their proposed approach can generate appropriate and expected PQL.

### 2.1.2. PENG-D for Controlled Natural Language

The next paper which has been studied is entitled “CLOnE: Controlled Language for Ontology Editing” which has been referred by Schwitter et al. [2004]. The authors stated that the number of querying by natural language is increasing sharply on the semantic web. However, there are some suggested format to retrieve the documents and resources on the web, users have to be familiar with complicated structure and format of the storing and retrieving interfaces. They attempted to provide a solution which retrieve the related information regarding submitted query by user by presenting the PENG-D, a proposal for a controlled natural language. The authors noted that they inspired some previous work like Schwitter [2002] and Schwitter [2004] in papers “. English as a Formal Specification Language” and “Representing Knowledge in Controlled Natural Language” but they have some defects and shortcomings. They mentioned that RDF schema is being used to preserve and address the web resources which suffers of limitation in knowledge representation with few modeling primitives and more expressive power for defining web resources. Furthermore, there is no any unique standard layer meta-modeling architecture which be able to assign two or more roles to elements in RDF specification so this makes it extremely difficult to layer more expressive ontology and rule languages on top of RDFS. “Controlled Natural Languages are subsets of natural language whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity.” Schwitter [2007].

The paper referred by Schwitter et al. [2004] mentioned that it is addressing new approach which is called PENG-D. Basically PENG Schwitter [2002], Schwitter [2003], Schwitter [2004] is a machine-oriented controlled natural language that has been developed to write specifications for knowledge representation. While PENG was designed for writing specifications that are first-order equivalent, the proposed language PENG-D has formal properties that are equivalent to DLP. PENG-D provides a clear computational pathway to layer more expressive constructions on top of it. The planned architecture of
the PENG-D system looks similar to the PENG system but offers support for ontology construction. The PENG-D system consists of four main components: a look-ahead text editor, a controlled language (CL) processor, an ontology component and an inference engine.

Since the authors are considering the way for prepositional phrase in natural language in survey so suggested approach for preposition phrase which has been mentioned by the authors in paper referred by Schwitter et al. [2004] is provided as follow.

The authors state that finding the prepositional phrase is like the complement statement which has been done by most of the syntactic structures that are approved for the subject position. It allows for the prepositional construction has ... as ... and for coordinated structures:

Nic is married to Sue.

Nic has Rex as dog.

Nic has Rex as dog and Tweety as bird.

The authors have not provided exact experiment and the result of that but they pointed that their proposed approach, PENG-D, which is a machine-oriented controlled natural language that has same expressivity as DLP. DLP offers a promising first-order based alternative that enables ontological definitions to be combined with rules. In this paper we referred to a number of deficiencies of RDFS as a “knowledge representation” language for the envisioned Semantic Web. Layering more complex ontology and rule languages on top of RDFS is not straightforward, because of its non-standard and non-fixed layer meta-modelling architecture. The relatively new DLP paradigm offers a promising first-order based alternative that enables ontological definitions to be combined with rules. To make such machine-processable information easily accessible for non-specialists, the authors also proposed the use of PENG-D, a machine-oriented controlled natural language that has the same expressivity as DLP.

The authors claim that they referred to number of deficiencies of RDFS as a “knowledge representation” language for the envisioned Semantic Web. Layering more
complex ontology and rule languages on top of RDFS is not straightforward, because of its non-standard and non-fixed layer meta-modelling architecture. Furthermore they claim that PENG-D is easy to write for non-specialists with the help of a look-ahead text editor, easy to read in contrast to RDF-based notations, and easy to translate into a corresponding machine processable format. In brief: PENG-D has the potential for complementing these more machine-oriented notations.

2.1.3. Context into Semantic Web Knowledge Bases
The author in paper cited by Stoermer et al. [2006] address that Knowledge Representation point of view RDF statements in general are context-free, and thus represent statements of universal truth, while documents contain context sensitive information. This paper can be useful since the way which looks at extracting context can be figure out and be used in ways for accommodating prepositional phrases. One small example can be two contradicted statement such as “Silvio Berlusconi is the Prime minister of Italy” and “Romano Prodi is the Prime minister of Italy”. This is very small contradiction which can be handled by some defined approach but in Semantic Web terms and in large number of uncoordinated information systems, it could be a serious problem.

The author believes that such contradictions, contradictory beliefs and facts which become semantically incorrect in the absence of additional pragmatic or contextual information are likely to impose serious problems on the coordination and interoperation of information systems in the Semantic Web. Stoermer states that the previous work has not used reification but implements context as a real extension of the RDF model theory, by moving from triples to quadruples for identifying the context to which a statement belongs. These previous ideas have not been pursued any further. Moreover all the currently available RDF tools would have to be extended in order to deal with such an RDF model.

All the statement which belong to a context should be presented in separate named RDF graph and the graph can be extended in a way which contexts can be appeared as standard object in RDF statement. Then the author tries to connect the two context to allow for reasoning across contexts. This aspect is seems to important and crucial since sensible queries can be issued and all relevant information is taken into account. So many
approaches have been thought for modeling these relations. One of them is the vocabulary of the context can be provided by implementer to describe relations between contexts. The compatibility relations are supposed to be modeled in following ways: 1. provide some limited relations and require all other systems which implement this system to take care of the defined relation. 2. Provide an ontology for context relations, so that there exists a vocabulary to describe these relations with the help of RDF. This approach is slightly more flexible, because the ontology is extendable. 3. Define a CR to be implemented as a semantic attachment, which can be thought of as a sort of plugin to the system.

Author states that is trying to establish close collaboration with two other research groups which can be used as experiment of this paper. One of the groups is responsible for developing the mentioned RDF triplestore and they are currently working on an implementation based on RDF named graphs.

Stoermer claims to propose a detailed solution to the problem of modeling contexts in the Semantic Web in a coherent and general way and also an evaluation of the MCS theory. Also mentioned it is able to put this theory to the test, and explore its limitations. Moreover, Provision of comparative experimentation results, to illustrate which possibilities exist, how they behave and whether they prove appropriate for real-world applications is other authors’ claim.

2.1.4. Spatially-Augmented Knowledgebase

One of the major problem which has been addressed in paper by Kolas et al. [2007] The authors state that RDF and modern triplestores are efficient at storing and querying data linked across multiple sources of information but they are not that much efficient it comes to spatial processing. The standard for storing spatial data involves object-oriented database with spatial features. But one of the draw backing of the object-relational model is that it does not have flexibility of RDF and triplestores that make them attractive for searching linked data across multiple sources. The authors pointed that mapping queries that include spatial instances and relationships to SPARQL is not easy. There are many possible ways that one could use SPARQL for spatial data, and the ideal way has yet to be found.
The authors propose SPAUK who’s its goal is to provide efficient spatial processing for spatial semantic systems. They chose SPAUK as a semantic knowledge base capable of supporting supplementary spatial indices.

Additionally, designing a system such that the addition of spatial processing to the system can be as transparent as possible to the user is the secondary goal of designing the SPAUK.

All data including semantic data and spatial data, is still presented as a graph. The authors mention that the knowledge base presents itself as a standard SPARQL endpoint. This allows any clients capable of interfacing via the SPARQL protocol to utilize SPAUK. The design should present one conceptual graph to its clients, and then queries can be over this graph by dividing appropriately into sub-queries which can be answered by the various parts of the knowledgebase. Spatial parts which includes locations and spatial relationship

Spatial parts of the query, including locations and spatial relationship should be sent to the spatial index and query processors. The nonspatial part of the query must be sent to the related triplestore. Results must be combined from the two parts to form a related answer. Furthermore, data which is inserted must find its way into the appropriate parts of the knowledge base.

The architecture of SPAUK is based on the Jena Semantic Web Framework and Joseki.

Although the formal analysis of the performance of using a supplemental spatial index in object-relation has not been done yet and only simple analysis of the algorithms involved, the authors claim that preliminary usage of the SPAUK system have shown that the approach is valid.

At the end of the paper, Kolas, Self [2007] mention that by attaching a semantic GIS client to the SPAUK system provides responsive spatial semantic query capability. They believe that this type of system enables a new class of semantic applications whose full potential cannot yet be conceived.

2.1.5. Natural Language Interfaces and a data-driven quantitative analysis

In the paper referred by Cimiano et al. [2009] the authors addressed the problem of how to accommodate different constitutive part of entered queries efficiently and improve the
precision and accuracy of the NLIs performance in answering the questions. One of the discussed part of speech queries which have been attempted to locate it is prepositional phrases which have been ignored in so many previous work of the natural language area research. The authors mentioned that the previous work referred by Cimiano et al. [2007] have not paid attention in particular quantitative analysis problems inherent in the task building natural language interfaces They stated that although there have been qualitative analysis of the problems involved in constructing NLIs, there has been no quantitative analysis grounding the qualitative characteristics of the problem in real data. The mentioned area can guide the system developers in the future also it can help them to focus on specific phenomenon encountered in NLIs and progress easily in the field by clearly designing and evaluating the solution to a specific phenomenon.

The authors noted to provide a quantitative analysis of the problem of constructing an NLI, they use the Geobase database which are being used frequently for natural language interfaces. The Geobase dataset describes states, cities, mountains, lakes, rivers and roads in the U.S., together with attributes such as area (state, lake), population (state, city), length (river), and height (mountain, location) etc. The Geobase has been converted to ontology language F-Language and OWL. When converting into OWL and F-Logic has been completed, 7concepts with a total 17 various relation have been used which the concepts used with their relations are given below:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>name, abbreviation, capital, density, population, area, code hasCity, border, highest point, lowest point</td>
</tr>
<tr>
<td>city</td>
<td>name, area, inState</td>
</tr>
</tbody>
</table>
Some of the information into one class merged into one class, some redundancies have been removed, and the location class has been added which includes a height attribute for the location in question. The authors noted that they manually created F-Logic queries yielding the appropriate answers as result when evaluated with the OntoBroker system, but also queries in generic logical form. The most important point of using F-Logic language in the OntoBroker system is that built-in functionality is provided for numerical comparison as well as aggregation operators for calculating minim, maximum, sums, etc. They attempted to annotate each of the questions together with characteristics that they regarded as relevant to our quantitative analysis.

The language which is used in the questions is rather simple, and contains a lot of 'light syntactic constructions' such as: Light preposition “of” (appearing in 21.52% of the sentences): What are the highest points of all the states? Or what are the major cities of texas? Light preposition “with” (appearing in 7.36% of the sentences): How many people live in the state with the largest population density? Or what are the cities of the state with the highest point? This can be conducted that relevant relations are not expressed obviously and are hidden implicitly behind the light constructions like prepositions “with” and “of”. Because of that so many shallow approaches ignore to consider linguistic details like prepositions phrases. Furthermore, they recognize that there are so many syntactic ambiguities which in case of prepositional phrases they also distinguish the case of prepositional phrase providing essentially the predict in copula construct.

The experiments show that by using PPs to the last constituent, correct decision in 99.27% of the cases for PP attachment including last and only attachment point cases as well as the copula case where the PP functions as predicate is predictable.
In addition to light and vague prepositions, there are some spatial prepositions like “in”, “next”, “though”. Although some systems have shown that their performances are acceptable by essentially ignoring prepositions, the more principled and successful solutions would capture the domain independent meaning of such spatial prepositions, allowing to reuse their meaning across domains. In the context of an ontology about American presidents, it makes a difference whether we ask for `Who was president after WWII', vs. `Who was president during WWII'. In general, systems would thus profit from capturing the meaning of prepositions explicitly, possibly even assigning them a domain independent meaning.

The authors mentioned they have started work in this area since a data-driven analysis has been missing so far while there have been many qualitative descriptions of the problems.

Basically Cimiano et al. [2009] believe that is not acceptable to neglect the prepositional phrases and some other details in natural language processing, because it can cause some unpredicted problems and require adhoc and principled extensions. The authors believe that constructing systems with deep syntactic and semantic processing capabilities will pay off in the long term.

2.1.6. Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases

The paper referred by Dylla et al. [2011] stated the knowledge base is inconsistent generally and not sufficient constraints are applied to knowledge base. Expectedly so many contradicts can be existed in knowledge base, to resolve these inconsistencies some form of consistency should be imported to knowledge base. Temporal annotation can be one of the most useful and efficient constraint which can not only express general constraints among facts but also add a finer granularity to the consistency reasoning itself. Furthermore, they stated that even when using simple time intervals for the representation of temporal annotations with such precedence constraints, the satisfiability problem is known to be NP-hard. The author of paper noted that the rules which have been considered in previous work referred by Wang et al. [2010] do not consider the inclusion of actual consistency constraints, where only some facts out of a given set may be set to true while other facts are considered false. The other shortcoming of pervious work which author
addressed is that they only consider positive lineage for example conjunctions and disjunctions only.

Dylla et al. [2011] introduce their new idea by defining a knowledge base $\mathbf{KB} = (F; C)$ as a pair consisting of a set of facts $F$ which each of them has weighted and temporal and a set of first-order consistency constraints $C$. To encode facts, they employed the used RDF, in which facts $F \in \text{Rel} \times \text{Entities} \times \text{Entities}$ are stored as triples consisting of a relation and a pair of entities. Moreover, the original RDF triplet have been extended structure in two ways: first, and second, to include time information into knowledge base, also assign a time interval of the form $[t_b; t_e)$ to each fact $f$.

They also provide some format of consistency constraint:

$$relE_1 (e_1; e_2; t_1) \land relE_2 (e_1; e_3; t_2) \land relA(e_2; e_3) \Rightarrow relT (t_1; t_2)$$

The authors mentioned that based on the type of constraint, the combinational complexity of resolving conflict is different and selecting which constraints should be formulated will be so crucial.

There are three different type of constraint which includes:

- Temporal disjoint
- Temporal precedence
- Mutual exclusion

They utilize the following template to express disjointness constraints.

$$relE(e_1; e_2; t_1) \land relE(e_1; e_3; t_2) \land e_2 \neq e_3 \Rightarrow disjoint(t_1; t_2)$$

They employ following template for precedence constraints shows the restriction of the time interval of an instance of $relE_1$ ends before the interval of a fact with $relE_2$ starts

$$relE_1 (e_1; e_2; t_1) \land relE_2 (e_1; e_3; t_2) \Rightarrow before (t_1; t_2)$$
They use the following temporal constraint for mutual exclusion. Mutual exclusion defines a set of facts which are all in conflict with each other, regardless of time.

\[
\text{relE}(e1; e2; t1) \land \text{relE}(e1; e3; t2) \land e2 \neq e3 \Rightarrow \text{false}
\]

The authors mentioned that their work has been implemented by Java 1.6 in about 3k lines of code. Additionally, they employed a greedy heuristic for the MWIS, which proved to perform best on the data among all the greedy methods which have been tried before. There are other means of approximating the MWIS problem, like stochastic optimization. However, they are even less scalable than greedy methods.

The dataset which they used contains data about the playsForClub, playsForNational, and hasWonPrize relations, which has been extended manually by dates of birth and death. Their algorithm showed impressive robustness. Moreover, the histogram of scheduling algorithm exhibits excellent behavior as in nearly every problem instance the optimal solution was found. The authors claim their approach works by identifying a subclass of first-order consistency constraints, which can be efficiently mapped to constraint graphs and be solved using results from scheduling theory.

The authors claim that their experiments show that applied approach performs superior to common heuristics that directly operate over the underlying Maximum Weight Independent Set problem in terms of both run-time and quality.

### 2.1.7. Models and Query Languages for Temporally Annotated RDF

The authors in paper referred by Analyti et al. [2012] provide a survey on the models and query languages for temporally annotated RDF.

The authors note that in most of the previous works, a temporally annotated RDF ontology is essentially a set of RDF triples associated with temporal constraints, where, in the simplest case, a temporal constraint is a valid temporal interval. Also the authors address
that problem that some of the works provide experimental results while the rest are purely theoretical.

The authors have considered the following papers: Hurtado et al. [2006], Perry et al. [2007], Gutierrez et al. [2007], Pugliese et al. [2008], McBride et al. [2009], Tappolet et al. [2009], Grandi [2010], Perry et al. [2011].

The researcher mentioned that one of previous work that has their own model theory is a tRDF query over a tRDF database \( D \) is a set of triples of the form \((s, p:\{T\}, o)\), \((s, p:<n:T>, o)\), \((s, p:[n:T], o)\), where \(s, p, o, T\) are possibly variables, with the constraint that each temporal variable appears only once. The works that extend RDFS entailment seems less efficient since it computes the RDFS closure of RDF triples at each time point.

This paper is a survey referred by Analyti et al. [2012] compares different approaches which the new idea of each of them is as follow:

A. The paper referred by Pugliese et al. [2008] proposes an approach which a temporal RDF (tRDF for short) database is a set of triples of the form \((s, p:\{T\}, o)\), \((s, p:<n:T>, o)\), \((s, p:[n:T], o)\), and tRDF query over a tRDF database \( D \) is a set of triples of the form \((s, p:\{T\}, o)\), \((s, p:<n:T>, o)\), \((s, p:[n:T], o)\), where \(s, p, o, T\) are possibly variables. The authors of survey mentioned that that paper presented an efficient algorithms for simple and conjunctive query answering, showing that the time complexity for answering a conjunctive query is in \( O(|R|^2*|P|)|Q| \), where \(|Q|\) is the number of simple queries in \( Q \).

B. In papers referred by Tappolet et al. [2009], the authors expressed instead of having RDF triples graphs are used both for saving space and for querying the temporal RDF database using standard SPARQL. The authors introduce through examples a query language, named \( \tau \)-SPARQL which extends the SPARQL query language for RDF graphs. Each \( \tau \)-SPARQL query can be translated into a SPARQL query. A \( \tau \)-SPARQL query that retrieves all foaf:Persons whose lifespan overlaps with Einstein’s is:

\[
\text{SELECT } ?s2, ?e2 \text{ ?person WHERE }
\text{[?s1, ?e1] \text{foaf:name “Albert Einstein”}}
\text{[?s2, ?e2] \text{time:}intervalOverlaps [?s1, ?e1]}
\text{[?s2, ?e2] \text{foaf:Person.}}
\]\n

This query is translated into a SPARQL query, as follows:

```sparql
SELECT ?s2 ?e2 ?person
WHERE{
  GRAPH ?g1 {?einstein foaf:name "Albert Einstein"}.
  ?g2 time:intervalOverlaps ?g1.
  GRAPH ?g2 {?person a foaf:Person.}
  ?g2 time:hasBeginning ?s2.
  ?g2 time:hasEnd ?e2.
}
```

The authors proposed an index structure for time intervals, called keyTree index, assuming that triples within named graphs have indices by themselves. The proposed index improves the performance of time point queries over an in-memory ordered list that contains the intervals’ start and end times.

In paper referred by Rodriguez et al. [2009] introduced the time-annotated RDF framework which is proposed for the representation and management of time-series streaming data. In particular, a TA-RDF graph is a set of triples <s[tS],p[tp], o[to]>, where <s,p,o> is an RDF triple and tS, tp, and to are time points. In other words, a TA-RDF graph relates streams at certain points in time. To translate a TA-RDF graph into a regular RDF graph, a data stream vocabulary is used, where dvs:belongsTo is a property that indicates that are source is a frame in a stream, dvs:hasTimestamp is a property indicating the timestamp of a frame, and dvs:Nil is a resource corresponding to the Nil timestamp.

An RDF graph G is the translation of a TA-RDF graph GTA iff (B is the set of blank nodes):

- `<s[tS], p[tp], o[to]> □ □ GTA □ ◻ ◻ rS, rp, ro`  
  - `((<rS, dvs:belongsTo, s> □ ◻ ◻ <rS, dvs:hasTimestamp, tS>) □ G□ □ rs)`  
  - `□ B) □ ◻ (tS= dvs:Nil [ □ ◻ rS=s]) □`  
  - `((<rp, dvs:belongsTo, p> □ ◻ ◻ <rp, dvs:hasTimestamp, tp>) □ G□ □ rp)`
A query language for the time-annotated RDF, called TASPARQL, is proposed which has a formal translation into normal SPARQL. The proposed system has been implemented on top of the Tupelo semantic middleware.

C. In paper referred by McBride et al. [2009], the authors considered an extension of RDFS with spatial and temporal information. In that survey, only the extension with temporal information has been considered. A set $D$ of RDF triples associated with their validity temporal interval $i$. Starting from $D$, the inference rules have been applied $A: ?i, B: ?i' \rightarrow C: ?i \cap ?i'$, where $A, B \rightarrow C$ is an RDFS entailment rule and $?i, ?i'$ are temporal interval variables, until a fixpoint is reached. Then, the temporal intervals of the same RDF triple are combined, creating maximal temporal intervals.

Based on these maximal temporal intervals, a formal extension of the SPARQL language is proposed, called SPARQL-ST, supporting however only the AND and FILTER operations. The TEMPORAL FILTER condition is precisely defined supporting all interesting conditions between temporal intervals including Allen’s temporal interval relations.

D. The other novel approach is a general framework for representing, reasoning, and querying annotated RDFS data is presented. The authors show how their unified reasoning framework can be instantiated for the temporal, fuzzy, and provenance domain.
The models and query languages of temporally annotated RDF has been reviewed. The authors believe that approaches that have their own model theory. The researchers claim that the paper referred by McBride & Butler [2009] entitled “Representing and Querying Validity Time in RDF and OWL: A Logic-Based Approach” achieve query answering using directly maximal temporal intervals achieving a higher performance. However it is not able to return maximal intervals within a temporal interval of interest. Furthermore, they claim that the paper referred by Zimmermann, Lopes, Pollere et al. [2012] which its title is “A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data” cannot always provide appropriate answer for entered query.

2.1.8. Summary

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Authors</th>
<th>Major contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2004</td>
<td>A Controlled English Query Interface for Ontologies</td>
<td>Bernstein, Kaufmann, Fuchs, and von</td>
<td>Presents a natural language front end to semantic web querying. The frontend allows formulating queries in (ACE), a subset of natural English. Each ACE query is translated into a discourse representation structure which is then translated into the semantic web querying language PQL.</td>
</tr>
<tr>
<td>2004</td>
<td>Controlled Natural Language meets the Semantic Web</td>
<td>Schwitter &amp; Tilbrook</td>
<td>Presents PENG-D, a proposal for a controlled natural language that can be used to express</td>
</tr>
<tr>
<td>Year</td>
<td>Title</td>
<td>Author(s)</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2006</td>
<td>Introducing Context into Semantic Web Knowledge Bases</td>
<td>Stoermer</td>
<td>Presents an approach which is based on the logical theory of Multi Context Systems and the principles of Locality and Compatibility. The proposed approach states that contexts can be seen in a peer-to-peer view.</td>
</tr>
<tr>
<td>2007</td>
<td>Spatially Augmented Knowledgebase</td>
<td>Kolas, Self</td>
<td>Presents a prototype system for storing spatial data and Semantic Web data together in a SPatially-Augmented Knowledgebase (SPAUK) without sacrificing query efficiency.</td>
</tr>
<tr>
<td>2009</td>
<td>Natural Language Interfaces: What is the Problem? - A data-driven quantitative analysis</td>
<td>Cimiano &amp; Minock</td>
<td>Presents a quantitative analysis for testing database</td>
</tr>
<tr>
<td>2011</td>
<td>Resolving Temporal Conflicts in Inconsistent RDF Knowledge Bases</td>
<td>Dylla, Sozio &amp; Theobald</td>
<td>Presents a declarative reasoning framework to express and process temporal consistency</td>
</tr>
</tbody>
</table>
2.2.1. PANTO: A Portable Natural Language Interface to Ontologies

In order to attain the formal knowledge in ontologies which is used to represent the taxonomy of domain and plays one of the most important role in semantic web, users are required to be familiar with some formats like the ontology syntax such as RDF and OWL, query languages such as RDQL2 and SPARQL3, the structure of the target ontology. In paper referred by Wang, Xiong, Zhou & Yu [2007] state the previous research which has been done by Bernstein, Kaufmann, Gohring & Kiefer [2007] expressed there is undeniable gap between the logic of the semantic web and real-world users. That would be too difficult for users to remember the mentioned format. On the other hand, semantic web requires ontology syntax to have expected efficiency. In paper referred by Wang, Xiong, Zhou & Yu [2007] the authors considered related paper referred by Kaufmann, Bernstein, Zumstein [2006] which they have following shortcoming and defects, the authors stated it is too tough for machines to understand the ordinary natural language because of its ambiguity and complexity. Although NLP community have been attempting to solve and they reach up to 90% in precision and recall, it has so far to recognize words and resolve it.

On the other hand, even parsing natural languages would have been done successfully, there should be some obstacles to translate them to appropriate formal queries. For example the stored vocabularies in knowledge base is totally different with user’s entered words so one of the challenges is to map the user’s words to existing vocabulary in the database. The authors mention that the other challenges in this area is how to utilize semantic information in knowledge base which includes lexical and syntactic information of parsed queries to acquire the formal query. One more thing is that various representations need different methods to interpret the user queries. Although SPARQL is
introduced as standard query language for semantic web community, novel approaches are attempting to translate natural language to them.

The new idea which Wang et al. [2007] propose includes following major steps: the nominal phrases are extracted from parse tree and they will be shown in intermediate representation which is called QueryTriples and then they are mapped to OntoTriples which are depicted with entities in ontology by PANTO and then OntoTriples will be translated to SPARQL by targets and modifiers extracted from parse tree. The authors state that they investigate some problems with translating natural language queries to SPARQL queries and some complicated and advanced features of the semantic like negation, comparative and superlative modification are considered in their new idea.

The authors implemented PANTO as stand-alone web application, it uses StanfordParse7 which produces one parse tree for each input. Test data which has been used in this experiment are based on Mooney which is being used to evaluate the natural language interface. Geography data in United States, restaurant information and job announcement are three domains in this dataset. Original Prolog database is translated into OWL as ontologies in current experiment.

The goal of the experiments is to quantitatively assess the performance and effectiveness of our approach in the current implementation.

The authors state to assess the correct rate that how many of the translated queries correctly represent the semantics of the original natural language queries,

The metrics precision and recall are being used to compare the output with the manually generated SPARQL queries. For each domain, precision means the percentage of correctly translated queries in the queries that PANTO produced an output; recall refers to the percentage of queries that PANTO produced an output in the total query set. Since the queries are prepared for evaluating natural language interface to database, some features in the queries are not supported by the current version of SPARQL.
The authors state that the total processing time of query processing was less than one second (0.878) and running time is based on the scale of the ontology and also complexity of the query which is the length and number of the clauses. The authors claim that precision PANTO provides acceptable result for recall and precision in compared with Querix which achieved a precision 86.08 and recall 87.11. The authors experimented on the Mooney data and they claim that queries show that all the correctly parsed natural language queries can be correctly translated into QueryTriples and then be mapped to OntoTriples. They claim that their experiments on three different ontologies have shown that the PANTO approach produces promising results also their novel approach helped bridge the gap between the logic-based semantic web and real-world users.

Also the authors claim PANTO can cover the whole query scope which are supported by AquaLog since Aqualog can parse queries and PANTO can generate them too.

### 2.2.2. An event based denotational semantics for natural language queries to data represented in triplestores

The problem which Frost, Amour & Fortier [2013] addressed in the paper entitled “An event based denotational semantics for natural language queries to data represented in triplestores” is how to allocate prepositional phrases and represent data in the semantic web so that it can be easy to update and access using natural language queries. The authors stated that in the previous works a wide-coverage natural language query interface to triplestores which is based on formal easily-implementable natural-language semantics have not been created.
The authors stated that they present an event-based semantics for natural language which is denotational, each word and phrase has well-defined mathematical meaning, is compositional, each composite expression meaning is the meaning of each component of the expression, is referentially transparent, the meaning of the word or phrase after syntactic disambiguation is same and is Mantogovian correspondence, there is one-to-one correspondence between the syntactic and semantic rule. Frost, Amour & Fortier [2013] noted that they describe the semantics using notation from set theory and recursive function theory.

In the first step, they begin by defining a triplestore called data and in next step retrieval function getts is defined to return the set of triples matching given field value(s).

\[
\text{getts (a, ANY, ANY) = } \{ (x,y,z) \mid (x,y,z) \in \text{data} \& x = a \}
\]

Also the authors introduced some operations which are totally depend on how data is stored and retrieved from triples and they said complex operators definitions are more easier by using the set theory function map. For the semantic part authors state that proper nouns denote function which gets the set of entities as argument and input and return true if the sent input is a member of the set otherwise the output is False. The authors introduced collect function such that when applied to a binary relation, it “collects” values from pairs:

\[
\text{collect rel = } \{(x, \{y \mid (x,y) \in \text{rel}\}) \mid (x, z) \in \text{rel}\}
\]

E.g. collect \{((a,2), (b,3), (a,1), (c,4), (a,7))\}

\[
=>> \{((a, \{2,1,7\})), (b,\{3\}), (c,\{4\})\}
\]

The authors have implemented a prototype of the semantics directly as function definitions in the programming language Miranda but they mentioned that their experiments are reachable by the other programming languages which support high-order functions like Lisp, Haskell, Scheme, MI and python.

Natural language query processor is created by integrating the semantics with a parser using an executable attribute-grammar environment. The authors mention an
example as result for query conversion to semantic expression. The following query is converted to semantic expression: “Which gangster who stole a car in 1899 or 1908 joined a gang which was joined by Torrio?”

which (gangster $that (steal_with_time (a car) (date_1908 $term_or date_1899)))

(join (a (gang $that (joined_by torrio))))

which evaluates to give: {ENT “capone”}.

Frost et al. [2013] claim that their work creates a computer-implementable Montague-like formal semantics of natural-language with an explicit denotation of transitive verbs which can accommodate arbitrary-nested quantification and prepositional phrases.

2.2.3. An Event-Driven Approach for Querying Graph-Structured Data Using Natural Language

The major problem which has been addressed by Frost, Agboola, Matthews, & Donais [2014] is that the most of systems and approaches in natural language systems ignore prepositional phrases since they make the processing so complex. However, prepositions cannot be neglected because in some cases the user requests information regarding specific location or time which is essential to consider the prepositions like “in”, “at”, “through”, etc. The paper by Frost, Agboola, Matthews, & Donais [2014] proposes an approach to locate the prepositional phrases with using a revised version of Montague Semantic. One of the previous approaches which authors appoint to them is by Frost, Amour, and Fortier [2013] which the authors stated that it suffers a defect and shortcoming. The approach proposed by Frost, Amour, and Fortier [2013] is that it can only accommodate queries such as:

“Capon joined the Five Points Gang”

The applied entity-based approaches cannot accommodate queries with prepositional phrase such as “Capon joined the Five Points Gang in 1914”. This problem relates to the entity-based approach. The paper which has been cited by Frost, Agboola,
Matthews, & Donais [2014] introduce their own proposed Event-Driven method as a solution to this problem.

In the paper referred by Frost et al. [2014] introduce event-based method instead of entity-based which can considered as high performance alternative. The authors develop a semantics that is well-known formal semantic of English called Montague Semantic. The authors note that they changed Montague Semantic to create a computationally tractable form called FLMS which is basically suitable basis for natural language query interface to relational database. FLMS is being modified to form EV-FLMS form which is suitable basis for query event-based triplestores. For example the following fact “Capon stoles a car in 1918 in Manhattan” will be represented as below:

{(EV 1004, REL "type", TYPE "steal_ev"),
 (EV 1004, REL "subject", ENT "capone"),
 (EV 1004, REL "object", ENT "car1"),
 (EV 1004, REL "year", ENTRNUM 1918),
 (EV 1004, REL "location", ENT "Manhattan")

Which contains both temporal and spatial phrases and the information related to them can be retrieved easily. The authors noted since the FLMS is based on the sets and relations between sets, they create new functions to retrieve the related data.

The Frost et al. [2014] state for testing their proposed approach, they used the Miranda programming language because:

- It has based on operations and list comprehension which correspond to the “relative set notation”.
- The proposed semantic uses higher-order functions can be defined directly in Miranda.
- Miranda has simpler syntax in compared to other higher-order functional language.
Their proposed idea can be tested because the definitions are executable specifications.

The authors defined a triplestore called data. A sample of data has been represented below:

data =

[(EV 1000, REL "type", TYPE "born_ev"),
(EV 1000, REL "subject", ENT "capone"),
(EV 1000, REL "year", ENTNUM 1899),
(EV 1000, REL "location", ENT "brooklyn"),
(EV 1001, REL "type", TYPE "join_ev"),
(EV 1001, REL "subject", ENT "capone"),
(EV 1001, REL "object", ENT "fpg"),
(EV 1002, REL "type", TYPE "membership"),
(EV 1002, REL "subject", ENT "capone"),
(EV 1002, REL "object", ENT "thief"),
(EV 1002, REL "year", ENTNUM 1908 ),
(EV 1003, REL "type", TYPE "join_ev"),
(EV 1003, REL "subject", ENT "capone"),
(EV 1003, REL "object", ENT "bowery"),
(EV 1004, REL "type", TYPE "steal_ev"),
(EV 1004, REL "subject", ENT "capone"),
(EV 1004, REL "object", ENT "car_1"),
(EV 1004, REL "year", ENTNUM 1918),

91
(EV 1004, REL "location", ENT "manhattan"),
(EV 1005, REL "type", TYPE "smoke_ev"),
(EV 1005, REL "subject", ENT "capone"),
(EV 1006, REL "type", TYPE "membership")]

They defined some basic retrieval functions returns triples from data which match given
text
field value(s):

getts (a,ANY,ANY) = [(x,y,z) | (x,y,z) < data; x = a]
getts (ANY,ANY,c) = [(x,y,z) | (x,y,z) < data; z = c]

etc.

There are some operations in their approach also

first (a,b,c) = a
second (a,b,c) = b
third (a,b,c) = c
thirdwithfirst (a,b,c) = (c, a) etc.

The authors stated that some more complex operations can be defined based on the simple
ones like:

get_subj_for_event ev = thirds (getts (ev, REL "subject", ANY))
get_subjs_for_events evs = concat (map get_subj_for_event evs)
They introduce one of the useful operation which can help to return all the subject of an event of a given type:

\[
\text{get_subjs_of_event_type event_type} = \text{get_subjs_for_events events}
\]

where

\[
\text{events} = \text{firsts (getts (ANY, REL "type", TYPE event_type))}
\]

For example: \(\text{get_subjs_of_event_type "smoke"} \Rightarrow \{\text{ENT "capone"}\}\)

The authors stated that the denotation of a noun is the set of entities based on FLMS which are members of the set associated with that noun. The get_members function returns that set as a list.

An example use of this operator is:

\[
\text{get_members "person"} \Rightarrow \{\text{ENT "capone"}, \text{ENT "torrio"}\}
\]

They defined denotations as functions with an appropriate name, e.g. person. These denotations can then be applied to each other in the program, as shown on the next page, to create the meanings of more complex phrases.

\[
\text{person} = \text{get_members "person"}
\]

\[
\text{gang} = \text{get_members "gang"}
\]

\[
\text{car} = \text{get_members "car"}
\]

\[
\text{thief} = \text{get_members "thief"}
\]

\[
e.g. \text{gang} \Rightarrow \{\text{ENT "fpg", ENT "bowery"}\}
\]
In order to retrieve the transitive verbs which are more complex, something similar to the image in the FLMS approach is proposed. They created “images” for an event using the following:

\[
\text{make_image et = collect (concat [(thirds.gets . get) (ev, REL "subject", ANY) | ev <- events]) where events = (firsts . get) (ANY, REL "type", TYPE et)}
\]

An example application:

\[
\text{make_image "join_ev" => [(ENT "capone", [EV 1001, EV 1003]), (ENT "torrio", [EV 1009])]}
\]

Prepositional phrases can be accommodated by having the parser convert the list of prepositional phrases to a possibly empty list of “prepositional pairs”. Each pair consists of a REL value and a termphrase. For example, the phrase “in 1908 or 1918, in Manhattan” which consists of two prepositional phrases is converted to:

\[
[(\text{REL "year", year_1908 $termor year_1918}), (\text{REL "location", "manhattan"})]
\]

The definition of each transitive verb is redefined to make use of this list to filter the events which are in the image of the event-type associated with that transitive verb before the termphrase which is the argument to the denotation of the transitive verb is applied to the set of objects associated with the event. A recursive function called filter_ev applies each prepositional phrase in turn as a filter to each event:

\[
\text{steal’ tmph preps = [ subj | (subj, evs) <- image_steal; tmph (concat [(thirds.gets) (ev, REL "object", ANY) | ev <- evs; filter_ev ev preps])}
\]

94
filter_ev event [] = True

filter_ev event (prep:list_of_preps) = ((snd (prep)) ((thirds.getts)(event,fst (prep),ANY)))

& filter_ev event list_of_preps

For example:

steal' (a car) [(REL "year", year_1908 $termor year_1918), (REL "location", "manhattan")]

=> [ENT "capone"]

In paper cited by Frost, Agboola, Matthews & Donais [2014] claim that their proposed approached which is based on events, has the following six properties:

- The semantics is denotational in the sense that English words and phrases have well-defined mathematical meaning.
- The meaning of a composite phrase can be created by applying simple operations to the meanings of its components
- It is referentially transparent in the sense that the meaning of a word or phrase (after syntactic disambiguation) is the same no matter in what context it appears.
- There is a one-to-one correspondence between the semantic rules describing how the meaning of a phrase is computed from its components and the syntactic rules describing the structure of the phrase.
- It is computationally tractable.
- The meanings of words are defined directly in terms of primitive triplestore retrieval operations.
These properties enable NL triplestore query processor to be implemented as a highly modular syntax-directed interpreters. Consequently, the query processors can easily be extended to accommodate new language constructs such as prepositional phrases.

### 2.2.4. Summary

<table>
<thead>
<tr>
<th>Year</th>
<th>Author</th>
<th>Title of Paper</th>
<th>Major Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>Wang, Xiong, Zhou, &amp; Yu</td>
<td>PANTO: A Portable Natural Language Interface to Ontologies</td>
<td>A novel approach which is called PANTO that has been implemented as stand-alone web application, it produces one parse tree for each input. PANTO approach produces promising results also their novel approach helped bridge the gap between the logic-based semantic web and real-world users.</td>
</tr>
<tr>
<td>2013</td>
<td>Frost, Amour and Fortier</td>
<td>An event based denotational semantics for natural language queries to data represented in triplestores</td>
<td>Presenting a new event-based denotational semantics which can be used as the basis for natural language (NL) query interfaces to triplestores.</td>
</tr>
<tr>
<td>2014</td>
<td>Frost, Agboola, Matthews and Donais</td>
<td>An Event-Driven Approach for Querying Graph Structure Data Using Natural Language</td>
<td>Entity-based triples cannot be used for development of semantic theories of complex prepositional phrases. They propose an alternative approach, which uses “event-based” triplestores, treats English queries as expressions of the lambda calculus which can be</td>
</tr>
</tbody>
</table>
evaluated directly with respect to the triplestore.

2.2.5. Cited by

<table>
<thead>
<tr>
<th>Author</th>
<th>Cited By</th>
</tr>
</thead>
</table>

3. Conclusion:

Fifteen papers were identified which studied prepositional phrases in the natural language query to semantic web and they are exactly on the topic of survey and 12 of them were chosen as most important papers. I reviewed those 12 papers in detail which interestingly in spite of importance of the natural language to semantic web nowadays, not many researches have worked on prepositional phrases because of complexity and difficulty. The most successful and important approaches deal with mentioned issue were proposed by Dr. Frost who firstly improved Montague semantic which is one of the robust approach in semantic area and proposed new idea of the EV-FLMS secondly which can accommodate and locate the prepositions successfully in natural language query to semantic web. The revised version of the Montague Semantic is novel and can help processors to improve their performance hopefully.

4. ACKNOWLEDGEMENT

I have honor to acknowledge the guidance and knowledge of Dr. Frost in this survey who help me to find deeply understanding of my survey topic and finding the more related papers
Appendix i - Bibliography


Appendix III – Annotated Bibliography

5.1. Analyti et al. 2012


Problem:
In this paper, the authors provide a survey on the models and query languages for temporally annotated RDF.
The author mention that in most of the previous works, a temporally annotated RDF ontology is essentially a set of RDF triples associated with temporal constraints, where, in the simplest case, a temporal constraint is a valid temporal interval. Also the authors address that problem that some of the works provide experimental results while the rest are purely theoretical.

Previous Work:


• C. Gutierrez, C. A. Hurtado, and A. A. Vaisman, "Introducing Time into RDF", IEEE Transactions on Knowledge and Data Engineering, 19(2), 2007, 207-218.


Shortcoming:
The researcher mentioned that one of previous work that has their own model theory is a tRDF query over a tRDF database $D$ is a set of triples of the form $(s, p:[T], o), (s, p:<n:T>)$,
o), (s, p:[n:T], o), where s, p, o, T are possibly variables, with the constraint that each temporal variable appears only once.

The works that extend RDFS entailment seems less efficient since it computes the RDFS closure of RDF triples at each time point.

**New Idea:**

This paper is a survey and compares different approaches which the new idea of each of them is as follow:

E. The paper referred by (Pugliese, Udrea & Subrahmanian, 2008) proposes an approach which a *temporal RDF* (*tRDF* for short) *database* is a set of triples of the form (s, p:{T}, o), (s, p:<n:T>, o), (s, p:[n:T], o). And *tRDF* query over a *tRDF* database D is a set of triples of the form (s, p:{T}, o), (s, p:<n:T>, o), (s, p:[n:T], o), where s, p, o, T are possibly variables. The authors of survey mentioned that that paper presented an efficient algorithms for simple and conjunctive query answering, showing that the time complexity for answering a conjunctive query is in \(O((|R|2^{|P|}|Q|))\), where |Q| is the number of simple queries in Q.

F. In papers referred by (Tappolet & Bernstein 2009), the authors expressed instead of having RDF triples graphs are used both for saving space and for querying the temporal RDF database using standard SPARQL. The authors introduce through examples a query language, named \(\tau\)-SPARQL which extends the SPARQL query language for RDF graphs. Each \(\tau\)-SPARQL query can be translated into a SPARQL query.

A \(\tau\)-SPARQL query that retrieves all foaf:Persons whose lifespan overlaps with Einstein’s is:

```
SELECT ?s2, ?e2 ?person WHERE {

[?s1, ?e1] ?einstein foaf:name “Albert Einstein”
```
This query is translated into a SPARQL query, as follows:

```
SELECT ?s2, ?e2 ?person WHERE{
  GRAPH ?g1 {?einstein foaf:name "Albert Einstein".}
  ?g2 time:intervalOverlaps ?g1.
  GRAPH ?g2 {?person a foaf:Person.}
  ?g2 time:hasBeginning ?s2.
  ?g2 time:hasEnd ?e2.}
```

The authors proposed an index structure for time intervals, called *keyTree index*, assuming that triples within named graphs have indices by themselves. The proposed index improves the performance of time point queries over an in-memory ordered list that contains the intervals’ start and end times.

In paper referred by (Rodriguez, McGrath, Liu, & Myers, 2009) introduced the time-annotated RDF framework which is proposed for the representation and management of time-series streaming data. In particular, a TA-RDF graph is a set of triples <s[tS],p[tp], o[to]>, where <s,p,o> is an RDF triple and tS, tp, and to are time points. In other words, a TA-RDF graph relates streams at certain points in time. To translate a TA-RDF graph into a regular RDF graph, a data stream vocabulary is used, where dvs:belongsTo is a property that indicates that are source is a frame in a stream, dvs:hasTimestamp is a property indicating the timestamp of a frame, and dvs:Nil is a resource corresponding to the Nil timestamp.

An RDF graph G is the translation of a TA-RDF graph GTA iff (B is the set of blank nodes):

```
<s[tS], p[tp], o[to]> □ □ GTA □ □ rS, rp, ro
```
A query language for the time-annotated RDF, called TASPARQL, is proposed which has a formal translation into normal SPARQL. The proposed system has been implemented on top of the Tupelo semantic middleware.

G. In paper referred by (McBride and Butler, 2009), the authors considered an extension of RDFS with spatial and temporal information. In that survey, only the extension with temporal information has been considered. A set $D$ of RDF triples associated with their validity temporal interval $i$. Starting from $D$, The inference rules has been applied $A : i, B : i' \rightarrow C : i \cap i'$, where $A, B \rightarrow C$ is an RDFS entailment rule and $i, i'$ are temporal interval variables, until a fixpoint is reached. Then, the temporal intervals of the same RDF triple are combined, creating maximal temporal intervals.

Based on these maximal temporal intervals, a formal extension of the SPARQL language is proposed, called SPARQL-ST, supporting however only the AND and FILTER operations. The TEMPORAL FILTER condition is precisely defined supporting all interesting conditions between temporal intervals including Allen’s temporal interval relations.
H. The other novel approach is a general framework for representing, reasoning, and querying annotated RDFS data is presented. The authors show how their unified reasoning framework can be instantiated for the temporal, fuzzy, and provenance domain.

Experiment:

Not more experiments were provided for studied paper and authors of survey mentioned that no experimental results are provided.

Result and claim:

This paper has reviewed models and query languages of temporally annotated RDF. The authors believe that approaches that have their own model theory. The researchers claim that the paper referred by (McBride & Butler, 2009) entitled “Representing and Querying Validity Time in RDF and OWL: A Logic-Based Approach” achieve query answering using directly maximal temporal intervals achieving a higher performance. However it is not able to return maximal intervals within a temporal interval of interest. Furthermore, they claim that the paper referred by (Zimmermann, Lopes, Pollere & Straccia,, 2012) which its title is “A General Framework for Representing, Reasoning and Querying with Annotated Semantic Web Data” cannot always provide appropriate answer for entered query.

5.2. Bernstein et al. 2004

Problem:

The semantic web demonstrates the sharply growing knowledge base which should allow users to query over this huge amount of data with formal logic and query which can be also easy for users leaning of that query language. There is huge gap between the knowledge of the users and available query languages so far so because of that reason it is being difficult for them to query over them. The current paper tried to addressed this problem and bridge the gap between the logic-based semantic web and real-world users.

Previous Work:


Shortcoming:

Although the authors have not noted any shortcoming of the previous works, they stated that they could not find any other application of controlled natural language querying of semantic web content. Furthermore, they stated that work on natural language interfaces to data bases has largely tapered off since the 80’s.

New Idea:

The authors proposed their new approach by introducing ACE which is small subset of the English meaning which each of the ACE sentence is correct, also ACE contains English grammar which is contains set of constructions and interpretations rule. For being simple
semantics and syntactical complexity of the English is removed from the ACE. APE, Attempto Parsing Engine, which has been implemented in Prolog as a Define Clause Grammar translate the ACE to DRS- discourse representation structure that logically represent the text. A DRS consists of discourse referents such as quantified variables which represents the objects of a discourse, and of conditions for the discourse referents. The conditions can be logical atoms or complex conditions built from other DRSs and logical connectors like negation, disjunction, and implication.

Afterward the DRS is translated to semantic web query language PQL which can be used for query an ontology. PQL’s two major statement types are ATTRIBUTE and RELATION.

**Experiment:**

For validation of the proposed approach they combined Prolog and Java components, as APE and the rewriting framework are programmed in SICStus Prolog, and the user interface and the query engine are also programmed in Java. They executed number of real-world queries and then compared its retrieval performance with two keyword-based retrieval approaches, both of those approaches have proven that the output is suitable for end-user.

**Claim:**

The authors claim that their proposed approach can generate appropriate and expected PQL.

5.3. Cimiano et al. 2009

Problem:

The authors addressed the problem of how to accommodate different constitutive part of entered queries efficiently and improve the precision and accuracy of the NLIs performance in answering the questions. One of the discussed part of speech queries which have been attempted to locate it is prepositional phrases which have been ignored in so many previous work of the natural language area research.

Previous Work:


P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer

Shortcoming:

The authors mentioned that the previous works have not paid attention in particular quantitative analysis problems inherent in the task building natural language interfaces. They stated that although there have been qualitative analysis of the problems involved in constructing NLIs, there has been no quantitative analysis grounding the qualitative characteristics of the problem in real data. The mentioned area can guide the system developers in the future also it can help them to focus on specific phenomenon encountered
in NLIs and progress easily in the field by clearly designing and evaluating the solution to a specific phenomenon.

**New Idea:**

The authors noted to provide a quantitative analysis of the problem of constructing an NLI, they use the Geobase database which are being used frequently for natural language interfaces. The Geobase dataset describes states, cities, mountains, lakes, rivers and roads in the U.S., together with attributes such as area (state, lake), population (state, city), length (river), and height (mountain, location) etc. The Geobase has been converted to ontology language F-Language and OWL. When converting into OWL and F-Logic has been completed, 7 concepts with a total 17 various relation have been used which the concepts used with their relations are given below:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>name, abbreviation, capital, density, population, area, code hasCity, border, highest point, lowest point</td>
</tr>
<tr>
<td>city</td>
<td>name, area, inState</td>
</tr>
<tr>
<td>river</td>
<td>name, length, owsThrough</td>
</tr>
<tr>
<td>mountain</td>
<td>name, inState, height</td>
</tr>
<tr>
<td>road</td>
<td>number, passesThrough</td>
</tr>
<tr>
<td>lake</td>
<td>name, area, inState</td>
</tr>
<tr>
<td>location</td>
<td>name, inState, height</td>
</tr>
</tbody>
</table>

Some of the information into one class merged into one class, some redundancies have been removed, and the location class has been added which includes a height attribute for the location in question.

The authors noted that they manually created F-Logic queries yielding the appropriate answers as result when evaluated with the OntoBroker system, but also queries in generic logical form.
The most important point of using F-Logic language in the OntoBroker system is that built-in functionality is provided for numerical comparison as well as aggregation operators for calculating minimum, maximum, sums, etc. They attempted to annotate each of the questions together with characteristics that they regarded as relevant to our quantitative analysis.

**Experiment:**

The language which is used in the questions is rather simple, and contains a lot of `light syntactic constructions' such as:

Light preposition `of’ (appearing in 21.52% of the sentences): What are the highest points of all the states? (210) or what are the major cities of texas? (235)

Light preposition `with' (appearing in 7.36% of the sentences): How many people live in the state with the largest population density? (104), or what are the cities of the state with the highest point? (209)

This can be conducted that relevant relations are not expressed obviously and are hidden implicitly behind the light constructions like prepositions “with” and “of”. Because of that so many shallow approaches ignore to consider linguistic details like prepositions phrases.

Furthermore, they recognize that there are so many syntactic ambiguities which in case of prepositional phrases they also distinguish the case of prepositional phrase providing essentially the predict in copula construct.

The experiments show that by using PPs to the last constituent, correct decision in 99.27% of the cases for PP attachment including last and only attachment point cases as well as the copula case where the PP functions as predicate is predictable.

In addition to light and vague prepositions, there are some spatial prepositions like “in”, “next”, “though”. Although some systems have shown that their performances are acceptable by essentially ignoring prepositions, the more principled and successful
solutions would capture the domain independent meaning of such spatial prepositions, allowing to reuse their meaning across domains.

In the context of an ontology about american presidents, it makes a difference whether we ask for `Who was president after WWII', vs. `Who was president during WWII'. In general, systems would thus profit from capturing the meaning of prepositions explicitly, possibly even assigning them a domain independent meaning.

Claim:

The authors mentioned they have started work in this area since a data-driven analysis has been missing so far while there have been many qualitative descriptions of the problems.

Basically the authors believe that is not acceptable to neglect the prepositional phrases and some other details in natural language processing, because it can cause some unpredicted problems and require adhoc and principled extensions. The authors believe that constructing systems with deep syntactic and semantic processing capabilities will pay off in the long term.

5.4. Dylla et al. 2011

**Problem:**

The authors stated the knowledge base is inconsistent generally and not sufficient constraints are applied to knowledge base. Expectedly so many contradicts can be existed in knowledge base, to resolve these inconsistencies some form of consistency should be imported to knowledge base.

Temporal annotation can be one of the most useful and efficient constraint which can not only express general constraints among facts but also add a finer granularity to the consistency reasoning itself. Furthermore, they stated that even when using simple time intervals for the representation of temporal annotations with such disjointness and precedence constraints, the satisfiability problem is known to be NP-hard.

**Previous Work:**


Shortcoming:

The authors state that the rules which have been considered in previous work do not consider the inclusion of actual consistency constraints, where only some facts out of a given set may be set to true while other facts are considered false. The other shortcoming of previous work which author addressed is that they only consider positive lineage for example conjunctions and disjunctions only.

New Idea:

The authors introduce their new idea by defining a knowledge base $KB = (F; C)$ as a pair consisting of a set of facts $F$ which each of them has weighted and temporal and a set of first-order consistency constraints $C$. To encode facts, they employed the used RDF, in which facts $F \in \text{Rel} \times \text{Entities} \times \text{Entities}$ are stored as triples consisting of a relation and a pair of entities. Furthermore, the original RDF triplet have been extended structure in two ways: first, and second, to include time information into knowledge base, also assign a time interval of the form $[tb; te)$ to each fact $f$.

They also provide some format of consistency constraint:

$$relE1(e1; e2; t1) \land relE2(e1; e3; t2) \land relA(e2; e3) \rightarrow relT(t1; t2)$$

The authors mentioned that based on the type of constraint, the combinational complexity of resolving conflict is different and selecting which constraints should be formulated will be so crucial.

There are three different type of constraint which includes:
• Temporal disjoint
• Temporal precedence
• Mutual exclusion

They utilize the following template to express disjointness constraints.

$$relE(e1; e2; t1) \land relE(e1; e3; t2) \land e2 \neq e3 \implies \text{disjoint}(t1; t2)$$

They employ following template for precedence constraints shows the restriction of the time interval of an instance of relE1 ends before the interval of a fact with relE2 starts

$$relE1(e1; e2; t1) \land relE2(e1; e3; t2) \implies \text{before}(t1; t2)$$

They use following temporal is for mutual exclusion. Mutual exclusion defines a set of facts which are all in conflict with each other, regardless of time.

$$relE(e1; e2; t1) \land relE(e1; e3; t2) \land e2 \neq e3 \implies \text{false}$$

**Experiment:**
The authors mentioned that their work has been implemented by Java 1.6 in about 3k lines of code. Additionally, they employed a greedy heuristic for the MWIS, which proved to perform best on the data among all the greedy methods which have been tried before. There are other means of approximating the MWIS problem, like stochastic optimization. However they are even less scalable than greedy methods.

The dataset which they used contains data about the playsForClub, playsForNational, and hasWonPrize relations, which has been extended manually by dates of birth and death.
Result:
Their algorithm showed impressive robustness. Moreover, the histogram of scheduling algorithm exhibits excellent behavior as in nearly every problem instance the optimal solution was found.

Claim:
The authors claim their approach works by identifying a subclass of first-order consistency constraints, which can be efficiently mapped to constraint graphs and be solved using results from scheduling theory. They claim that their experiments show that applied approach performs superior to common heuristics that directly operate over the underlying Maximum Weight Independent Set problem in terms of both run-time and quality.

5.5. Frost et al. 2014

Problems:
Much research has attempted to query the semantic web via natural language. Most of them ignore prepositional phrases since they make the processing so complex. They cannot be neglected because in some cases the user requests information regarding specific location or time which need to consider the prepositions like “in”, “at”, “through”, etc. The paper by (Frost, Agboola, Matthews, & Donais, 2014) proposes an approach to locate the prepositional phrases with using a revised version of Montague Semantic.

Previous Work:
Shortcoming:

The previous work can only accommodate queries such as:

“Capon joined the Five Points Gang”

The applied entity-based approaches cannot accommodate queries with prepositional phrase such as “Capon joined the Five Points Gang in 1914”. This problem relates to the entity-based approach. The authors introduce their own proposed method as a solution to this problem.

New Idea:

The authors propose a new approach which is event-based instead of being entity-based. They develop a semantics that is well-known formal semantic of English called Montague Semantic. The authors note that they changed Montague Semantic to create a computationally tractable form called FLMS which is basically suitable basis for natural language query interface to relational database. FLMS which has been proposed by the authors in their previous work is being modified to form EV-FLMS form which is suitable basis for query event-based triplestores. For example the following fact “Capon stoles a car in 1918 in Manhattan” will be represented as below:

{(EV 1004, REL "type", TYPE "steal_ev"),
 (EV 1004, REL "subject", ENT "capone"),
 (EV 1004, REL "object", ENT "car1"),
 (EV 1004, REL "year", ENTRNUM 1918),
Which contains both temporal and spatial phrases and the information related to them can be retrieved easily. The authors noted since the FLMS is based on the sets and relations between sets, they create new functions to retrieve the related data.

**Experiment:**

The author’s state for testing their proposed approach, they used the Miranda programming language because:

- It has based on operations and list comprehension which correspond to the “relative set notation”.
- The proposed semantic uses higher-order functions can be defined directly in Miranda.
- Miranda has simpler syntax in compared to other higher-order functional language.
- Their proposed idea can be tested because the definitions are executable specifications.

They defined a triplestore called data. A sample of data has been represented below:

```plaintext
data =
[(EV 1000, REL "type", TYPE "born_ev"),
 (EV 1000, REL "subject", ENT "capone"),
 (EV 1000, REL "year", ENNUM 1899),
 (EV 1000, REL "location", ENT "brooklyn"),
 (EV 1001, REL "type", TYPE "join_ev"),
 (EV 1004, REL "location", ENT "Manhattan")]
```
(EV 1001, REL "subject", ENT "capone"),
(EV 1001, REL "object", ENT "fpg"),
(EV 1002, REL "type", TYPE "membership"),
(EV 1002, REL "subject", ENT "capone"),
(EV 1002, REL "object", ENT "thief"),
(EV 1002, REL "year", ENTNUM 1908 ),
(EV 1003, REL "type", TYPE "join_ev"),
(EV 1003, REL "subject", ENT "capone"),
(EV 1003, REL "object", ENT "bowery"),
(EV 1004, REL "type", TYPE "steal_ev"),
(EV 1004, REL "subject", ENT "capone"),
(EV 1004, REL "object", ENT "car_1"),
(EV 1004, REL "year", ENTNUM 1918),
(EV 1004, REL "location", ENT "manhattan"),
(EV 1005, REL "type", TYPE "smoke_ev"),
(EV 1005, REL "subject", ENT "capone"),
(EV 1006, REL "type", TYPE "membership")]

They defined some basic retrieval functions returns triples from data which match given field value(s):

getts (a,ANY,ANY) = [(x,y,z) | (x,y,z) <- data; x = a]
getts (ANY,ANY,c) = [(x,y,z) | (x,y,z) <- data; z = c]

etc.
There are some operations in their approach also

first (a,b,c) = a
second (a,b,c) = b
third (a,b,c) = c
thirdwithfirst (a,b,c) = (c, a) etc.

The authors stated that some more complex operations can be defined based on the simple ones like:

get_subj_for_event ev = thirds (getts (ev, REL "subject", ANY))
get_subjs_for_events evs = concat (map get_subj_for_event evs)

They introduce one of the useful operation which can help to return all the subject of an event of a given type:

get_subjs_of_event_type event_type = get_subjs_for_events events where
events = firsts (getts (ANY, REL "type", TYPE event_type))

For example:

get_subjs_of_event_type "smoke" => [ENT "capone"]

The authors stated that the denotation of a noun is the set of entities based on FLMS which are members of the set associated with that noun. The get_members function returns that set as a list.

An example use of this operator is:
They defined denotations as functions with an appropriate name, e.g. person. These denotations can then be applied to each other in the program, as shown on the next page, to create the meanings of more complex phrases.

```
get_members "person" => [ENT "capone", ENT "torrio"]
```

For retrieving the transitive verbs which are more complex, something similar to the image in the FLMS approach is proposed. They created “images” for an event using the following:

```
make_image et = collect (concat [(thirdswithfirsts . getts) (ev, REL "subject", ANY)| ev <- events]) where events = (firsts . getts) (ANY, REL "type", TYPE et)
```

An example application:

```
make_image "join_ev" => [(ENT "capone", [EV 1001, EV 1003]),
(ENT "torrio", [EV 1009])]
```

Prepositional phrases can be accommodated by having the parser convert the list of prepositional phrases to a possibly empty list of “prepositional pairs”. Each pair consists of a REL value and a termphrase. For example, the phrase “in 1908 or 1918, in Manhattan” which consists of two prepositional phrases is converted to:
(REL "year", year_1908 $termor year_1918), (REL "location", "manhattan")]

The definition of each transitive verb is redefined to make use of this list to filter the events which are in the image of the event-type associated with that transitive verb before the termphrase which is the argument to the denotation of the transitive verb is applied to the set of objects associated with the event. A recursive function called filter_ev applies each prepositional phrase in turn as a filter to each event:

```
steal’ tmph preps = [ subj | (subj, evs) <- image_steal;
  tmph (concat [(thirds.getts) (ev, REL "object", ANY) | ev <- evs;
    filter_ev ev preps])]
filter_ev event [] = True
filter_ev event (prep:list_of_preps) = ((snd (prep)) ((thirds.getts)
  (event,fst (prep),ANY)))
& filter_ev event list_of_preps
```

For example:

```
steal’ (a car) [(REL "year", year_1908 $termor year_1918), (REL "location", "manhattan")]
=> [ENT "capone"]
```

Claim and Result:
The authors claim that their proposed approached, which is based on events, has the following six properties:

- The semantics is denotational in the sense that English words and phrases have well-defined mathematical meaning.
- The meaning of a composite phrase can be created by applying simple operations to the meanings of its components.
- It is referentially transparent in the sense that the meaning of a word or phrase (after syntactic disambiguation) is the same no matter in what context it appears.
- There is a one-to-one correspondence between the semantic rules describing how the meaning of a phrase is computed from its components and the syntactic rules describing the structure of the phrase.
- It is computationally tractable.
- The meanings of words are defined directly in terms of primitive triplestore retrieval operations.

These properties enable NL triplestore query processor to be implemented as a highly modular syntax-directed interpreters. Consequently, the query processors can easily be extended to accommodate new language constructs such as prepositional phrases.

5.6. Frost et al. 2013


Problem:
The problem is how to allocate prepositional phrases and represent data in the semantic web so that it can be easy to update and access using natural language queries.

**Previous Work:**


**Shortcoming of previous work:**

In the previous works a wide-coverage natural language query interface to triplestores which is based on formal easily-implementable natural-language semantics have not been created yet.

**New Idea:**
The authors stated that they present an event-based semantics for natural language which is denotational, each word and phrase has well-defined mathematical meaning, is compositional, each composite expression meaning is the meaning of each component of the expression, is referentially transparent, the meaning of the word or phrase after syntactic disambiguation is same and is Mantogovian correspondence, there is one-to-one correspondence between the syntactic and semantic rule.

The authors mentioned that they describe the semantics using notation from set theory and recursive function theory.

In the first step, they begin by defining a triplestore called data and in next step retrieval function getts is defined to return the set of triples matching given field value(s).

getts (a, ANY, ANY) = { (x,y,z) | ( x,y,z) ∈ data & x = a}

Also they introduced some operations which are totally depend on how data is stored and retrieved from triples and they said complex operators definitions are more easier by using the set theory function map.

For the semantic part authors state that proper nouns denote function which gets the set of entities as argument and input and return true if the sent input is a member of the set otherwise the output is False.

**Experiment:**

The authors have implemented a prototype of the semantics directly as function definitions in the programming language Miranda but they mentioned that their experiments are reachable by the other programming languages which support high-order functions like Lisp, Haskell, Scheme, MI and python.
Result:

Natural language query processor is created by integrating the semantics with a parser using an executable attribute-grammar environment. The authors mention an example as result for query conversion to semantic expression. The following query is converted to semantic expression: “Which gangster who stole a car in 1899 or 1908 joined a gang which was joined by Torrio?”

\[
\text{which (gangster } \text{that (steal} \text{with time (a car) (date} \text{1908 } \text{term or date} \text{1899)))}
\]

\[
\text{(join (a (gang } \text{that (joined} \text{by torrio)))})
\]

which evaluates to give: \{ENT “capone”\}.

Claim:

The authors claim that their work creates a computer-implementable Montague-like formal semantics of natural-language with an explicit denotation of transitive verbs which can accommodate arbitrary-nested quantification and prepositional phrases.

Citation:

5.7. Kolas et al. 2007


**Problem:**

The authors state that RDF and modern triplestores are efficient at storing and querying data linked across multiple sources of information but they are not that much efficient it comes to spatial processing. The standard for storing spatial data involves object-oriented database with spatial features. But one of the drawbacks of the object-relational model is that it does not have flexibility of RDF and triplestores that make them attractive for searching linked data across multiple sources.

**Previous Work:**


**Shortcoming:**

The authors stated that mapping queries that include spatial instances and relationships to SPARQL is not easy. There are many possible ways that one could use SPARQL for spatial data, and the ideal way has yet to be found.

**New Idea:**
The authors propose SPAUK whose goal is to provide efficient spatial processing for spatial semantic systems. They chose SPAUK as a semantic knowledge base capable of supporting supplementary spatial indices.

Additionally, designing a system such that the addition of spatial processing to the system can be as transparent as possible to the user is the secondary goal of designing the SPAUK.

All data including semantic data and spatial data, is still presented as a graph. The authors mention that the knowledge base presents itself as a standard SPARQL endpoint. This allows any clients capable of interfacing via the SPARQL protocol to utilize SPAUK. The design should present one conceptual graph to its clients, and then queries can be over this graph by dividing appropriately into sub-queries which can be answered by the various parts of the knowledgebase. Spatial parts which include locations and spatial relationship

Spatial parts of the query, including locations and spatial relationship should be sent to the spatial index and query processors. The nonspatial part of the query must be sent to the related triplestore. Results must be combined from the two parts to form a related answer. Furthermore, data which is inserted must find its way into the appropriate parts of the knowledge base.

The architecture of SPAUK is based on the Jena Semantic Web Framework and Joseki.

Claim:

Although the formal analysis of the performance of using a supplemental spatial index in object-relation has not been done yet and only simple analysis of the algorithms involved, the authors claim that preliminary usage of the SPAUK system have shown that the approach is valid.

Conclusion:
At the end of the paper, the authors mention that by Attaching a semantic GIS client to the SPAUK system provides responsive spatial semantic query capability. They believe that this type of system enables a new class of semantic applications whose full potential cannot yet be conceived.

5.8. Schwitter et al. 2004


**The problem:**

The number of querying by natural language is increasing sharply on the semantic web. However, there are some suggested format to retrieve the documents and resources on the web, users have to be familiar with complicated structure and format of the storing and retrieving interfaces. The current paper attempts to provide a solution which retrieve the related information regarding submitted query by user by presenting the PENG-D, a proposal for a controlled natural language.

**Previous work:**


Shortcoming of previous work:

RDF schema is being used to preserve and address the web resources which suffers of limitation in knowledge representation with few modeling primitives and more expressive power for defining web resources. Furthermore, there is no any unique standard layer meta-modeling architecture which be able to assign two or more roles to elements in RDF specification so this makes it extremely difficult to layer more expressive ontology and rule languages on top of RDFS.

New Idea:

The authors mentioned that they are addressing new approach which is called PENG-D. Basically PENG is a machine-oriented controlled natural language that has been developed to write specifications for knowledge representation. While PENG was designed for writing specifications that are first-order equivalent, the proposed language PENG-D has formal properties that are equivalent to DLP. PENG-D provides a clear computational pathway to layer more expressive constructions on top of it.

The planned architecture of the PENG-D system looks similar to the PENG system but offers support for ontology construction. The PENGD system consists of four main components: a look-ahead text editor, a controlled language (CL) processor, an ontology component and an inference engine.

Experiment:

Since we are considering the way for prepositional phrase in natural language in our survey so suggested approach for preposition phrase which has been mentioned by the authors in current paper is provided as follow.
The authors state that finding the prepositional phrase is like the complement statement which has been done by most of the syntactic structures that are approved for the subject position. It allows for the prepositional construction has ... as ... and for coordinated structures:

Nic is married to Sue.

Nic has Rex as dog.

Nic has Rex as dog and Tweety as bird.

Result:

The authors have not provided exact experiment and the result of that but they pointed that their proposed approach, PENG-D, which is a machine-oriented controlled natural language that has same expressivity as DLP. DLP offers a promising first-order based alternative that enables ontological definitions to be combined with rules.

In this paper we referred to a number of deficiencies of RDFS as a “knowledge representation” language for the envisioned Semantic Web. Layering more complex ontology and rule languages on top of RDFS is not straightforward, because of its non-standard and non-fixed layer meta-modelling architecture. The relatively new DLP paradigm offers a promising first-order based alternative that enables ontological definitions to be combined with rules. To make such machine-processable information easily accessible for non-specialists, we proposed the use of PENG-D, a machine-oriented controlled natural language that has the same expressivity as DLP.

Claim:

The authors claim that they referred to number of deficiencies of RDFS as a “knowledge representation” language for the envisioned Semantic Web. Layering more complex ontology and rule languages on top of RDFS is not straightforward, because of its non-
standard and non-fixed layer meta-modelling architecture. Furthermore they claim that PENG-D is easy to write for non-specialists with the help of a look-ahead text editor, easy to read in contrast to RDF-based notations, and easy to translate into a corresponding machine processable format. In brief: PENG-D has the potential for complementing these more machine-oriented notations.

Citation:


M. Muhlhauser. Smart products: An introduction. In Constructing Ambient Intelligence, pages 158 -164. 2008

5.9. Stoermer et al. 2006


Problem:

The author address that Knowledge Representation point of view RDF statements in general are context-free, and thus represent statements of universal truth, while documents contain context sensitive information. One small example can be two contradicted
statement such as “Silvio Berlusconi is the Prime minister of Italy” and “Romano Prodi is the Prime minister of Italy”. This is very small contradiction which can be handled by some defined approach but in Semantic Web terms and in large number of uncoordinated information systems, it could be a serious problem.

The authors believe that such contradictions, contradictory beliefs and facts which become semantically incorrect in the absence of additional pragmatic or contextual information are likely to impose serious problems on the coordination and interoperation of information systems in the Semantic Web.

**Shortcoming:**

The previous work has not used reification but implements context as a real extension of the RDF model theory, by moving from triples to quadruples for identifying the context to which a statement belongs. These previous ideas have not been pursued any further. Moreover all the currently available RDF tools would have to be extended in order to deal with such an RDF model.

**New Idea:**

The authors state that the notion of context into RDF to attack the previously mentioned problem and to limit the scope of an RDF statement to the context in which it is relevant. The new idea of their paper is based on the logical theory of Multi Context Systems and the principles of Locality and Compatibility.

Basically, the proposed approach states that contexts can be seen in a peer-to-peer view, resembling more general aspects such as human beliefs, agent knowledge and other distributed systems.

All the statement which belong to a context should be presented in separate named RDF graph and the graph can be extended in a way which contexts can be appeared as standard object in RDF statement.
Then the author tries to connect the two contexts to allow for reasoning across contexts. This aspect is seems to important and crucial since sensible queries can be issued and all relevant information is taken into account. So many approaches have been thought for modeling these relations. One of them is the vocabulary of the context can be provided by implementer to describe relations between contexts. The compatibility relations are supposed to be modeled in following ways: 1. provide some limited relations and require all other systems which implement this system to take care of the defined relation. 2. Provide an ontology for context relations, so that there exists a vocabulary to describe these relations with the help of RDF. This approach is slightly more flexible, because the ontology is extendable. 3. Define a CR to be implemented as a semantic attachment, which can be thought of as a sort of plugin to the system

The important aspect is that reasoning within a context follows standards mechanisms, as the non-elementary view on the axioms does not require to keep track of the context they are relevant for. Relations between contexts however, are to be expressed in so-called compatibility relations.

**Experiment:**

Authors state that they are trying to establish close collaboration with two other research groups which can be used as experiment of the current paper. They mentioned that one of the groups is responsible for developing the mentioned RDF triplestore and they are currently working on an implementation based on RDF named graphs.

**Claim:**

The authors claim they propose a detailed solution to the problem of modeling contexts in the Semantic Web in a coherent and general way and also an evaluation of the MCS theory. They also mentioned that they are able to put this theory to the test, and explore its limitations. Moreover, Provision of comparative experimentation results, to illustrate
which possibilities exist, how they behave and whether they prove appropriate for real-world applications is other authors’ claim.

5.10. Wang 2004

Wang, C., Xiong, M., Zhou, Q., & Yu, Y. (n.d.). *PANTO: A Portable Natural Language Interface to Ontologies*. APEX Data and Knowledge Management Lab, Department of Computer Science and Engineering, Shanghai JiaoTong University, Shanghai, 200240, P.R.China.

**The Addressed Problem**

In order to attain the formal knowledge in ontologies which is used to represent the taxonomy of domain and plays one of the most important role in semantic web, users are required to be familiar with some formats like the ontology syntax such as RDF and OWL, query languages such as RDQL2 and SPARQL3, the structure of the target ontology. The authors state the previous research which has been done by Bernstein et al. expressed there is undeniable gap between the logic of the semantic web and real-world users. That would be too difficult for users to remember the mentioned format. On the other hand, semantic web requires ontology syntax to have expected efficiency.

**Previous Paper:**

The authors addressed some related papers:

Shortcoming

According to the authors, it is too tough for machines to understand the ordinary natural language because of its ambiguity and complexity. Although NLP community have been attempting to solve and they reach up to 90% in precision and recall, it has so far to recognize words and resolve it.

On the other hand, even parsing natural languages would have been done successfully, there should be some obstacles to translate them to appropriate formal queries. For example the stored vocabularies in knowledge base is totally different with user’s entered words so one of the challenges is to map the user’s words to existing vocabulary in the database.

The authors mention that the other challenges in this area is how to utilize semantic information in knowledge base which includes lexical and syntactic information of parsed queries to acquire the formal query. One more thing is that various representations need different methods to interpret the user queries. Although SPARQL is introduced as standard query language for semantic web community, novel approaches are attempting to translate natural language to them.

New Idea:

The new idea which authors propose includes following major steps: the nominal phrases are extracted from parse tree and they will be shown in intermediate representation which is called QueryTriples and then they are mapped to OntoTriples which are depicted with entities in ontology by PANTO and then OntoTriples will be translated to SPARQL by targets and modifiers extracted from pars tree. The authors state that they investigate some problems with translating natural language queries to SPARQL queries and some complicated and advanced features of the semantic like negation, comparative and superlative modification are considered in their new idea.

Experiment:
The authors implemented PANTO as stand-alone web application, it uses StandfordParse which produces one parse tree for each input. Test data which has been used in this experiment are based on Mooney which is being used to evaluate the natural language interface. Geography data in United States, restaurant information and job announcement are three domains in this dataset. Original Prolog database is translated into OWL as ontologies in current experiment.

The goal of the experiments is to quantitatively assess the performance and effectiveness of our approach in the current implementation.

The authors state to assess the correct rate that how many of the translated queries correctly represent the semantics of the original natural language queries,

The metrics precision and recall are being used to compare the output with the manually generated SPARQL queries. For each domain, precision means the percentage of correctly translated queries in the queries that PANTO produced an output; recall refers to the percentage of queries that PANTO produced an output in the total query set. Since the queries are prepared for evaluating natural language interface to database, some features in the queries are not supported by the current version of SPARQL.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Geography</th>
<th>Restaurant</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original MooneyQueries#</td>
<td>880</td>
<td>250</td>
<td>641</td>
</tr>
<tr>
<td>Selected Testing Queries#</td>
<td>877</td>
<td>238</td>
<td>517</td>
</tr>
<tr>
<td>Precision</td>
<td>88.05%</td>
<td>90.87%</td>
<td>86.12%</td>
</tr>
<tr>
<td>Recall</td>
<td>85.86%</td>
<td>96.64%</td>
<td>89.17%</td>
</tr>
</tbody>
</table>

**Result:**

The authors state that the total processing time of query processing was less than one second (0.878) and running time is based on the scale of the ontology and also complexity of the query which is the length and number of the clauses. The authors claim that precision PANTO provides acceptable result for recall and precision in compared with Querix which achieved a precision 86.08 and recall 87.11. The authors experimented on the Mooney data
and they claim that queries show that all the correctly parsed natural language queries can be correctly translated into QueryTriples and then be mapped to OntoTriples. They claim that their experiments on three different ontologies have shown that the PANTO approach produces promising results also their novel approach helped bridge the gap between the logic-based semantic web and real-world users.

**Claims:**

The authors claim PANTO can cover the whole query scope which are supported by AquaLog since Aqualog can parse queries and PANTO can generate them too.
VITA AUCTORIS

NAME: Elham Emami

PLACE OF BIRTH: Tehran, Iran

YEAR OF BIRTH: 1988

EDUCATION:
- University of Alzahra, B.Sc. Software Engineering, Tehran, Iran, 2006
- University of Windsor, M.Sc. Computer Science, Windsor, ON, 2015