Extending a set-theoretic implementation of Montague Semantics to accommodate n-ary transitive verbs.

Maxim Roy
University of Windsor

Extending a set-theoretic implementation of Montague Semantics to accommodate n-ary transitive verbs

By
Maxim Roy

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
2005

Your file Votre référence
ISBN: 0-494-09850-3
Our file Notre référence
ISBN: 0-494-09850-3

Abstract

Natural-language querying of databases remains an important and challenging area. Many approaches have been proposed over many years yet none of them has provided a comprehensive fully-compositional denotational semantics for a large sub-set of natural language, even for querying first-order non-intentional, non-modal, relational databases. One approach, which has made significant progress, is that which is based on Montague Semantics. Various researchers have helped to develop this approach and have demonstrated its viability. However, none have yet shown how to accommodate transitive verbs of arity greater than two. Our thesis is that existing approaches to the implementation of Montague Semantics in modern functional programming languages can be extended to solve this problem. This thesis is proven through the development of a compositional semantics for n-ary transitive verbs ($n \geq 2$) and implementation in the Miranda programming environment.

[Keywords: Compositional, non-intentional, Montague Semantics, set-theoretic, relational database, denotational]

Dedication

To Sicily

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. T. Collet-Najem, Dr. S. Goodwin and Dr. C. Ezeife for reading my thesis report and for their valuable comments.

And a special thanks to my lovely wife Sicily for her encouragement and support.

Table of Contents

List of Tables

List of Figures

Chapter 1 PREFACE

1.1 Introduction

This report describes the development of an efficient compositional semantics for natural-language queries, which include quantification, nouns, proper nouns, term phrases, verb phrases, negation, coordination ("and" and "or") and n-ary transitive verbs ($n \geq 2$). It is important because it will increase the capability of natural-language query interfaces to databases.

1.2 Outline of the report

This report is about extending a set-theoretic version of Montague Semantics to accommodate n-ary transitive verbs where $n \geq 2$. Chapter 1 contains the thesis statement and motivation for it. Chapter 2 of the report is a general introduction to Montague Semantics. The use of Montague Semantics in database query processing is described in Chapter 3. Chapter 4 describes Montague's approach to transitive verbs. Chapter 5 gives an overview on the use of Montague Semantics in database-query processing. Chapter 6 contains an overview of an existing implementation of a set-theoretic version of a sub-set of Montague Semantics and an extension of the set-theoretic approach to accommodate negation. Chapter 7 discusses the problem addressed in this thesis work - to extend an existing set-theoretic approach to accommodate n-ary transitive verbs where $n \geq 2$, and summarizes two initial approaches that were developed and analyzed as part of this thesis work. Chapter 8 describes the final approach, which we claim supports the thesis. Chapter 9 of the report contains future directions and conclusions. A comprehensive survey on the use of Montague-like semantics is also attached as appendix at the end of the report.

Chapter 2 INTRODUCTION

2.1 Problem addressed

Natural-language querying of databases remains an important and challenging area. Many approaches have been proposed over many years yet none of them has provided a fully-compositional denotational semantics for a large sub-set of natural language even for querying first-order non-intentional, non-modal, relational databases. One approach, which has made significant progress, is that which is based on Montague Semantics. Various researchers have helped to develop this approach and have demonstrated its viability (see appendix B for a comprehensive survey on the use of Montague and Montague-like compositional semantics in natural-language database query processing). After conducting the survey it was found that no one have yet shown how to accommodate transitive verbs of arity greater than two, which is the subject of my thesis.

2.2 Montague's Approach

One of the most influential functional approaches to natural-language interpretation was developed by Richard Montague. In the early seventies, Montague (1971) 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. In particular, he claimed that the objects denoted by phrases of a natural language denote functions in a function space constructed over a set of objects of a few primitive types. For each syntactic category of a natural language, Montague claimed that there is a corresponding semantic type, and for each syntactic rule that shows how a complex syntactic construct can be built from simpler constructs, there is a corresponding semantic rule that shows how the meaning of the complex construct can be computed from the meaning of its parts. Montague was one of the first to develop a compositional semantics for a substantial part of English (details in Chapter 3 & 4). By compositional semantics we mean that the meaning of a compound sentence is determined by the meanings of its constituents and the way they are put together to form the sentence.

2.3 Set-theoretic treatment of Montague Semantics

Montague's approach has been used in the past as the basis for the development of natural-language processors. However, direct implementation of Montague semantics is not computationally viable owing to the fact that the interpretation of phrases involving quantification requires characteristic functions to be applied to all, possibly-infinite number of, representations of entities in the universe of discourse. One approach to overcome this problem was proposed by Frost and Launchbury (1989). In that approach, noun and verb phrases denote sets directly rather than denoting characteristic functions of sets. A fully-compositional simple natural-language database-query processor was built which could accommodate nouns, intransitive verbs, (two-place) transitive verbs, proper nouns, adjectives, determiners, conjunction, disjunction and arbitrarily-nested quantification. The approach was subsequently extended to accommodate negation (Frost and Boulos 2002)(details in Chapter 4).

2.4 Limitation of the set-theoretic approach

The approach to natural-language database-query processing developed by Frost and Launchbury (FL) is very limited in its scope. It does not accommodate modality, intentionality, or transitive verbs with arity greater than two. Work on the first two of these limitations is being undertaken but is a long-term project. The third limitation is the subject of this report.

2.5 Thesis Statement

"It is possible to extend the set-theoretical compositional semantics developed by Frost et al to accommodate n-ary transitive verbs, ($n \geq 2$) by re-defining all denotations to involve sets of attributes rather than simple entities, without loss of compositionality. "

2.6 How the new semantics will be evaluated?

Below we state some specific objectives for our semantics:

1) We will state examples of types of questions that our semantics will be able to handle, give a small grammar for example languages, and show that relatively small sets of semantic definitions can be used to compute the values for languages with hundreds of thousands of queries.
2) The new semantics will maintain the orthogonality of the old semantics, i.e. that the meaning of all (disambiguated) words is independent of context, and that the rules of composition are also independent of context.
3) The new semantics will maintain the syntactic/semantic correspondence i.e. phrases of the same syntactic category denote functions of the same semantic type.

To demonstrate that our semantics meets these objectives we will implement illustrative example query processors based on our semantics, in Miranda, and then:

a) Define example query languages and compute their size, and show results from execution of example queries.
b) Discuss the Miranda program, showing how words are given a single meaning, and give examples of queries where the context is different.
c) Give the results of using the Miranda type inference system (::) on examples of phrases of the same syntactic category to show that they denote functions of the same semantic type.

Orthogonality and the syntactic/semantic relationship guarantee that our semantics will be compositional in the sense that the meaning of expressions of a very large query language can be computed using a very small number of semantic rules.

We are using the functional programming language Miranda only to help to investigate and illustrate our approach to the problem. We will use two problem domains to illustrate the approach. One domain handles queries about the solar system and the other domain handles queries about authors and books.

The objective of our research is not to create a Miranda program for a particular set of queries but to develop a compositional semantics, which could be implemented in different programming languages and which could be used for various databases.

2.7 Why the thesis is important

The FL approach is unable to deal with queries such as "When did Hall discover Phobos?", "With what did Hall discover Phobos?" and "When and who discovered Deimos?" This thesis provides an approach on how to deal with such queries. This thesis is important because it will increase the capability of natural-language query interfaces to databases. The originality of our approach is that implementation of transitive verbs are higher-order functions which return sets of attribute values as results rather than just truth values or lists of entities as in the FL approach.

Our final result is a compositional semantics for quantifications, negation, nouns, proper nouns, term phrases and n-place transitive verbs and helps in building more powerful NL interfaces than existing NL interfaces which convert queries to SQL (which can't handle all forms of negation).

2.8 Contribution to Computational Linguistics, Computer Science and S/W Engineering

Our research contributes to computational linguistics as we are extending an existing linguistics theory and demonstrating the tractabily of its implementation. Montague's semantics didn't provide much about how to handle n-place transitive verbs and here in this thesis work we are extending Montague's semantics by developing a new approach to handle n-place transitive verbs. As discussed in the sub-section 4.2 no one else appears to have solved this problem.

It has a contribution to Computer Science as we are developing a denotational semantics for a natural language by mapping from each syntactic category into a suitable semantic domain. By doing so, we are adding to the growing belief that the denotational semantics approach to programming-language specification has application in natural-language interface design. In computer science, denotational semantics is one of the approaches to formalize the semantics of computer programs. Denotational semantics is a standard tool for language design and definition. The compositional nature of a denotational semantics is a real boon for proving properties of programs and languages. Montague semantics, which we are extending, is a form of denotational semantics for idealized fragments of English.

It also has a contribution to engineering as we are developing a new semantic approach for natural-language query processing, which is efficient and has good software engineering characteristics such as modularity and orthogonality.

Chapter 3 MONTAGUE SEMANTICS

3.1 Introduction to Montague Semantics

Model-theoretic semantics of natural language is a way of analyzing the meanings of NL expressions. The technique was introduced by Richard Montague (1971) in two classical papers entitled "Universal Grammar" and "The Proper Treatment of Quantification in Ordinary English" which is known as PTQ. Universal Grammar, which is a predominantly theoretical paper, refers to the branch of mathematics called universal algebra from which the main techniques were adopted. PTQ, on the other hand, applies these theoretical principles to 'ordinary English'. Grammars based on Montague's PTQ are called Montague grammars.

Richard Montague (1971) was one of the first to develop a compositional semantics for English. Later Partee (1973) describes some extensions to Montague grammar. Bennet (1974) also worked on some extensions. Thomason Richmond (1975) gave an introduction to Montague semantics. Partee's "Montague Grammar and Transformational Grammar" (1975) was seen as the first introductory text to describe Montague Semantics. Dowty, Wall & Peters later published another comprehensive text called "Introduction To Montague Semantics" (1981).

A Montague grammar is a grammar for a particular fragment of natural language which consists of three components: the Syntax, a syntactic analysis of the expressions of the fragment, the Translation, translating natural language into a logical language, and the Model Theory or the Semantics, a (model-theoretic) interpretation of the expressions of the logical language

The translation is "meaning preserving", hence, the meaning assigned to the formula of the logical sentence by the interpretation is also the meaning assigned to the natural language sentence that the formula translates.

In order to be familiar with Montague's PTQ grammar, one should have some knowledge about lambda calculus. The lambda calculus was developed by Alonzo Church (1941). Church recognized that an expression containing x, such as ......x........ defines a function of x. He introduced the notation: λx [formula containing x] as a name for that function. The expression λx [formula containing x] is called an λ-expression or λ-abstraction. A rule of λ-conversion may be written as follows:

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

In the formula [....α......], each free occurrence of x is replaced with α, the result is [....α......].

The following description of Montague Semantics is derived from the book "Montague Semantics" by Dowty, Wall and Peters (1981).

Montague gave some examples of using lambda notation in natural language to define the semantics of expressions. For example considering the following two sentences: "Every man eats." and "Every man sleeps.". The usual translation of this sentence in predicate logic is:

$\forall x(M(x)\rightarrow E(x))$ and $\forall x(M(x) \rightarrow S(x))$

These sentences are instances of a more general sentence whose translation is a second-order logic formula, i.e. they are λ-conversions of the λ-expression:

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

the first conversion is

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

and the second one is

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

In his theory Montague made a distinction between the sense (intension) of an expression and reference (extension). The reference of an expression corresponds to semantic (truth) value of this expression; the sense corresponds to the meaning of the expression. The distinction between sense and reference is important when operators such as *believe* are used. For example, "John believes A" cannot be described as function of the references of its parts but can be described as a function of the senses of these parts. The intensionality in natural language is induced by prepositional attitude verbs such as: *think, believe, regret* etc.

In the rules below of the PTQ grammar we shall denote by α^i the intension of an expression α and by α^e the extension of an expression α. The cancellation rule ^{ei}, which is important in simplification of the expressions produced by the translation of sentences in natural language, is: $\alpha^{i,e} = \alpha^{e,i} = \alpha$

Considering the intensionality, the expressions for determinants *every* and *a* will be:

$\lambda P[\lambda Q[\forall x(P^e(x) \rightarrow Q^e(x))]]$ and $\lambda P[\lambda Q[\exists x(P^e(x) \wedge Q^e(x))]]$ respectively.

Before introducing the rules of the PTQ grammar the categories used in the PTQ grammar are explained below:

Category Name	Categorical Definition of Name	Nearest Transformational Equivalent
t	t	Sentence
CN	CN	Common noun
IV	IV	Intransitive Verb
T	t/IV	Term Phrases and Proper Name
IAV	IV/IV	Intransitive Adverb
TV	IV/T (=IV/(t/IV))	Transitive Verb
T/CN	(t/IV)/CN	Determiner
t/t	t/t	Sentence Adverb
IV/t	IV/t	Sentence-complement Verb
IV//IV	IV//IV	Infinitive-complement Verb
IAV/T	(IV/IV)/T	Preposition

Table: 2-1. Categories used in PTQ grammar

In a categorical grammar an expression of category A/B (or of A//B) combines with an expression of category B to give an expression of category A. The different between A/B and A//B is that both are distinct categories but denote values of the same logical type.

Syntactic rules in the PTQ grammar forming complex expressions have the following general form:

S_n: If $\alpha \in P_A$ and $\beta \in P_B$ then $F_n(\alpha, \beta) \in P_C$, where $F_n(\alpha, \beta)$ is.........

Here n is the number of the syntactic rule, A and B are the syntactic categories of the input expressions, C is the syntactic category of the new expression formed by the rule and F_n is the name of the structural operation of the rule and in place of the ellipsis is a description of what this operation does.

Now some of the rules of PTQ grammar with examples are described. Readers who are familiar with Montague Semantics can skip to chapter 4 on page 20.

3.2 Subject-Predicate and Determiner-noun rules

One of the first rules described in PTQ is **S4**, which is a syntactic rule and is called the subject-predicate rule. This rule takes a term phrase and a verb phrase and combines them to form a sentence. The rule is as follows:

S4: If $\alpha \in P_T$ and $\beta \in P_{IV}$ then $F_4(\alpha, \beta) \in P_t$, where $F_4(\alpha, \beta) = \alpha\beta'$, and $\beta'$is the result of replacing the first verb in β by its third person singular present form.

Each syntactic rule S_n has associated with it a translation rule T_n with the same numerical subscript. The translation rule associated with S4 is as follows:

T4: If $\alpha \in P_T$ and $\beta \in P_{IV}$ and α, β translate into α', β' respectively, then $F_4(\alpha, \beta)$ translates into $\alpha'(\beta'^{i})$

Now let us consider a simple English sentence and translate it according to the rule. For example, let us consider the sentence: **Tom talks.**

Tom translates into $\lambda P\ [P^e(t)]$

where P is a variable over properties of individuals and t is an individual constant corresponding to the person called Tom.

Now translating the sentence according to the rule T4 will be as follows:

Tom talks

1. Tom $\Rightarrow \lambda P\ [\ P^e\ (t)]$
2. talks $\Rightarrow$ talk'
3. $\lambda P\ [P^e(t)](talk'^{i})$ [From 1,2 by T4]
4. $talk'^{e,i}\ (t)$ [Lambda conversion]
5. talk' (t) [Cancellation rule]

Now let us consider the second rule **S2** that is called the Determiner-Noun rule. This rule combines words like **every** with words like **human** to produce partial sentences like **every human** or **a** with **dog** to give partial sentences like **a dog** etc. The rule is as follows:

S2: If $\alpha \in P_{T/CN}$ and $\beta \in P_{CN}$ then $F_2(\alpha, \beta) \in P_T$, where $F_2(\alpha, \beta) = \alpha'\ \beta$ and α' is α

except in the case where α is a and the first word in β begins with a vowel; here α' is an.

The associated translation rule of S2 is T2, which is as follows:

T2: If $\alpha \in P_{T/CN}$ and $\beta \in P_{CN}$ and α ,β translate into α', β' respectively, then $F_2(\alpha, \beta)$ translates into $\alpha'(\beta'^{i})$

The below example sentences can be translated first using the rule **T2** and then the rule **T4.** For Example,

Every human walks.
The sun shines.
A dog barks.

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: $\lambda P\ [\lambda Q\ \exists x\ [\ P^e(x) \wedge Q^e(x)]]$

Now translating the first sentence above according to the rules T2 and T4 is as follows:

Every human walks.

1. every $\Rightarrow$ $\lambda P[\lambda Q \forall x[\ P^e(x) \rightarrow Q^e(x)]]$
2. human $\Rightarrow$ human$'$
3. every human $\Rightarrow$ $\lambda P[\lambda Q \forall x[P^e(x) \rightarrow Q^e(x)]](\text{human}'^{i})$ [From 1,2 by T2]
4. $\lambda Q \forall x[\text{human}'^{e,i}(x) \rightarrow Q^e(x)]$ [Lambda conversion]
5. $\lambda Q \forall x[\text{human}'(x) \rightarrow Q^e(x)]$ [Cancellation Rule]
6. walks $\Rightarrow$ walks$'$
7. every human walks $\Rightarrow$ $\lambda Q \forall x\ [\text{human}'(x) \rightarrow Q^e(x]\ (\text{walks}'^{i})$ [From 5,6 by T4]

9. $\forall x[\text{human}'(x) \rightarrow \text{walks}'^{e,i}(x)]$ [Lambda conversion]
10. $\forall x[\text{human}'(x) \rightarrow \text{walks}'(x)]$ [Cancellation Rule]

Similarly the other two sentences can be translated using the rule T2 and the rule T4.

The sun shines can be translated into:

$\exists x\ [\forall x\ [\text{sun}'(x) \Leftrightarrow x\ = y]\ \wedge \text{shines}'(y)]$

And **A dog barks** can be translated into:

$\exists x\ [\ \text{dog}'(x) \wedge \text{barks}'(x)]$

3.3 Conjoined Sentences, Verb Phrases and Term Phrases

The first conjunction rule described in PTQ is S11a, which takes two sentences such as **Tom talks** and **every human walks** and combines them by rule S11 to produces sentences like **tom talks and every human walks**. The rule is as follows:

S11a: If $\alpha, \beta \in P_t$, then $F_{11a}(\alpha, \beta) \in P_t$, where $F_{11a}(\alpha, \beta) = \alpha$ and β

The associated translation rule of S11a is T11a which is:

T11a: If $\alpha, \beta \in P_t$ and α ,β translate into α', β' respectively, then $F_{11a}(\alpha, \beta)$ translates into $[\alpha' \wedge \beta']$

For example, with the help of the rule T11a we can translate a simple English sentence like **Tom talks and every human walks.** into :

$[\text{ talk'}(t) \wedge \forall x [\text{ human'}(x) \rightarrow \text{walks'}(x)]]$

where **Tom talks** can be translated into **talk'(t)** and **every human walks** can be translated into $\forall x [\text{ human'}(x) \rightarrow \text{walks'}(x)]$ [Derived in section 2]

The rule S11b is as follows:

S11b: If $\alpha, \beta \in P_t$, then $F_{11}(\alpha, \beta) \in P_t$, where $F_{11b}(\alpha, \beta) = \alpha$ or β

The associated translation rule of S11b is T11b, which is:

T11b: If $\alpha, \beta \in P_t$ and α ,β translate into α', β' respectively, then $F_{11b}(\alpha, \beta)$ translates into $[\alpha' \vee \beta']$

Another three conjunction rule are **S12a, S12b** and **S13**. The rules **S12a** and **S12b** take two verb phrases (member's of P_{IV}) and combines them together. And the rule **S13** takes two term phrases (member's of P_T) and combines them together. The rules **S12a**, **S12b** and **S13** are as follows:

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

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

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

The associated translation rule of S12a , S12b and S13 are:

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

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

T13: If α, $\beta \in P_T$ and α ,β translate into α', β' respectively, then $F_{13}(\alpha, \beta)$ translates into $\lambda P\ [\alpha'(P) \vee \beta'(P)]$

Now let as consider a simple English sentence for example **Every human walks or talks**. And translate the sentence with the help of the rule T12b.

Every human walks or talks.

1. every human $\Rightarrow$ $\lambda Q \forall x[\text{human}'(x) \rightarrow Q^e(x)]$
   [Derived in section 2]
2. walk $\Rightarrow$ walk'
3. talk $\Rightarrow$ talk'
4. walk or talk $\Rightarrow$ $\lambda x[\text{walk}'(x) \vee \text{talk}'(x)]$ [From 2,3 by T12b]
5. every human walks or talks $\Rightarrow$
   $\lambda Q \forall y[\text{human}(y) \rightarrow Q^e(y)](\lambda x[\text{walk}'(x) \vee \text{talk}'(x)]^i)$
   [From 1,4 by T4]
6. $\forall y[\text{human}'(y) \rightarrow \lambda x[\text{walk}'(x) \vee \text{talk}'(x)]^{e,i}(y)]$
   [Lambda conversion]
7. $\forall y[\text{human}'(y) \rightarrow \lambda x[\text{walk}'(x) \vee \text{talk}'(x)](y)]$
   [Cancellation Rule]
8. $\forall y[\text{human}'(y) \rightarrow [\text{walk}'(y) \vee \text{talk}'(y)]]$
   [Lambda conversion]

3.4 Anaphoric Pronouns as bound variables; Scope Ambiguities and related clauses

The rule that deals with pronouns is the rule **S14**. The rule **S14** is as follows:

S14: If $\alpha \in P_T$, $\beta \in P_t$, then $F_{14,n}(\alpha, \beta) \in P_t$

The translation rule of S14 is:

T14: If $\alpha \in P_T$, $\beta \in P_t$ and α ,β translate into α', β' respectively, then $F_{14,n}(\alpha, \beta)$ translates into $\alpha'(\lambda x_n\ \beta'^i)$

Now for example we will use the rule **T14** to translated sentences like **A women sings and she dances.** Pronouns are translated as follows:

he$_n$ translates into $\lambda P\ [\ P^e(x_n)]$

So the translation is of the above example using the rule **T14** is as follows:

A women sings and she dances.

1. he_2 $\Rightarrow$ $\lambda P[P^e(x_2)]$
2. sing $\Rightarrow$ sing'
3. dance $\Rightarrow$ dances'
4. he_2 sings $\Rightarrow$ $\lambda P\ [\ P^e(x_2)](\text{sing}'^i)$ [From 1,2 by T4]
5. $\text{sing}'^{i,e}(\ x_2\ )$ [Lambda conversion]

6. $\text{sing}'(x_2)$ [Cancellation Rule]
7. he_2 dances $\Rightarrow \lambda P\,[P^e(x_2)]\,(\text{dance}'^{i})$ [From 1,3 by T4]
8. $\text{dance}'^{i,e}(x_2)$ [Lambda conversion]
9. $\text{dance}'(x_2)$ [Cancellation Rule]
10. he_2 sings and he_2 dances $\Rightarrow$
$[\text{sing}'(x_2) \wedge \text{dance}'(x_2)]$
11. a woman $\Rightarrow \lambda P\,\exists x\,[\text{woman}'(x) \wedge P^e(x)]$
12. a woman sings and she dances $\Rightarrow$
$\lambda P \exists x[\text{woman}'(x) \wedge P^e(x)]\,(\lambda x_2[\text{sing}'(x_2) \wedge \text{dance}'(x_2)]^{i})$
[From 10,11 by T14]
13. $\exists x\,[\text{woman}'(x) \wedge \lambda x_2[\text{sing}'(x_2) \wedge \text{dance}'(x_2)]^{i,e}(x)]$
[Lambda convention]
14. $\exists x\,[\text{woman}'(x) \wedge \lambda x_2[\text{sing}'(x_2) \wedge \text{dance}'(x_2)](x)]$
[Cancellation Rule]
15. $\exists x\,[\text{woman}'(x) \wedge [\text{sing}'(x) \wedge \text{dance}'(x)]]$
[Lambda conversion]

To illustrate how **S14** accounts *de dicto/de re* ambiguities in complements of verbs like *believe*, the syntactic rule **S7** has been introduced . The rule **S7** is as follows:

S7 : If $\alpha \in P_{IV/t}$, $\beta \in P_t$, then $F_7(\alpha, \beta) \in P_{IV}$, where $F_7(\alpha, \beta) = \alpha$ **that** β

The translation rule of S7 is :

T7 : If $\alpha \in P_{IV/t}$, $\beta \in P_t$ and α ,β translate into α', β' respectively, then $F_7(\alpha, \beta)$ translates into $\alpha'(\beta'^{i})$

For example the sentence **Tom believes that a dog barks** can be translated as described below with the help of the rule **T7.**

Tom believes that a dog barks.
1. a dog barks $\Rightarrow \exists x\,[\text{dog}'(x) \wedge \text{barks}'(x)]$
2. believe $\Rightarrow$ believe'
3. believe that a dog barks $\Rightarrow$
$\text{believe}'(\exists x\,[\text{dog}'(x) \wedge \text{barks}'(x)]^{i})$ [From 1,2 by T7]
4. Tom $\Rightarrow \lambda P\,[P^e(t)]$
5. Tom believe that a dog barks $\Rightarrow$
$\lambda P[P^e(t)](\text{believe}'(\exists x[\text{dog}'(x) \wedge \text{barks}'(x)]^{i})^{i}$
[From 3,4 by T4]
6. $\text{believe}'(\exists x\,[\text{dog}'(x) \wedge \text{barks}'(x)]^{i})^{i,e}(t)$
[Lambda conversion]
7. $\text{believe}'(t, \exists x[\text{dog}'(x) \wedge \text{barks}'(x)]^{i})$
[Relational notation and Cancellation rule]

Another way of translating the above sentence is:

Tom believes that a dog barks.

1. he$_2$ barks $\Rightarrow$ bark$'(x_2)$
2. believe that he$_2$ barks $\Rightarrow$
   believe$'([\text{bark}'(x_2)]^i\ )$ [By T7]
3. Tom believes that he$_5$ barks $\Rightarrow$
   $\lambda P[P^e(t)]$ (believe$'([\text{bark}'(x_2)]^i\ )^i)$ [By T4]
4. believe$'([\text{bark}'(x_2)]^i)^{i,e}(t)$ [Lambda conversion]
5. believe$'(t, [\text{bark}'(x_2)]^i)$ [Relational notation and Cancellation rule]
6. a dog $\Rightarrow \lambda P\ \exists x\ [\ \text{dog}'(x) \wedge P^e\ (x)]$
7. Tom believes that a dog barks $\Rightarrow$
   $\lambda P \exists x\ [\text{dog}'(x) \wedge P^e(x)]\ (\ \lambda x_2\ [\text{believe}'(t, [\text{bark}'(x_2)]^i)]^i)$
   [By T14]
8. $\exists x[\text{dog}'(x) \wedge \lambda x_2[\text{believe}'(t, [\text{bark}'(x_2)]^i)]^{i\ ,e}(x)]$
   [Lambda conversion]
9. $\exists x[\text{dog}'(x)\ \wedge \text{believe}'(t, [\text{bark}'(x)]^i)]$
   [Lambda conversion]

Similarly, the sentence **Every human believes that a dog barks** can be translated into:

$\exists x[\text{dog'}(x) \wedge \forall y[\text{human'}(y) \rightarrow \text{believe'}\ (y\ , [\text{bark'}(x)]^i)]]$

and

$\forall y[\text{human'}(y) \rightarrow \exists x[\text{dog'}(x) \wedge \text{believe'}\ (y\ , [\text{bark'}(x)]^i)]]$

The rule **S3** which is a relative clause rule takes a common noun and a sentence and outputs a new phrase of the category common noun(CN). The rule S3 is as follows:

S3: If $\alpha \in P_{CN}$ and $\beta \in P_t$, then $F_{3,n}(\alpha, \beta) \in P_{CN}$ where $F_{3,n}(\alpha, \beta) = \alpha$ **such that** β'

The translation rule of S3 is:

T3: If $\alpha \in P_{CN}$, $\beta \in P_t$ and α ,β translate into α', β' respectively, then $F_{3,n}(\alpha, \beta)$ translates into $\lambda x_n [\alpha'(x_n)\ \wedge \beta')]$

For example English sentences like **Every dog such that it barks runs** can be translated using the rule **T3.** The translation is:

Every dog such that it barks runs

1. he$_2$ barks $\Rightarrow$ bark$'(x_2)$ [previously derived]
2. dog such that it barks $\Rightarrow$
   $\lambda x_2\ [\ \text{dog}'(x_2)\ \wedge \text{bark}'(x_2)]$ [By T3]
3. every dog such that it barks $\Rightarrow$
   $\lambda P \lambda Q \forall x[P^e(x) \rightarrow Q^e(x)]\ (\lambda x_2[\text{dog}'(x_2)\ \wedge \text{bark}'(x_2)]^i)$ [By T2]
4. $\lambda Q \forall x[\lambda x_2[\text{dog}'(x_2)\ \wedge \text{bark}'(x_2)]^{i\ ,e}(x)\ \rightarrow Q^e(x)]$
   [Lambda conversion]

5.$\lambda Q \forall x[\lambda x_2 \text{ [dog}'(x_2) \wedge \text{bark}'(x_2)](x) \rightarrow Q^e(x)]$
[Cancellation Rule]

6.$\lambda Q \forall x[[\text{dog}'(x) \wedge \text{bark}'(x)] \rightarrow Q^e(x)]$
[Lambda conversion]

7.every dog such that it barks runs $\Rightarrow$
$\lambda Q \forall x \;[[\text{dog}'(x) \wedge \text{bark}'(x)] \rightarrow Q^e(x)](\text{runs}'^{i})$ [By T4]

8.$\forall x \;[[\text{dog}'(x) \wedge \text{bark}'(x)] \rightarrow \text{runs}'(x)]$
[Lambda conversion, Cancellation Rule]

3.5 Transitive verbs, Meaning Postulates And Non-Specific Readings

The rule S5 combines a transitive verb with a term phrase and outputs an IV-phrase (Intransitive Verb). The rule S5 is as follows:

S5: If $\alpha \in P_{TV}$, $\beta \in P_T$, then $F_5(\alpha, \beta) \in P_{IV}$ where $F_5(\alpha, \beta) = \alpha\beta$
The translation rule of S5 is:
T5: If $\alpha \in P_{TV}$, $\beta \in P_T$ and α, β translate into α', β' respectively, then $F_5(\alpha, \beta)$ translates into $\alpha'(\beta'^{i})$
For example sentence like **Tom seeks a dog.** can be translates using the rule T5 as follows:

Tom seeks a dog

1.seek $\Rightarrow$ seek'

2.a dog $\Rightarrow$ $\lambda Q \exists x[\text{dog}'(x) \wedge Q^e(x)]$ [Previously derived]

3.seek a dog $\Rightarrow$ $\text{seek}'(\lambda Q \;\exists x[\text{dog}'(x) \wedge Q^e(x)]^{i})$
[From 1,2 by T5]

4.Tom seeks a dog $\Rightarrow$
$\lambda P \;[P^e(t)] \;(\text{seek}'(\lambda Q \;\exists x[\text{dog}'(x) \wedge Q^e(x)]^{i})^{i})$ [By T4]

5.$\text{seek}'(\lambda Q \;\exists x \;[\text{dog}'(x) \wedge Q^e(x)]^{i})^{i\,'e}(t)$ [Lambda conversion]

6.$\text{seek}'(\lambda Q \exists x \;[\text{dog}'(x) \wedge Q^e(x)]^{i})(t)$ [Cancellation Rule]

7.$\text{seek}'(t, \;\lambda Q \exists x \;[\text{dog}'(x) \wedge Q^e(x)]^{i})$ [Relational notation]

At this point no more simplification is possible.

Another way of translating the above sentence is:

Tom seeks a dog.

1. $\text{he}_0 \Rightarrow \lambda P[P^e(x_0)]$ [Basic expression]

2. seek $\Rightarrow$ seek'

3. seek $\text{him}_0 \Rightarrow$
$\text{seek}'(\lambda P[P^e \;(x_0)]^{i})$ [By T5]

4.Tom seeks $\text{him}_0 \Rightarrow$
$\lambda P[P^e \;(t)](\text{seek}'(\;\lambda P[P^e(x_0)]^{i})^{i}$ [By T4]

5.$\text{seek}'(\;\lambda P \;[\;P^e(x_0)]^{i})^{i\,'e}(t)$ [Lambda conversion]

6. $seek'(t, \lambda P [P^e(x_0)]^i)$ [Cancellation Rule and Relational Notation]

7. a dog $\Rightarrow$ $\lambda Q \exists x [dog'(x) \wedge Q^e(x)]$ [Previously derived]

8. Tom seeks a dog $\Rightarrow$
$\lambda Q \exists x [dog'(x) \wedge Q^e(x)](\lambda x_0 [seek'(t,\lambda P [P^e(x_0)]^i)]^i)$ [By T14]

9. $\exists x [dog'(x) \wedge \lambda x_0 [seek'(t, \lambda P [P^e(x_0)]^i)]^{i,e}(x)]$ [Lambda conversion]

10. $\exists x [dog'(x) \wedge \lambda x_0 [seek'(t,\lambda P [P^e(x_0)]^i)](x)]$ [Cancellation rule]

11. $\exists x [dog'(x) \wedge [seek'(t, \lambda P [P^e(x)]^i)]]$ [Lambda conversion]

Montague introduced a special notation:

$$\delta_* = \lambda y \lambda x [\delta(\lambda P[P^e(y)]^i)(x)], \text{ where } \delta \in ME_{f(TV)}$$

So, now we can continue from 11 from above:

11. $\exists x[dog'(x) \wedge [seek'(t, \lambda P [P^e(x)]^i)]]$

12. $\exists x[dog'(x) \wedge [seek'(\lambda P [P^e(x)]^i)(t)]]$ [Relational notation]

13. $\exists x[dog'(x) \wedge [\lambda z [seek'(\lambda P[P^e(x)]^i)(z)(t)]]$ [λ-conversion]

14. $\exists x[dog'(x) \wedge [\lambda y [\lambda z [seek'(\lambda P [P^e(y)]^i)(z)]](x)(t)]]$ [λ-conversion]

15. $\exists x[dog'(x) \wedge [seek'_*(x)(t)]]$ [δ_* notation]

16. $\exists x[dog'(x) \wedge [seek'_*(t, x)]]$ [Relation notation]

The verb **be** is not translated into a non-logical constant **be'**, but is translated as :

be translates into $\lambda \Phi \lambda x \Phi(\lambda y [x = y]^i)^e$

This is the most complex expression assigned as a translation of any English word in Montague semantics (according to Dowty et al, 1981). The best way to understand it is to first compute a translation using it. Some examples are derived below.

Tom is John.

1. be $\Rightarrow$ $\lambda \Phi \lambda x \Phi(\lambda y [x = y]^i)^e$ [Basic expression]

2. John $\Rightarrow$ $\lambda P [P^e(j)]$ [Basic expression]

3. be John $\Rightarrow$
$\lambda \Phi \lambda x \Phi(\lambda y [x = y]^i)^e (\lambda P [P^e(j)]^i)$ [By T5]

4. $\lambda x \lambda P[P^e(j)]^{e,i}(\lambda y [x = y]^i)$ [Lambda conversion]

5. $\lambda x \lambda P[P^e(j)] (\lambda y [x = y]^i)$ [Cancellation Rule]

6. $\lambda x[\lambda y [x = y]^{i,e}(j)]$ [Lambda conversion]

7. $\lambda x [x = j]$ [Lambda conversion]

8. Tom is John $\Rightarrow$ $\lambda P [P^e(t)](\lambda x[x = j]^i)$ [By T4]

9.$\lambda x\ [\ x = j\ ]^{e,i}\ (t)$ [Lambda conversion]
10.$\lambda x\ [\ x = j\ ]\ (t)$ [Cancellation Rule]
11.$t = j$ [Lambda conversion]

Tom is a human.
1.a human $\Rightarrow \lambda Q\ \exists x\ [human'(x) \wedge Q^{e}(x)]$ [Previously derived]
2.be a human $\Rightarrow \lambda\Phi\lambda x\ \Phi(\ \lambda y[x = y\ ]^{i})^{e}(\ \lambda Q\exists x\ [human'(x) \wedge Q^{e}(x)]^{i}\ )$ [By T5]
3.$\lambda\Phi\lambda x\ \Phi(\lambda y[\ x = y\ ]^{i})^{e}(\ \lambda Q\ \exists z\ [human'(z) \wedge Q^{e}(z)]^{i}\ )$ [Alphabetic variant of 2]
4.$\lambda x\ \lambda Q\exists z[human'(z) \wedge Q^{e}(z)]^{i}\ (\lambda y[\ x = y]^{i})^{e}$ [Lambda conversion]
5.$\lambda x\ \lambda Q\exists z[human'(z) \wedge Q^{e}(z)]\ (\lambda y[\ x = y\ ]^{i})$ [Cancellation Rule]
6.$\lambda x\ \exists z[human'(z) \wedge \lambda y[\ x = y]^{i,e}(z)]$ [Lambda conversion]
7.$\lambda x\ \exists z[human'(z) \wedge \lambda y[\ x = y\ ](z)]$ [Cancellation Rule]
8.$\lambda x\ \exists z\ [human'(z) \wedge x = z]$ [Lambda conversion]
9.Tom is a human $\Rightarrow \lambda P[P^{e}(t)](\lambda x\exists z[human'(z) \wedge x = z\ ]^{i})$ [By T4]
10.$\lambda x\ \exists z\ [human'(z) \wedge x = z\ ]^{i,e}(t)$ [Lambda conversion]
11.$\lambda x\ \exists z\ [human'(z) \wedge x = z\ ](t)$ [Cancellation Rule]
12.$\exists z\ [human'(z) \wedge\ t= z\ ]$ [Lambda conversion]
13.$human'(t)$ [By principle of first-order logic with identity]

3.6 Adverbs And Infinitive Complement Verbs

The PTQ grammar includes both sentence adverbs such as **necessarily** and verb-phrases adverbs such as **slowly**. The adverb necessarily which doesn't translate into a non-logical constant is translated in terms of a special symbol . The adverb **necessarily** translates into $\lambda p\ [\quad p]^{e}$ where p is a variable over propositions.

For example sentence like **Necessarily Tom talks**. Can be translated as follows:

Necessarily Tom talks
1. he_2 talks $\Rightarrow talk'(x_2)$ [previously derived]
2.necessarily $\Rightarrow \lambda p\ [\quad p]^{e}$ [Basic Expression]
3.necessarily he_2 talks $\Rightarrow \lambda p\ [\quad p]^{e}\ ([talk'(x_2)]^{i})$ [By T2]
4. $[\ talk'(x_2)]$ [Lambda conversion and Cancellation rule]
5.necessarily Tom talks $\Rightarrow$
$\lambda P[P^{e}(t)](\lambda x_2\ [talk'(x_2)]^{i})$ [By T14]
6.$\lambda x_2\ [talk'(x_2)](t)$ [Lambda conversion and Cancellation Rule]
7. $[talk'(t)]$ [Lambda conversion]

And sentence like **Tom runs slowly** can be translated as:
slowly'(^run')(t)

3.7 Negation

Montague didn't explain in details how to deal with negation but gave a rough idea about it. The rule **S17** deals with negation. The rule **S17** is:

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 β' is the result of replacing the first verb in β by its negative third person singular present;

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

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

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

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

The translation of S17 is as follows:

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

$F_{17a}(\alpha, \beta)$ translates into $\neg\alpha'(\, \beta'^{i})$

$F_{17b}(\alpha, \beta)$ translates into $F\alpha'(\, \beta'^{i})$

$F_{17c}(\alpha, \beta)$ translates into $\neg F\alpha'(\, \beta'^{i})$

$F_{17d}(\alpha, \beta)$ translates into $P\alpha'(\, \beta'^{i})$

$F_{17e}(\alpha, \beta)$ translates into $\neg P\alpha'(\, \beta'^{i})$

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 human doesn't run** can be translated to:

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

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

Chapter 4 MONTAGUE AND TRANSITIVE VERBS

4.1 Montague's approach to transitive verbs

Montague didn't mention much about n-place transitive verbs where n>2. In his semantics he described briefly how to handle 2-place transitive verbs and stated that it could be extended to handle n-place transitive verbs. Below we describe with examples how 2-place transitive verbs are handled in Montague's Semantics.

According to Montague, the semantic type of transitive verbs like "orbits" is as follows:

$f(TV) = f(IV/T) = << s, f(T) > ,f(IV)>$

$= <<s,<<s, f(IV)>, t>>,<<s,e>t>>$

$= <<s,<<s,<<s,e>,t>>,t>>,<<s,e>t>>$

As discussed in the previous chapter the rule S5 combines a transitive verb with a term phrase and outputs an IV-phrase (Intransitive Verb). The rule S5 is as follows:

S5: If $\alpha \in P_{TV}$, $\beta \in P_T$, then $F_5(\alpha, \beta) \in P_{IV}$ where $F_5(\alpha, \beta) = \alpha\beta$

The translation rule of S5 is:

T5: If $\alpha \in P_{TV}$, $\beta \in P_T$ and α ,β translate into α', β' respectively, then $F_5(\alpha, \beta)$ translates into $\alpha'(\beta'^{i})$

For example sentence like **Phobos orbits Mars.** can be translates using the rule T5 as follows:

Phobos orbits Mars

1.orbits $\Rightarrow$ orbits'

2.Mars$\Rightarrow$ $\lambda Q[Q^e(e\_mars)]$

3.orbits Mars $\Rightarrow$ orbits'$(\lambda Q\ [Q^e(e\_mars)]^i)$

[From 1,2 by T5]

4. Phobos $\Rightarrow$ $\lambda P\ [P^e(e\_phobos)]$

5.Phobos orbits Mars$\Rightarrow$

$\lambda P\ [P^e(e\_phobos)]\ (orbits'(\lambda Q\ [Q^e(e\_mars)]^i)^i)$ [By T4]

6.$(orbits'(\lambda Q\ [Q^e(e\_mars)]^i)^{i\ ,e})(e\_phobos)$ [Lambda conversion]

7.$orbits'(\lambda Q[Q^e(e\_mars)]^i)(e\_phobos)$ [Cancellation Rule]

8.orbits'(e_phobos, λQ[Q^e(e_mars)]^i) [Relational notation]

At this point no more simplification is possible.

Now Montague introduced a special notation:

$\delta_* = \lambda y \lambda x\ [\ \delta(\ \lambda P[P^e(y)]^i)(x)]$, where $\delta \in ME_{f(TV)}$

where δ_* is the binary relation associated with the denotation δ of the transitive verb. Note here that Montague does not give the meaning of the transitive verbs, for example [orbits] directly, consequently, his treatment of transitive verbs is very difficult to understand.

Now we continue from 8 from above:

```
8. orbits'(e_phobos, λQ[Q^e(e_mars)]^i)
9. orbits'( λQ [ Q^e(e_mars)]^i)(e_phobos)
                                        [Relational notation]
10. λz [orbits'(λQ[Q^e(e_mars)]^i)(z)(e_phobos)]
                                        [λ-conversion]
11.[λy [λz [orbits'( λQ [ Q^e(y)]^i)(z)]](e_mars)(e_phobos)]
                                        [λ-conversion]
12.[orbits'*(e_mars)(e_phobos)]   [δ* notation]
13. orbits'*(e_phobos, e_mars) [Relation notation]
```

So in Montague semantics **Phobos orbits Mars** is derived as **orbits'*(e_phobos, e_mars)**

4.2 Frost's approach to Transitive verbs

Below we derive the same expression **(Phobos orbits Mars)** in a simplified statement of Montague's transitive verbs (no intensionality), using an approach developed by Frost (unpublished communication). In this approach transitive verbs (for example "orbit") denote functions such as:

orbit => λz z (λyλx orbit_rel(x,y)).

Therefore, **Phobos orbits Mars**

1. **Phobos** ⇒ (λp p e_phobos)
2. **Mars** ⇒ (λq q e_mars)

3. **orbits** $\Rightarrow$ λz z (λyλx orbit_rel(x,y))
4. **orbits Mars** $\Rightarrow$ λz z (λyλx orbit_rel(x,y)) (λq q e_mars) [From 2 and 3]
5. **Phobos orbits Mars** $\Rightarrow$

   (λp p e_phobos) (λz z (λyλx orbit_rel(x,y)) (λq q e_mars)) [From 1 and 4]
6. (λp p e_phobos) ((λq q e_mars)(λyλx orbit_rel(x,y))) [Lambda conversion]
7. (λp p e_phobos) (λyλx orbit_rel(x,y) e_mars) [Lambda conversion]
8. (λp p e_phobos) (λx orbit_rel(x,e_mars)) [Lambda conversion]
9. (λx orbit_rel(x,e_mars) e_phobos) [Lambda conversion]
10. orbit_rel(e_phobos, e_mars) [Lambda conversion]

Note that this approach does not accommodate intensional and modal aspect of sentences. However, it was useful when developing the set-theoretic semantics for transitive verbs described in the next chapter.

4.3 Other researchers' treatment of transitive verbs in Montague-like semantics

McCawley (1974), Karttunen (1976), Ross (1976) and Larson (1997) talked about intensional transitive verbs but didn't discuss n-ary transitive verbs, $n \geq 2$.

The book "English Verb Classes and Alternations: A Preliminary Investigation" by Beth Levin also talks about transitive verbs but nothing about n-place transitive verbs.

(Miyagawa, Tsujioka, 2004) describes how to handle 3-place verbs in Japanese language, but nothing about other cases, n>3.

Other researchers like Partee, Dowty and Peter have talked about Montague semantics but have not provided any approaches to handle n-place transitive verbs.

Chapter 5 OVERVIEW OF USE OF MONTAGUE SEMANTICS IN DATABASE QUERY PROCESSING

5.1 Overview

There are many advantages to provide users with natural-language interfaces to data sources. In particular, when speech-recognition technology is used, it is useful to phrase queries in some form of pseudo natural-language as it is very difficult to "speak" a language such as SQL.

There are two ways to construct natural-language processors: by translation to a formal language such as SQL, or by direct interpretation by an evaluator based on some form of compositional semantics. The second approach has some advantages: 1) information concerning sub-phrases of the query, such as cost and size, can be presented to the user in an intelligible form before the query is processed, 2) for query-debugging purposes, the user can ask for the value of sub-phrases to be presented before the whole query is evaluated, 3) and the sub-set of natural-language can be readily extended if the evaluator has a modular structure based on the compositional semantics.

5.2 Montague Semantics in Database querying:

According to (Yonezaki and Enomoto 1980), Montague's Intensional Logic (IL) can be useful to the theory of databases in designing database systems, which handle historical data and provide a formal description of database semantics. However, they noted that direct implementation is impractical.

Variations of Montague Semantics have been proposed used as a semantic basis in a number of implemented systems for natural language (e.g. Clifford 1990), but none discussed transitive verbs, n >2.

(Frost and Launchbury, 1989) describe how in a functional-programming language, efficient natural-language parsers and interpreters can be implemented using a se-theoretic version of Montague Semantics. Frost and Launchbury refer to the book by Dowty, Wall and Peters (1981). It appears that Frost and Launchbury were amongst the first to use Montague semantics in database query processing.

It would also appear that Frost and Launchbury were the first to use a set-theoretic based implementation of Montague semantics. For example instead of interpreting 'every' like in Montague as:

$$[\text{every}] = \lambda p \lambda q\ [\ \forall p(x) \rightarrow q(x)\ ]$$

Frost and Launchbury used:

$$[\text{every}]_{FL} = \lambda p\ \lambda q\ \ p\ \subseteq q$$

(Frost and Saba 1990) implemented some of the concepts of Montague that can be used in natural-language interface to databases in an executable attribute grammar.

(Lapalme and Lavier 1990) showed how a larger part of Montague Semantics can be implemented in a pure higher-order functional programming language, but did not use a set-theoretic approach and were not concerned with efficiency.

(Frost and Boulos, 2002) developed an extension to the set-theoretic-based compositional semantics to accommodate phrases that include the word 'no'. The approach is based on an extended set theory in which 'negative' phrases denote infinite sets represented in complement form.

The below table shows the research on Montague Semantics in database query processing:

Year	Authors	Work
1980	Yonezaki and Enomoto	Montague's Intensional Logic (IL) can be useful in designing database systems which handle historical data
1989	Frost and Launchbury	Implementation of a set-theoretic version of sub-set of Montague Semantics
1990	Lapalme and Lavier	Montague Semantics can be implemented in a pure higher-order functional programming language
1990	Frost and Saba	Used Montague Semantics to implement natural-language interfaces to databases
2002	Frost and Boulos	Implemented compositional semantics for database queries based on a set-theoretic version of Montague semantics to accommodate negation

Table 4.1: Coverage of Natural-Language Semantics

5.3 Natural-language Interfaces based on SQL

Many of natural-language interfaces that have been developed are based on an SQL type approach. The following references are provided so that reader can compare SQL-based approaches with the Montague-based approach described in this thesis.

(Hasting, 1991) describes the design and implementation of an SQL based speech-recognition database-query system.

(Androutsopoulos, 1995) talks about using a language called TSQL2 in a natural language interface. The paper (Androutsopoulos, 1995) focuses on the TSQL2 in a natural-language interface for temporal databases and also in some point on the semantics of TSQL2.

(Reis and Mamede, 1997) present the Edite system, which is a natural-language interface to databases, and explore the advantage of joining natural-language processing with the expressiveness of graphical interfaces. Edite, a natural-language front-end for relational databases, is multi-lingual (Portuguese, French, English, Spanish). It is capable of answering written questions related to tourism by transforming them into SQL queries. The answer can be a list of resources, text, images or graphics depending of the questions. At present, the database contains 53000 tourism resources, arranged on 253 distinct types, which corresponds to 209 tables.

(Stratica, 2002) talks about a natural language processor for querying Cindi, which is also an SQL-based system.

A reliable natural-language interfaces to household appliances, which is also, an SQL-based interface is described in (Yates and Etzioni, 2003).

(Popescu, Etzioni and Kautz 2003) introduces a theoretical framework, which is the foundation for the fully implemented Precise NLI and proved that Precise guarantees a map for each question to the corresponding SQL query, for a broad class of semantically-tractable natural-language questions.

5.4 Limitation of SQL-based approaches

SQL-based approaches have several limitations. For example, the Edite system (Resi and Mamede, 1997) described in the previous section has some disadvantages such as – regarding the linguistic coverage it only accepts questions, no imperative or declarative statements are allowed. Moreover, another limitation is the set of restrictions imposed on the design and conception of the database. Most of the SQL-based approaches are also unable to handle negation, modality and intensionality.

Chapter 6 AN OVERVIEW OF AN EXISTING IMPLEMENTATION OF A SET-THEORETIC VERSION OF A SUB-SET OF MONTAGUE SEMANTICS

6.1 Overview of existing approach

Direct implementation of Montague semantics is impractical. A set-theoretic version of a first-order subset of Montague's approach, developed by Frost and Launchbury (FL 1989), is discussed in this section. However a simple conversion of Montague's treatment of negation does not work in this set-theoretic semantics, and is discussed later.

The following are examples of the types of the objects denoted by words and phrases of some syntactic categories in the FL semantics. These types are explained further later on. The notation `x -> y` denotes the type of functions from type `x` to type `y`.

```
noun        :: {entity}
intransverb:: {entity}
propernoun  :: {entity} -> bool
determiner  :: {entity} -> {entity} -> bool
transverb   :: ({entity -> bool}) -> {entity}
```

The following are examples of denotations of some words in the FL semantics. These denotations are also explained in more detail later.

```
d_planet    = {"mars", "earth" ..}
d_spins     = {"earth", "mars", "phobos", "deimos"..}
d_mars      = λs "mars" member s
d_every     = λsλt s subset t
d_no        = λsλt (s intersect t)={}
```

The following example illustrates how the meaning of a simple statement is evaluated. Note that d_<<x>> denotes meaning of the expression x.

```
d_<<every planet spins>>
```

```
=> λ x λ y(x subset y) d_planet d_spin

=> λ y (d_planet subset y) d_spin

=> (d_planet subset d_spin)

=>  True
```

We now explain these examples further. To do this, we use the notation of the functional-programming language Miranda. This notation is easier to read than lambda notation and also provides an environment in which one can experiment with the definitions.

According to Montague, noun and verb phrases denote characteristic functions. In the FL semantics, nouns and verb phrases denote sets of entities. These sets can be represented by lists in Miranda, for example:

```
d_moon    = ["deimos", "phobos" ..
d_planet = ["mars", "earth" ..
d_spins   = ["mars", "earth", "phobos", "deimos" ..
```

In the FL semantics, proper nouns (names) are implemented as functions, which take a list as input, and which return the boolean value `True` if the list contains the entry related to the proper noun, and `False` otherwise. For example, assuming that the function `member` has been defined appropriately:

```
d_sol    s = member s "sol"
d_mars   s = member s "mars"
d_earth s = member s "earth"
```

Accordingly, d_`<<mars spins>> => True` owing to the fact that application of `d_mars` to `d_spins` returns the value `True` because `"mars"` is a member of the list denoted by `d_spins`.

Quantifiers are implemented as higher-order functions, which take a list as input and which return a function of type `list -> bool` as output. For example, assuming that the functions `subset` and `intersection` have been defined appropriately:

```
d_every s t =  subset s t
d_a     s t =  intersection s t ~= []
d_no    s t =  intersection s t = []
```

Accordingly, `<<every planet spins>>` ⇒ `True` owing to the fact that partial application of the higher-order function `d_every` to `d_planet` returns the function `f` such that `f t = subset ["mars","earth"] t`. Application of `f` to `d_spins` returns `True` because `["mars","earth"]` is a subset of `["mars","earth","phobos","deimos"]`

Frost's approach to transitive verbs is described in sub-section 4.2. When converted to set-theoretic form, a transitive verb is implemented as a function whose argument is a predicate on sets. When it is applied to a particular predicate, it returns a set of entities as result. An entity is in the result set if the predicate is true of the entity's image under the associated relation. For example, 2-place transitive verbs are defined as shown in the following example:

```
d_orbit p = [x | (x, image_x) ← collect orbit_rel; p image_x ]
            where
            orbit_rel =  [("luna","earth"),  ("phobos","mars")
                          ("deimos","mars"), ("earth","sol"),
                            etc.
```

This definition uses a programming construct called a list comprehension. The general form of a list comprehension is: `[body | qualifiers]` where each qualifier is either a generator, of the form `var ← exp` or a filter, which is a Boolean expression used to restrict the range of the variables introduced by the generators. The collect function used in the above definition converts a relation of tuples `<a,b>` to a relation of tuples `<a,c>` where c is the image of a under the original relation. For example, applying collect to the relation `orbit_rel` above returns the following:

```
collect orbit_rel ⇒ [("luna",    ["earth"]),
                     ("phobos",  ["mars"]),
                     ("deimos",  ["mars"]),
                     ("earth",   ["sol"]),
                     ("mars",    ["sol"])]
```

Therefore, the meaning of the phrase "orbits mars" is obtained by applying `d_orbits` to `d_mars`:

```
d_orbits d_mars ⇒ [x|(x, image_x) ← collect orbit_rel; d_mars image_x]
                ⇒ [x|(x, image_x) ← [("luna",   ["earth"]),
                                     ("phobos", ["mars"]),
                                     ("deimos", ["mars"]),
                                     ("earth",   ["sol"]),
                                     ("mars",    ["sol"])];
                                      d_mars image_x]
                ⇒ ["phobos", "deimos"]
```

Passive verb phrases can be interpreted by simply inverting the order of the columns of the binary relation, which is used in the definition of the transitive verb.

The resulting semantics has been used in the implementation of a natural-language query processor, which is constructed as an executable specification of an attribute grammar. The processor is accessible at the following URL:

http://www.cs.uwindsor.ca/users/r/richard/miranda/wage_demo.html

6.2 An overview of an extension of the set-theoretic approach to accommodate negation

A problem with the FL semantics is that the denotation of the word "no" only works in some syntactic contexts, and fails in others, as illustrated below. For example the denotation of "sol orbits no moon" is:

```
⇒ <<sol orbits no moon >>
⇒ <<sol>> ( <<orbits>> ( <<no>> <<moon>>)
⇒ d_sol (d_orbits (d_no d_moon))
⇒ d_sol [ x | (x , image_x) <- collect orbut_rel;
        (d_no d_moon) image_x]
⇒ d_sol [ x | (x, image_x) <- collect orbit_rel;
        (intersection d_moon image_x) = []]
⇒ d_sol [x | (x, image_x) <- collect orbit_rel;
        (intersection ["deimos", "phobos"] image_x ) = [] ]
⇒ d_sol [ "deimos", "phobos", "mars" , "earth" ]
⇒ member [ "deimos" , "phobos", "mars", "earth"] "sol
⇒ False
```

Which is not the expected answer. The reason for the failure is that when collect is applied to orbit_rel it generates the following relation:

```
[("deimos", ["mars']), ("phobos", ["mars"]),
 ("mars",   ["sol"]),  ("earth",  ["sol"])]
```

Owing to the fact that the images of "deimos", "phobos", "earth" and "mars" have empty intersections with list `['deimos","phobos"]`, the meaning of the sub-expression "orbit no moon" is computed to be: `["deimos", "phobos", "mars", "earth"]`. This list does not include “sol”, and consequently, the evaluation of <<sol orbits no moon>> returns the incorrect result False.

Frost and Boulos(2002) have developed a method for accommodating negation which is based on the notion that a set can be represented in two ways: explicitly by enumerating all of its members, or implicitly by enumerating all of the members of its complement. In cases where a set is computed as the denotation of a phrase that involves a negation, it is represented using its complement.

To implement this approach, a new type set is introduced, which can be defined in Miranda as follows, where `[string]` is the type list of strings and `string` is a synonym for the type list of characters:

```
set ::= SET [string] | COMP [string]
```

The following are two examples of objects of type set. The first example represents the set whose members are ‘‘phobos’’ and ‘‘deimos’’. The second example denotes the set of all entities in the universe of discourse except ‘‘phobos’’ and ‘‘deimos’’, i.e. the set of “non moons”.

```
SET [''phobos'',''deimos''] COMP[''phobos'', ''deimos'']
```

To determine the cardinality of a set we define the function cardinality in terms of the cardinality of the set of all entities in the universe of discourse, where # computes the length of a list, and all entities denotes the set of all entities in the universe of discourse.

```
cardinality (SET s)  = #s
cardinality (COMP s) = #all_entities - (#s)
```

Operators on sets are redefined as follows:

```
c_member       (SET s)   e          = member s e
c_member       (COMP s)  e          =~(member s e)
c_union        (SET s)  (SET t)     = SET (union s t)
c_union        (SET s)  (COMP t)    = COMP (t -- s)
c_union        (COMP s) (SET t)     = COMP (s -- t)
c_union        (COMP s) (COMP t)    = COMP (intersection s t)
c_intersection (SET s)  (SET t)     = SET (intersection s t)
c_intersection (SET s)  (COMP t)    = SET (s -- t)
c_intersection (COMP s) (SET t)     = SET (t -- s)
c_intersection (COMP s) (COMP t)  = COMP (union s t)
c_subset       (SET s)  (SET t)   = subset s t
c_subset       (SET s)  (COMP t)  = (t -- s) = t
c_subset       (COMP s) (SET t)   = (#(union t s) = #all_entities)
c_subset       (COMP s) (COMP t)  = subset t s
```

Evaluation of the denotation of the phrase "non moon that spins" would result in the following operation:

```
c_intersection COMP [''phobos'', ''deimos'']
               SET [''mars'', ''earth'', ''phobos'', ''deimos'']

⇒ SET [''mars'', ''earth'']
```

Redefinition of nouns and quantifiers is straightforward:

```
d_moon       = SET [''deimos'', ''phobos'']
d_planet     = SET [''earth'',''mars'']
d_spins      = SET [''earth'',''deimos'',''mars'',''phobos'']
d_thing      = COMP []
d_sol   s    = c_member s ''sol''
d_mars  s    = c_member s ''mars''
d_every s t  = c_subset s t
d_a     s t  = cardinality (c_intersection s t) > 0
d_no    s t  = cardinality (c_intersection s t) = 0
```

The denotation of each transitive verb is redefined. In order to simplify the coding of denotations of transitive verbs, the common parts of such definitions can be abstracted into a higher-order function make_denotation_of_tv defined as follows:

```
make_denotation_of_tv r p
   = COMP (firsts_of r -- result), if p (SET []) = True
   = SET result, otherwise
      where
         result = [x | (x,image_x) <- collect r; p image_x]
  firsts_of []              = []
  firsts_of ((x,y):rest)    = x : firsts_of [(a,b)|(a,b) <- rest; a~=x]
```

This function can now be used to define the denotations of various transitive verbs. For example:

```
  d_orbits     = make_denotation_of_tv orbit_rel
  d_discovered = make_denotation_of_tv discover_rel
  discover_rel = [(''hall'',''phobos''),  (''hall'',''deimos'')
                  (''kuiper'',''uranus''),(''galileo'',''europa'')]
  orbit_rel    = [(''luna'',''earth''),   (''phobos'',''mars'')
                  (''deimos'',''mars''),  (''earth'',''sol'')
                  (''mars'',''sol'')]
```

In the revised semantics, the denotation of "orbits no planet" is:

```
<<orbits no planet>>
   ⇒ <<orbits>> (<<no>> <<planet>>)
   ⇒ d_orbits (d_no d_planet)
   ⇒ COMP (firsts_of orbit_rel -- result)
      where result = [x|(x,image_x) <- collect orbit_rel;
                     (d_no d_planet image_x)]
   ⇒ COMP (firsts_of orbit_rel -- result)
      where result = [x|(x,image_x) <- collect orbit_rel;
                (c_intersection [''earth'',''mars''] image_x) = []]
   ⇒ COMP ([''deimos'', ''phobos'', ''mars'', ''earth''] --
            [''earth'', ''mars''])
   ⇒ COMP [''phobos'', ''deimos'']
```

Meaning that everything except phobos and deimos "orbits no moon". Evaluation of "sol orbits no planet" now returns the expected answer:

```
<<sol>> <<orbits no planet>>
     ⇒ d_sol (COMP [''phobos'', ''deimos'']) from above
     ⇒ member (COMP [''phobos'', ''deimos'']) ''sol''
     ⇒ True
```

Chapter 7 THE PROBLEM AND INITIAL APPROACHES

7.1 The Problem: To extend set-theoretic approach to accommodate n-ary transitive verbs where n > 2

The approach to natural-language database-query processing developed by Frost, Launchbury (FL) is very limited in its scope. It does not accommodate modality, intensionality, or transitive verbs with arity greater than two. Work on the first two of these limitations is being undertaken but is a long-term project. The third limitation is the subject of this thesis.

In this report we claim that it is possible to extend the set-theoretical compositional semantics developed by Frost et al to accommodate n-ary transitive verbs (n $\geq$2) by re-defining all denotations to involve sets of attributes rather than simple entities.

Below we describe two initial approaches to extend the set-theoretic approach to accommodate n-ary transitive verbs, n $\geq$ 2, neither of which were entirely satisfactory. The final approach is described in chapter 8.

We describe some details of the approaches using the Miranda programming notation, which provides concise, formal and testable definitions.

7.2 Approach 1

Below we discuss one of the possible strategies, which we considered but found not to be viable.

In this approach, we tried to handle time aspect (discrete time points, not intervals) without modifying the set-theoretic version of Montague semantics as discussed in chapter 5. We tried to leave all the definitions of the "binary" transitive verbs as they are and projected binary relations (e.g. discover) from the n-ary relations (e.g. discover_time) so that the existing semantics still works. For example,

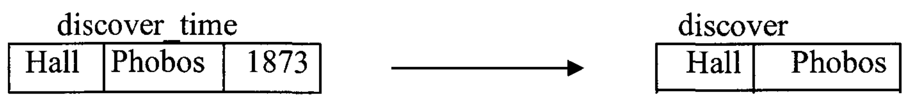

Figure 6.1 : Approach 1

In this approach, we use the “discover_time” relation when we need to deal with time. So in this case we will need two definitions of the “discover” semantic function one to handle questions with non-time and another to handle questions that involves time. For example,

non_time: `did hall [discover phobos ]` ^{1}

↓

using the discover function as currently defined.

time: `when did hall [discover phobos]` ^{2}

↓

a new discover function

The problem in this approach is that we not only need to have two definitions for “discover” but also two definitions for “phobos” “hall” etc. Also the one to one correspondence is lost as [discover phobos] ^{1} has same syntax as [discover phobos] ^{2} but the first one returns a set of entities and the second one returns a set of tuples so the return type is not same. For example,

```
non_time      << discover phobos >>  => [hall]
with_time     << discover phobos >>  => [(hall, 1870)]
```

Therefore, we need to change (“lift”) all instances of “discover’ to the time version and change other denotations that are required. This approach loses the syntactic category/semantic type correspondence and therefore loses compositionality.

7.3 Approach 2

In approach 2, we present semantics, which accommodate 3-place transitive verbs by handling the time aspect (again discrete time points, not intervals). This will allow us to accommodate phrases such as “discovered phobos in 1873”.

Therefore, the definitions of all denotations of words have to be little more complex than before. For example the definition of “Hall” has to be a little more complex. In the previous semantics <<“Hall”>> denotes a set of properties that are true of the entity$_{Hall}$. Now, in the extended semantics <<”Hall”>> will denote a set of properties with time stamps. By different time stamps we mean different conditions. For example

Time 0 means forever

Time –100 means True

And [ ] means False

In this approach, transitive verbs are higher-order functions, which return sets of attribute values as results rather than just truth values.

The semantics for 3-place transitive verbs in presented below:

Definitions:

In the modified semantics, nouns and verb phrases denote sets of tuples with the entities and time stamps in it. These sets can be represented by lists in Miranda, for example:

```
planet  = [ ENTTIME e_mars t0, ENTTIME e_uranus t0, ENTTIME e_earth t0]

spin    = [ENTTIME e_deimos t0,  ENTTIME e_phobos t0,
           ENTTIME e_Uranus t0,  ENTTIME e_europa t0,
           ENTTIME e_mars t0,    ENTTIME e_earth t0 ]

   where entity         ::= NAME [char]
         time           ::= TIME num
         entity_time    ::= ENTTIME entity time
         e_mars           = NAME "mars"
         t0               = TIME 0
```

In the modified semantics, proper nouns (names) are implemented as functions, which take a list as input, and which return list with time stamp in it if the list contains the entry related to the proper noun, and the empty list otherwise. For example:

```
  mars ents    = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "mars"]
```

Accordingly, `d_<<earth spins>>` ⇒ `[ENTTIME (NAME "earth") (TIME 0)]` here by `0` we mean spins forever.

Quantifiers are implemented as higher-order functions, which are defined as follows:

```
   every s t  = s, if subset s t
              = [], otherwise

   a s t      = [ENTTIME y ttt | (ENTTIME y tt) <- s;
                                 (ENTTIME z ttt) <- t; y = z]

   no s t = [], if res ~= []
          = [ENTTIME (NAME "true") (TIME (-100))], otherwise
                       where res = a s t
```

Accordingly, `d_<<every planet spins>>`⇒ `[ENTTIME (NAME "mars")  (TIME 0),`
```
                                  ENTTIME (NAME "uranus")(TIME 0),
                                  ENTTIME (NAME "earth") (TIME 0)]
```

which lists all the planets, `TIME 0` means that they have been spinning forever.

The words “and” and “or” are also implemented as higher-order functions, which are defined as follows:

```
term_or g h = r
          where r s = (g s) ++ (h s), if g s ~= []  \/ h s ~= []
                    = [], otherwise
term_and p q = r
          where r s = [], if p s = []  \/ q s = []
                    = (p s) ++ (q s) , otherwise
```

In the modified semantics transitive verbs are implemented as follows:

```
make_transitive_verb rel p =
                     mkset [ENTTIME x t|   (x, s) <- collect rel;
                           (ENTTIME y t)<- (p s);  p s  ~= []]

collect []              = []
collect ((EET x y z):t) = (x, (ENTTIME y z):
                          [ENTTIME b c |(EET a b c) <- t; a = x]):
                         collect[EET l m n|(EET l m n)<- t;l ~= x]

discover_rel = [ EET (NAME "hall")    (NAME "phobos") (TIME 1873),
                 EET (NAME "galileo") (NAME "europa") (TIME 1820),
                 EET (NAME "kuiper")  (NAME "uranus") (TIME 1860),
                 EET (NAME "hall")    (NAME "deimos") (TIME 1875)]

orbit_rel = [ EET e_deimos e_mars t0,  EET e_phobos e_mars t0,
              EET e_mars   e_sol  t0,  EET e_earth  e_sol  t0]
```

For example, in the new modified semantics evaluation of “`discover europa`” returns `[(Galileo, 1820)]`, which is the name of the discoverer and the time. Also evaluation of “`orbits mars`” now returns `[(deimos, 0), (phobos, 0)]` instead of just `[deimos, phobos]` as in previous semantics.

By applying the new collect function to the relation orbit_rel, the following is obtained:

```
collect orbit_rel = [(NAME "deimos",[ENTTIME (NAME "mars")(TIME 0)]),
                     (NAME "phobos",[ENTTIME (NAME "mars")(TIME 0)]),
                     (NAME "mars",  [ENTTIME (NAME "sol") (TIME 0)]),
                     (NAME "earth", [ENTTIME (NAME "sol"  (TIME 0)])]
```

So, the final result will be as follows:

```
orbit mars = mkset[ENTTIME x t| (x, s) <-
                 [(NAME "deimos", [ENTTIME (NAME "mars")(TIME 0)]),
                  (NAME" phobos", [ENTTIME (NAME "mars")(TIME 0)]),
                  (NAME "mars",   [ENTTIME (NAME "sol") (TIME 0)]),
                   (NAME "earth",   [ENTTIME (NAME "sol") (TIME 0)])];
                  (ENTTIME y t)<- (mars s); mars s  ~= []]
           =  [ENTTIME (NAME "deimos") (TIME 0),
               ENTTIME (NAME "phobos") (TIME 0)]
```

Similarly, `<<discovered_by Hall>>` will now return

```
                       [ENTTIME (NAME "phobos") (TIME 1873),
                        ENTTIME (NAME "deimos") (TIME 1875)].
```

Example queries:

Below are some example queries, which a Miranda program implementing approach 2 can handle.

```
Q: mars spins
A: [ENTTIME (NAME "mars") (TIME 0)]
```

The statement "mars spins" returns the entity mars and time 0 which means mars has been spinning forever.

```
Q: mars (orbits sol)
A: [ENTTIME (NAME "mars") (TIME 0)]
```

Similarly, the statement "mars orbits sol" returns the entity mars and time 0 which means mars has been orbiting sol forever. Here the entity mars is returned as answer, which is redundant information, but we return the entity and time in order to keep all the return types same of all same structure.

```
Q: (mars $term_and phobos ) spin
A: [ENTTIME (NAME "mars") (TIME 0),ENTTIME (NAME "phobos") (TIME 0)]
```

In the same way the query "mars and phobos spins" returns the entity mars and time 0 and entity phobos and time 0 which means they both have been spinning forever.

```
Q: hall (discovered (phobos $term_and deimos))
```

```
A:  [ENTTIME  (NAME  "hall")  (TIME  1873),ENTTIME  (NAME  "hall")  (TIME
1875)]
```

The statement "hall discovered phobos and deimos" returns the entity hall and time 1873 and the entity hall again and the time 1875 which means hall discovered phobos in 1873 and deimos in 1875. The answer is a little ambiguous, as it doesn't state which time is for phobos and which time is for deimos, but the order is preserved.

```
Q: hall (discovered (phobos $term_and europa))
A: []
```

The statement "hall discovered phobos and europa" returns the empty list, which means that the statement is false, so hall didn't discover both phobos and europa. We use the empty list to denote false.

```
Q: hall (discovered (phobos $term_or europa))
A: [ENTTIME (NAME "hall") (TIME 1873)]
```

The statement "hall discovered phobos or europa" returns the entity hall and time 1873 which means hall discovered one of the moons above which is hall discovered phobos in 1873. The answer is ambiguous, as it doesn't state if the time is for phobos or europa. We need to do other queries to find out the right answer.

```
Q: hall (discovered (no moon)
A: []
```

The statement "hall discovered no moon" returns empty list, which means that the statement is false.

```
Q: no man (discovered (no moon))
A: []
```

The query “no man discovered no moon” returns the empty list, which means that the statement is false.

```
Q: no planet (discovered (no moon))
A: [ENTTIME (NAME "true") (TIME (-100))]
```

The query “no planet discovered no moon” returns a strange result. However, it is not the empty list so it indicates that the statement is True.

```
Q: when did (hall (discover phobos))
A: [ENTTIME (NAME "hall") (TIME 1873)]
```

The query “when did hall discover phobos?” returns the entity hall and the time 1873 which means that hall discovered phobos in 1873.

Putting all the pieces together (approach 2):

Below we present the full program that handles 3-place transitive verbs for a small sub-set of English. The program can answer various questions about the solar system. Please note that the objective of our research is not to create a Miranda program for a particular set of queries but to develop a compositional semantics. We use Miranda only to formalize the definitions and illustrate an example of their use.

```
entity   ::= NAME [char]
e_mars     = NAME "mars"
e_deimos   = NAME "deimos"
e_hall     = NAME "hall"
e_phobos   = NAME "phobos"
e_galileo  = NAME "galileo"
e_europa   = NAME "europa"
e_kuiper   = NAME "kuiper"
e_uranus   = NAME "uranus"
e_sol      = NAME "sol"
e_earth    = NAME "earth"

time ::= TIME num
t0     = TIME 0
t1     = TIME 50

entity_time ::= ENTTIME entity time

planet  = [ENTTIME e_mars t0, ENTTIME e_uranus t0, ENTTIME e_earth t0]
planets = planet
moon    = [ENTTIME e_deimos t0,
           ENTTIME e_phobos t0,
           ENTTIME e_europa t0]
moons = moon
spin  = [ENTTIME e_deimos t0, ENTTIME e_phobos t0,
         ENTTIME e_uranus t0, ENTTIME e_europa t0,
         ENTTIME e_mars t0, ENTTIME e_earth t0 ]
spins = spin
man   = [ENTTIME e_kuiper t0, ENTTIME e_hall t0, ENTTIME e_galileo t0]
men   = man

mars ents    = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "mars"]
hall ents    = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "hall"]
phobos ents  = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "phobos"]
galileo ents = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "galileo"]
europa ents  = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "europa"]
kuiper ents  = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "kuiper"]
deimos ents  = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "deimos"]
uranus ents  = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "uranus"]
sol ents     = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "sol"]
earth ents   = [ENTTIME s t | ENTTIME s t <- ents; s = NAME "earth"]

set_ent_time ::= SET_ENT_TIME [entity_time]

every s t = s, if subset s t
          = [], otherwise

a s t = [ENTTIME y ttt | (ENTTIME y tt) <- s;(ENTTIME z ttt) <- t; y=z]

no s t = [], if res ~= []
       = [ENTTIME (NAME "true") (TIME (-100))], otherwise
         where res = a s t

intersect = a
union s t = s ++ ( t -- s)
subset x y = (x -- y) = []

noun_and s t = [ENTTIME y tt | (ENTTIME y tt) <- s] ++ [ ENTTIME z ttt
| (ENTTIME z ttt) <- t]

noun_or s t = union s t

verb_and s t = f
               where f ents = s ents ++ t ents

term_or g h = r
              where r s = (g s) ++ (h s), if g s ~= []  \/ h s ~= []
                        = [], otherwise

term_and p q = r
               where r s = [], if p s = []  \/ q s = []
                         = (p s) ++ (q s) , otherwise


make_transitive_verb rel p = mkset [ENTTIME x t| (x, s) <- collect rel;
                                    (ENTTIME y t)<- (p s); p s  ~= []]

collect []              = []
collect ((EET x y z):t) = (x, (ENTTIME y z):[ENTTIME b c |
                          (EET a b c) <- t; a = x]): collect
                          [EET l m n | (EET l m n) <- t; l ~= x]

entity_entity_time ::= EET entity entity time
entity_setentity_time ::= E_setET entity [entity_time]  time

discover_rel = [ EET (NAME "hall")    (NAME "phobos") (TIME 1873),
                 EET (NAME "galileo") (NAME "europa") (TIME 1820),
                 EET (NAME "kuiper")  (NAME "uranus") (TIME 1860),
                 EET (NAME "hall")    (NAME "deimos") (TIME 1875)]

orbit_rel = [ EET e_deimos e_mars t0, EET e_phobos e_mars t0,
              EET e_deimos e_sol t0,  EET e_phobos e_sol t0,
              EET e_mars e_sol t0,    EET e_earth e_sol t0]

discovered        = discover
discover          = make_transitive_verb discover_rel
is_discovered_by = make_transitive_verb (invert discover_rel)

orbit          = make_transitive_verb orbit_rel
orbits         = orbit
is_orbited_by = make_transitive_verb (invert orbit_rel)

invert rel = [ EET y x z | EET x y z <- rel]

remove_dup [ ]                 = [ ]
remove_dup ((ENTTIME x y): es) = (ENTTIME x y): remove_dup es,
                                             if ~member1 x es
                               = remove_dup es, otherwise

member1 p [] = False
member1 p ((ENTTIME x y):es) = True \/ member1 p es, if p=x
                             = False \/ member1 p es, otherwise

how_many s t = #(remove_dup (a s t ))
which = a
what x = x
does [TIME 0] = [TIME 100]
does [] = []
is = does
when f res = res
did res = res ~= [ ]
who x = x
```

7.4 Critical analysis of Approach 2

We can see from the example queries of approach 2 that some of the answers returned by the Miranda program are ambiguous. Some times we need to ask more questions to resolve this ambiguity. Investigation also showed that if we want to extend this approach to accommodate 4-place and 5-place transitive verbs, adding more information to different relations (e.g. discover_rel, orbit_rel) is not sufficient. We also need to change the definition of all the denotations of words (e.g. hall, phobos, deimos etc). Hence, the approach has lost "extensibility" as the old definitions require substantial changes, to accommodate extra arguments for verbs. Also loss of orthogonality as different definitions are required for n=3, n=4 n=5 verbs etc. To overcome this problem we developed another approach, which is described in the next chapter.

Chapter 8 THE FINAL APPROACH

8.1 The approach

In this approach transitive verbs are higher-order functions, which return sets of attributes values (things) as results rather than just truth-values as in the FL approach. In the new modified semantics `[discover phobos]` will return all information that is available in the relation e.g.

```
d_<< "discover phobos" >> => [[Person    "Hall",
                                Moon      "Phobos",
                                TIME       1870,
                                Implement "with a telescope"…etc]
```

And question like "Did hall discovered phobos" should return yes or no plus some other information like when and with what etc. Phrases such as "who" "when" "how_many", "with _what", etc will filter out all the unnecessary information.

The following are examples the types of the objects denoted by words and phrases of some syntactic categories in the modified semantics.

```
noun        :: [[things]]
intransverb:: [[things]]
propernoun  :: [[things]] -> [[things]]
determiner  :: [[things]] -> [[things]] -> [[things]]
transverb   :: ([[things]] -> [[things]]) -> [[things]]
```

where `things` denote sets of attributes values.

Below we show how an illustrative example query processor based on the new semantics can be implemented in Miranda.

8.2 Example implementation 1: A solar-system database

In the modified semantics nouns and verb phrases denote sets of sets with the entities in it. These sets can be represented by lists of lists in Miranda, for example:

```
entity == [char]
time   == num

         things   ::= Person       entity
                    | Implement    entity
                    | Moon         entity
                    | Time         time
                    | Planet       entity
                    | Sun          entity
                    | Color        entity

planet = [[Planet "mars"], [Planet "uranus"], [Planet "earth"]]

moon   = [[Moon "phobos"], [Moon "deimos"], [Moon "europa"]]
```

In the modified semantics, proper nouns (names) are implemented as functions, which take a list of lists as input, and which return a list of lists if the list contains the entry related to the proper noun, and empty list otherwise. For example:

```
mars l = [ s | s <- l; member s (Planet "mars")]
```

Accordingly, d_`<<mars spins>> => [[Planet "mars"]]`, which indicates that the planet mars spins, as the statement is true.

Quantifiers are implemented as higher-order functions, which are defined as follows:

```
every  s t   = s, if subset s (remove t)
             = [], otherwise

remove  t    = [ [a] | a:aa <- t]

subset x  y  = True , if (x -- y) = []
             = False, otherwise

a s t        = [l:mm | l:ll <- s; m:mm <- t; l = m]

no s t       = [], if res ~= []
             = [[Value "true"]], otherwise
                  where res = a s t
```

Accordingly, `d_<<every moon spins>> => [[Moon "phobos"],`
```
                                        [Moon "deimos"],
                                        [Moon "europa"]]
```
which is non-empty indicating True.

The words "and" and "or" are also implemented as higher-order functions, which are defined as follows:

```
   term_or g h = r
             where r s = (g s) ++ (h s), if g s ~= []  \/ h s ~= []
                       = [], otherwise

  noun_and s t = union s t

  union s t = s ++ ( t -- s)

  verb_and s t = f
               where f ents = s ents ++ t ents

  term_and p q = r
               where r s = [], if p s = []  \/ q s =
                         = (p s) ++ (q s) , otherwise
```

The denotation of the phrase `(term_and p q)` is a function `r` which takes a list of entities s as input and which return a list of entities by appending the values `(p s )` and `(q s)`.

From example,

`(phobos $term_and deimos) spin ⇒ [[Moon "phobos"],[Moon "deimos"]]`

In the modified semantics transitive verbs are implemented as follows:

```
 make_transitive_verb rel p = mkset [x : t| (x, s) <- collect rel;
                                     t <-(p s); p s  ~= []]

 collect []            = []
 collect ((x:t):r)     = (x,t:[s | (a:s)<- r; a = x]): collect
                         [l:f | (l:f) <- r; l ~= x]

 discover_rel = [[(Person    "hall"),
                  (Moon      "phobos"),
                  (Time       1873),
                  (Implement "with telescope")],
                 [(Person    "hall"),
                  (Moon      "deimos"),
                  (Time       1875),
                  (Implement "with telescope")],
                 [(Person    "kuiper"),
                     (Moon      "uranus"),
                     (Time       1860),
                     (Implement "with telescope")],
                    [(Person    "galileo"),
                     (Moon      "europa"),
                     (Time       1820),
                     (Implement "with telescope")]]


  orbit_rel = [[(Moon "deimos"),  (Planet "mars")],
               [(Moon "phobos"),  (Planet "mars")],
               [(Planet "mars"),  (Sun "sol")],
                [(Planet "earth"), (Sun "sol")]]

discovered  = discover
discover    = make_transitive_verb discover_rel
orbit       = make_transitive_verb orbit_rel
orbits      = orbit
```

For example, in the new modified semantics evaluation of `<<discover europa>>` returns

```
[[Person    "galileo",
  Moon      "europa",
  Time       1820,
  Implement "with telescope"]].
```

Also evaluation of "orbits mars" now return

```
              [[Moon "deimos", Planet "mars"],
               [Moon "phobos", Planet "mars"]]
```

By applying the new collect function to the relation orbit_rel, the following is obtained:

```
   collect orbit_rel = [(Moon   "deimos", [[Planet "mars"]]),
                        (Moon   "phobos", [[Planet "mars"]]),
                        (Planet "mars",   [[Sun    "sol"]]),
                        (Planet "earth",  [[Sun    "sol"]])]
```

So, the final result will be as follows:

```
  orbit mars = mkset [x : t| (x, s) <-
                     [(Moon   "deimos", [[Planet "mars"]]),
                      (Moon   "phobos", [[Planet "mars"]]),
                      (Planet "mars",   [[Sun    "sol"]]),
                      (Planet "earth",  [[Sun    "sol"]])];
                       t <-(mars s); mars s  ~= []]


              =[[Moon "deimos", Planet "mars"],
                [Moon "phobos", Planet "mars"]]
```

Similarly, verb `discover` will be treated as follows:

```
 discovered p = mkset [x : t| (x, s) <- collect discover_rel;
```

```
                                   t <-(p s); p s~= []]
```

```
    collect discover_rel                                   discover_rel
 Hall     [[Phobos, 1873, Telescope]                  Hall     Phobos  1873 Telescope
          [Demios, 1875, Telescope]]    collect       Hall     Deimos  1875 Telescope
 Kuiper   [[Uranus, 1860, Telescope]]     <--         Kupier   Uranus  1860 Telescope
 Galileo  [[Europa, 1820, Telescope]]                 Galileo  Europa 1820 Telescope
```

```
discovered phobos ⇒
 mkset [x : t| (x, s) <-  collect
  [(Person "Hall",    [[Moon "phobos",Time 1873,Implement "with telescope"],
                       [Moon "deimos",Time 1875,Implement "with telescope"]]),
   (Person "kuiper", [[Moon "uranus",Time 1860,Implement "with telescope"]]),
   (Person "galileo",[[Moon "europa",Time 1820,Implement "with telescope"]])];
                                                t <-(Phobos s); Phobos s~= []]

discovered phobos ⇒ [[Person    "hall",
                      Moon      "phobos",
                      Time       1873,
                      Implement "with telescope"]]
```

Similarly, `<<discovered_by Hall>>` will now return

```
[[ Moon      "phobos",
   Person    "hall",
   Time       1873,
   Implement "with telescope"],
 [ Moon      "deimos",
   Person    "hall",
   Time       1875,
   Implement "with telescope"]]
```

And phrases like `<<when did hall discover phobos>>` will filter out necessary information. For example:

```
 <<hall discovered phobos>>  => [[ Person      "hall",
                                   Moon        "phobos",
                                   Time         1873,
                                   Implements  "with telescope"]]
```

And `<<when did hall discover phobos>> ⇒ [Time 1873]`

The complete program listing is given in the appendix.

8.3 Example implementation 2: A database about books and authors

Now, we can use the definitions of every, a, no, term_and, term_or, transitive verbs etc in a new application.

In this semantics nouns phrases denote set of sets with the entities in it.

```
book  = [[Book "Hamlet"], [Book "Merchant of Venice"],
        [Book "Rage of Angels"],[Book "If Tomorrow Comes"]]


where entity == [char]
      time == num
things  ::= Name       entity
          | Book       entity
          | Place      entity
          | Time       time
          | Value      entity
```

In this semantics, proper nouns (names) are implemented as functions, which take a list of lists as input, and which return a list of lists if the list contains the entry related to the proper noun, and empty list otherwise. For example

```
shakespeare l  = [ (a:as) | (a:as) <- l; a = (Name "shakespeare")]
```

In this semantics transitive verbs are implemented as the above application. So, the same definition of transitive verb of our approach can be used for the verbs like “write”.

By applying new collect to the relation written_rel, the following is obtained:

```
written_rel = [[(Name "shakespeare"),
                (Book "Hamlet"),
                (Time 1573),
                (Place "England")],
               [(Name "shakespeare"),
                (Book "Merchant of Venice"),
                (Time 1575),
                (Place "England")],
               [(Name "sidney"),
                (Book "Rage of Angels"),
                (Time 1950),
                 (Place "USA")],
                [(Name "sidney"),
                 (Book "If Tomorrow Comes"),
                 (Time 1940),
                 (Place "USA")]]


collect written_rel =
[(Name "shakespeare",[[Book "Hamlet",            Time 1573,Place "England"],
                      [Book "Merchant of Venice",Time 1575,Place "England"]]),
     (Name "sidney", [[Book "Rage of Angels",    Time 1950,Place "USA"],
                      [Book "If Tomorrow Comes", Time 1940,Place "USA"]])]
```

So, the final result will be as follows:

```
wrote hamlet = mkset [x : t| (x, s) <-
[(Name "shakespeare",[[Book "Hamlet",             Time 1573,Place "England"],
                      [Book "Merchant of Venice", Time 1575,Place"England"]]),
(Name "sidney",     [[Book "Rage of Angels",      Time 1950,Place "USA"],
                     [Book "If Tomorrow Comes",   Time 1940,Place "USA"]])];
                                          t <-(hamlet s); hamlet s  ~= []]

            = [[Name  "shakespeare",
               Book   "Hamlet",
               Time    1573,
               Place "England"]]
```

Similarly, `<<was_written_by Shakespeare>>` return

```
[[Book "Hamlet",           Name "shakespeare", Time 1573,Place "England"],
 [Book "Merchant of Venice",Name "shakespeare",Time 1575,Place "England"]]
```

which is everything that Shakespeare wrote.

And phrases like `<<when Shakespeare wrote hamlet>>,<<where did Shakespeare write hamlet>> etc` will filter out necessary information.

For example:

```
<<Shakespeare wrote hamlet>> => [[Name" shakespeare", Book "Hamlet",
                                  Time 1573,Place "England"]]
```

And `<< when Shakespeare wrote hamlet >>      => [Time 1573]`

```
    << where did Shakespeare wrote hamlet>>  => [Place "England"]
```

The complete program listing is given in the appendix.

Chapter 9 EVALUATION OF THE FINAL APPROACH

9.1 Overview

Below we again specify the objectives for our semantics as discussed in as in chapter 2:

a) We will state examples of types of questions that our semantics will be able to handle, give small grammars for them, and compute the sizes of the example languages.

b) The new semantics will maintain the orthogonality of the old semantics- i.e. that the meaning of all (disambiguated) words is independent of context, and that the rules of composition are also independent of context.

c) The new semantics will maintain the syntactic/semantic correspondence i.e. phrases of the same syntactic category denote functions of the same semantic type.

And the thesis statement is

> "It is possible to extend the set-theoretical compositional semantics developed by Frost et al to accommodate n-ary transitive verbs, ($n \geq 2$) by re-defining all denotations to involve sets of attributes rather than simple entities, without loss of compositionality. "

We now discuss how our approach meets these objectives and proves the thesis.

9.2 A Grammar for example query processor #1

Below we present a small grammar based on our semantics, which can answer various questions about our solar system:

A small grammar (recursive)::

```
query  ::=             term_phrase   verb_phrase
         | who         term_phrase   verb_phrases
         | when        term_phrase   verb_phrases
         | with_what   term_phrase   verb_phrases
         | which       term_phrase   verb_phrase
         | how_many    term_phrase   verb_phrase
```

```
verb_phrase ::= transitive_verb_phrase
              | intransitive_verb_phrase

transitive_verb_phrase ::=transitive_verb  join_term_phrase

intransitive_verb_phrase ::= intransitive_verb
                           | intransitive_verbs  join_verb intransitive_verb

join_term_phrase ::= term_phrase
                   |  term_phrase term_join join_term_phrase

term_phrase ::= proper_nouns
              | det_phrase

det_phrase ::= determiner noun_phrase

noun_phrase ::= noun
              | noun noun_join noun_phrase
              | adjective noun

adjective ::= red

determiner ::= a | no | every

term_join ::= and | or

verb_join ::= and | or

noun_join::= and | or

transitive_verb ::= discover | orbit

intranstive_verb ::= spin

noun ::= moon | planet

proper_noun ::= Hall | Galileo | Phobos | Deimos
```

The above grammar is recursive, so the size of the language is infinitive and can't be calculated. Below we present a small grammar of depth recursion 2 and calculate its size.

Non-Recursive:

Query^{1555920} ::= term_phrase^{40} verb_phrase^{6483}
| who^{1} term_phrase^{40} verb_phrases^{6483}
| when^{1} term_phrase^{40} verb_phrases^{6483}
| with_what^{1} term_phrase^{40} verb_phrases^{6483}
| which^{1} term_phrase^{40} verb_phrase^{6483}
| how_many^{1} term_phrase^{40} verb_phrase^{6483}

verb_phrase^{6483} ::= transitive_verb_phrase^{6480}
| intransitive_verb_phrase^{3}

transitive_verb_phrase^{6480} ::= transitive_verb^{2} term_phrase^{40}
| transitive_verb^{2} term_phrase^{40} term_join^{2} term_phrase^{40}

intransitive_verb_phrase^{3} ::= intransitive_verb^{1}
| intransitive_verb^{1} join_verb^{2} intransitive_verb^{1}

term_phrase^{40} ::= proper_nouns^{6}
| det_phrase^{34}

det_phrase^{34} ::= noun_phrase^{34}

noun_phrase^{34} ::= noun^{2}
| noun^{2} noun_join^{2} noun^{2}
| noun^{2} noun_join^{2} adjective^{1} noun^{2}
| adjective^{1} noun^{2} noun_join^{2} noun^{2}
| adjective^{1} noun^{2} noun_join^{2} adjective^{1} noun^{2}

adjective^{1} ::= red^{1}

determiner^{3} ::= a^{1} | no^{1} | every^{1}

term_join^{2} ::= and^{1} | or^{1}

verb_join^{2} ::= and^{1}| or^{1}

noun_join^{2}::= and^{1} | or^{1}

transitive_verb^{2} ::= discover^{1} | orbit^{1}

intranstive_verb^{1} ::= spin^{1}

noun^{2} ::= moon^{1} | planet^{1}

```
proper_noun^6 ::= Hall^1
                | Galileo^1
                | Phobos^1
                | Deimos^1
                | Europa^1
                | Mars^1
```

From the above we can see that even for a small non-recursive grammar the size of the language is **1555920.** This language is a sub-set of the language that can be interpreted by our small example query processor. Therefore, the example illustrates the compositionality of the approach as small semantic definitions can be used to interpret expressions of very large languages.

9.3 Example queries for solar system processor:

Below are some example queries about solar system, which the new semantics can accommodate:

```
Q: hall (discovered phobos)
A:[[Person "hall",Moon "phobos",Time 1873,Implements "with telescope"]]
```

The query "hall discovered phobos" returns a list containing entity hall, entity phobos, time 1873 and implement telescope which means hall discovered phobos in 1873 with a telescope. So the query returns all the information, which is related with hall's discovery of phobos.

```
Q: which person (discovered deimos)
A: [[Person "hall"]]
```

The query "which person discovered deimos" returns a list containing entity hall that is the name of the person who discovered deimos.

```
Q: which person (discovered europa)
A: [[Person "galileo"]]
```

The query "which person discovered europa" returns a list containing entity galileo that is the name of the person who discovered europa.

```
Q: how_many planets spin
A: 3
```

The query "how_many planet spins" computes how many planets spin by doing intersection on the spins set and the planet set and returns 3(according to the information in our database) that means the number of planets that spin.

```
Q: which moon (orbit mars)
A:[[Moon "phobos"],[Moon "deimos"]]
```

The query "which moon orbit mars" returns a list containing the entity phobos in a list and entity deimos in another list which is the names of the moons that orbit mars.

```
Q: which moon spins
A: [[Moon "phobos"],[Moon "deimos"],[Moon "europa"]]
```

The query "which moon spins" return a list containing the entity phobos in a list, entity deimos in a list and entity europa in a list which is the names of the moons that spin.

```
Q: which planet (orbit sol)
A:[[Planet "mars"],[Planet "earth"]]
```

The query "which planet orbit sol" returns a list containing the entity mars in a list and entity earth in another list which is the names of the planets that orbit sol.

```
Q: a moon spins
A: [[Moon "phobos"],[Moon "deimos"],[Moon "europa"]]
```

The statement "a moon spins" return a list containing the entity phobos in a list and entity deimos in a list and europa in a list which is the names of the all moons that spins instead of just returning true or false, which is little ambiguous. Here as the query is not returning an empty list therefore the statement is true, if the query returns an empty list then the statement is false.

```
Q:what (was_discovered_by hall)
A:[[Moon "phobos"],[Moon "deimos"]]
```

The query "what was discovered by hall" returns a list containing entity phobos, and entity deimos in a list which is all the things that hall discovered.

```
Q: with_what did (hall (discovered phobos))
A: [Implements "with telescope"]
```

The query "with what hall discovered phobos" returns a list containing entity telescope that is the implement that hall used to discover phobos.

```
Q: when did (hall (discovered (phobos $term_and deimos)))
A: [Time 1873,Time 1875]
```

The query "when hall discovered phobos and deimos" returns a list containing time 1873, and time 1875 which is the times when hall discovered phobos and deimos. The answer is little ambiguous, as it doesn't specify which time is for phobos and which time is for deimos. We need additional query to the database to get that information.

```
Q: hall (discovered phobos)
A: [[Person "hall",Moon "phobos",Time 1873,Implements "with
telescope"]]
```

The statement "hall discovered phobos" return a list containing the entity hall, entity phobos, time 1873, implement telescope in a list that is all information that has to do with hall's discovery of phobos instead of just returning true or false. This is a little ambiguous. Here as the statement is not returning an empty list therefore the statement is true, if the query returns an empty list then the statement is false.

```
Q: which moon (was_discovered_by hall)
A: [[Moon "phobos"],[Moon "deimos"]]
```

The query "which moon was discovered by hall" returns a list containing the entity phobos in a list, and the entity deimos in a list which is names of all the moons that were discovered by hall.

```
Q:every planet spins
A:[[Planet "mars"],[Planet "uranus"],[Planet "earth"]]
```

The query "every planet spins" return a list containing the entity mars in a list and entity uranus in a list and earth in a list which is the names of the all planets that spins instead of just returning true or false.

```
Q:((when $verb_and who) (discovered phobos))
A:[Time 1873,Person "hall"]
```

The query "when and who discovered phobos" returns a list containing the time and the name of the person who discovered phobos.

9.4 Example queries for the authors and books processor:

Below are some example queries about authors and books, which the new semantics can accommodate:

```
Q: (when $verb_and where_did)(Shakespeare (write hamlet))
A: [Time 1573,Place "England"]
```

The query "when and where Shakespeare wrote hamlet" returns a list containing the time entity 1573 and the place entity England, which is the time and place when and where Shakespeare wrote hamlet.

```
Q: what (was_written_by Shakespeare)
A: [Book "Hamlet",Book "Merchant of Venice"]
```

The query "what was written by Shakespeare" returns a list containing the book entity Hamlet and the book entity Merchant of Venice (according to our database) which is the names of all the books/plays that Shakespeare wrote.

```
Q: who (wrote hamlet)
A: [Name "Shakespeare"]
```

The query "who wrote hamlet" returns a list containing the entitiy shakespeare which is the name of the author who wrote hamlet.

```
Q: how_many books (were_written_by shakespeare)
A: 2
```

The query "how_many books were written by shakespeare" computes how many books shakespeare wrote by doing intersection on the books set and the set returned from the query (written by Shakespeare) and returns 2(according to the information in our database) which is the number of books Shakespeare wrote.

```
Q: what (was_written_by sidney)
A: [Book "Rage of Angels", Book "If Tomorrow Comes"]
```

The query "what was written by Sidney" returns the book entity "rage of angels" and the book entity "if tomorrow comes" which is all the books Sidney wrote according to the information in our database.

```
Q: when did (Shakespeare (write hamlet))
```

```
A: [Time 1573]
```

The query "when did Shakespeare write hamlet" returns a list containing the time entity 1573, which is the time, when Shakespeare wrote hamlet.

9.5 Syntactic/Semantics correspondence

The resulting semantics is fully-compositional. In this approach there is a syntactic/semantic correspondence, that is, phrases of the same syntactic category denote functions of the same semantic type. The type of the denotation of the phrase "every planet" is of the same type as the denotation of the proper noun "Earth". This is consistent with FL's implementation of Montague's approach, which states that words and phrases of the same syntactic category should denote semantic values of the same type. For example using the Miranda type inference system on examples of phrases of the same syntactic category shows that they denote functions of the same semantic type. For example, in our semantics all term phrases has the same semantic type:

```
           Hall                  ::  [[things]]->[[things]]
           Hall $and Kuiper      ::  [[things]]->[[things]]
           A moon                ::  [[things]]->[[things]]
           A (moon $or planet)   ::  [[things]]->[[things]]
           every moon            ::  [[things]]->[[things]]
           no planet             ::  [[things]]->[[things]]
```

And all verb phrases have the same semantic type:

```
          discovered phobos                    :: [[things]]
          discovered (phobos $and deimos)      :: [[things]]
          orbits mars                          :: [[things]]
          was_discovered_by hall               :: [[things]]
          was_discovered_by (hall $or kuiper)  :: [[things]]
          spins                                :: [[things]]
          every moon spins                     :: [[things]]
          hall (discovered (every moon))       :: [[things]]
          hall (discovered phobos)             :: [[things]]
          discovered (no planet)               :: [[things]]
```

```
All noun phrases have the same semantic type:
         moon                  :: [[things]]
         planet                :: [[things]]
         moon $noun_and planet :: [[things]]
```

From above we can see that words and phrases of the same syntactic category denote semantic values of the same type in our semantics.

9.6 Orthogonality

The semantics is orthogonal like Montague's. Many words that appear in different syntactic contexts denote a single function therefore avoiding the need to assign different meaning in these different contexts. For example in the phrases like

1) **Hall** discovered phobos.
   Phobos was discovered by **Hall**.
   Hall and deimos

   Above **Hall** has the same meaning in these three different contexts.

2) **Every** moon spins.
   Hall discovered **every** moon.

   Here also **every** has the same meaning for different contexts.

So our semantics is highly orthogonal as in our semantics the meaning of the majority the words are independent of context. But there is some loss of orthogonality for "and" as we need three different "and" (noun and, verb and, and term and) to handle nouns, terms and verbs. However, this was also a problem with the FL approach.

Orthogonality and the syntactic/semantic relationship guarantee that our semantics will be compositional in the sense that the meaning of expressions of a very large query language can be computed using a very small number of semantic rules. As our semantics meets all the objects therefore we can say that our thesis is proven.

Chapter 10 CONCLUDING COMMENTS

10.1 What has been achieved?

a) The Thesis Statement:

> "It is possible to extend the set-theoretical compositional semantics developed by Frost et al to accommodate n-ary transitive verbs, ($n \geq 2$) by re-defining all denotations to involve sets of attributes rather than simple entities, without loss of compositionality. "

has been proven by:

1. Developing a grammar, even limited to of depth of recursion of 2, for small sub-set for a tiny database, defined by 80 lines of semantics and database, can answer approximately 1,500,000 queries.
2. Showing the results from execution of example queries.
3. Showing that the new semantics maintains orthogonality.
4. And also showing that the new semantics maintains the syntactic/semantic correspondence.

b) A new way to think of semantics for transitive verbs with the arity greater than 2, has been developed.

c) The new approach can be used to define the semantics for transitive verbs of arbitrary n by adding necessary information to the relations and by declaring the new attributes. Existing definitions don't need to be changed. Hence the approach is highly extensible.

10.2 Contribution to Computational Linguistics and Computer Science

Our research has contribution to computational linguistics as we have extended an existing linguistics theory developed by Montague and demonstrated the tractability of its implementation. Montague didn't provide much about how to handle n-place transitive verbs and in our thesis work we have extending the Montague Semantics to handle n-ary transitive verbs.

Montague's approach to transitive verbs is convoluted (see page 20,21). Our approach is to handle n-place transitive verbs in an extending set-theoretic approach of Montague semantics where transitive verbs using λ-notation are defined as: "λz z(λy λx verb(x,y))" which is a straightforward denotation of transitive verbs in Montague style.

Also owing to the one-to-one correspondence between the syntax and semantic rules, our semantics can be readily implemented in a syntax-directed evaluator with a speech-recognition front-end.

10.3 Suggestions for Future Work

This approach could be extended to handle queries like "`which planet lies between earth and mars?`". Currently our approach doesn't handle this type of construct. It can also be extended to include negation using the set-theoretic approach to accommodate negation developed by Frost and Boulos (2002). Also queries like "`John and Mary went to dinner at 7 pm`" are not handled by our approach as the statement has different meaning like did they go together or separately. They investigation of such extensions is appropriate future work.

Bibliography

[1] Androutsopoulos I., Ritchie G. D., and Thanisch P.(1995) Experience using tsql2 in a natural language interface. In: *Proceedings of the International Workshop on Temporal Databases*, Zurich, pp. 113-132.

[2] Androutsopoulos, I. and Ritchie G.D. and Thanisch P. (1995) Natural Language Interfaces to Databases--an introduction. Journal of Language Engineering, Vol 1, No 1, pp. 29-81.

[3] Bennett, Michael. (1974) Some Extensions of a Montague Fragment of English, University of California at Los Angeles: PhD. dissertation; distributed by Indiana University Linguistics Club.

[4] Clifford J. (1990) Formal Semantics and Pragmatics for Natural Language Querying. Cambridge University Press.

[5] Dowty, D. R., Wall, R. E. and Peters, S. (1981) *Introduction to Montague Semantics.* D. Reidel Publishing Company, Dordrecht, Boston, Lancaster, Tokyo.

[6] Frost, R. A. and Launchbury, E. J. (1989) Constructing natural language interpreters in a lazy functional language'. *The Computer Journal – Special edition on Lazy Functional Programming*, **32**(2) 108 – 121.

[7] Frost R. A. and Saba W. S. (1990) A database interface based on Montague's approach to the interpretation of natural language. *International Journal of Man-Machine Studies*, 33(2): 149-176.

[8] Frost, R. A. and Chitte, S. (1999) A new approach for providing natural-language speech access to large knowledge bases. *Proceedings of the Pacific Association of Computational Linguistics Conference PACLING '99*, University of Waterloo, August 1999, 82–89.

[9] Frost R.A., Boulos P. (2002) An Efficient Compositional Semantics for Natural-Language Database Queries with Arbitrarily-Nested Quantification and Negation. *Canadian Conference on AI 2002*: 252-267

[10] Hasting J. D. (1991) Design and Implementation of a Speech Recognition Database Query System, M.S. Department, University of Wyoming.

[11] Karttunen, L. (1976) "Discourse Referents," in J. McCawley (ed.) Syntax and Semantics 7: *Notes From the Linguistic Underground.* (pp. 363-385) New York: Academic Press.

[12] Lapalme, G. and Lavier, F. (1990) Using a functional language for parsing and semantic processing. Publication 715a, Departement d'informatique et recherché operationelle, Universite de Montreal

[13] Larson, R., M. den Dikken and P. Ludlow, (1997) "Intensional Transitive Verbs and Abstract Clausal Complementation", *Linguistic Inquiry.*

[14] Levin, Beth. (1993) English Verb Classes and Alternations: A Preliminary investigation. Chicago: University of Chicago Press.

[15] McCawley, J. (1979) "On Identifying the Remains of Deceased Clauses," in J. McCawley (pp. 74-85).

[16] Miyagawa, Shigeru & Takae, Tsujioka. (2004) Argument structure and ditransitive verbs in Japanese. *Journal of East Asian Linguistics* 13: 1-38.

[17] Popescu M. A., Etzioni O. and Kautz H., (2003) Towards a Theory of Natural Language Interfaces to Databases. IUI.

[18] Reis P., Mamede N., Matias J. (1997) Edite -- A Natural Language Interface to Databases: a New Dimension for an Old Approach in "*Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism*", ENTER' 97, Edinburgh, Scotland.

[19] Ross, J. (1976) "To Have Have and to Not Have Have", in M. Jazayery, E. Polom, and W, Winter (eds.) *Linguistic and Literary Studies in Honor of Archibald Hill.* (pp. 263-270).

[20] Stratica N., Kosseim L. and Desai B.C. (2002) A Natural Language Processor for Querying Cindi . *In Proceedings of International Conference Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet* (SSGRR 2002s), L'Aquila, Italy.

[21] Yates A. and Etzioni O. and Weld D. (2003) Reliable natural language interfaces to household appliances. IUI-03.
[Keywords: Natural language interface, database, appliance, and planner.]

[22] Yonezaki N. and Enomoto H. (1980) Database system based on intensional logic, COLING-80, pp. 220-27

Related Publications by the Author

[1] Roy M. and Frost R.A. (2004) Extending Montague Semantics for Use in Natural-Language Database-Query Processing. *Canadian Conference on AI2004:567-568*

Appendix A – Program listing

Program listing:

Below we present an illustrative example query processor, based on our semantics in Miranda, which can answer various questions about our solar system:

```
entity == [char]
time == num

things   ::=  Person       entity
            | Implements entity
            | Moon         entity
            | Time         time
            | Planet       entity
            | Sun          entity


planet   = [[Planet "mars"], [Planet "uranus"], [Planet "earth"]]

planets = planet

moon = [[Moon "phobos"], [Moon "deimos"], [Moon "europa"]]

moons = moon

spin = [[Moon "phobos"],   [Moon "deimos"],
        [Planet "uranus"], [Moon "europa"],
        [Planet "mars"],   [Planet "earth"],
        [Sun "sol"]]

spins = spin

people = [[Person "hall"], [Person "galileo"], [Person "kuiper"]]


mars     l = [ s | s <- l; member s (Planet "mars")]

hall     l = [ s | s <- l; member s (Person "hall")]

phobos   l = [ s | s <- l; member s (Moon "phobos")]

galileo l = [ s | s <- l; member s (Person "galileo")]

europa   l = [ s | s <- l; member s (Moon "europa")]

kuiper   l = [ s | s <- l; member s (Person "kuiper")]

deimos   l = [ s | s <- l; member s (Moon "deimos")]

uranus   l = [ s | s <- l; member s (Planet "uranus")]

sol      l = [ s | s <- l; member s (Sun "sol")]
```

```
earth  l = [ s | s <- l; member s (Planet "earth")]


every  s t        = s, if subset s (remove t)
                  = [], otherwise

remove  t    = [ [a] | a:aa <- t]

subset x  y  = True , if (x -- y) = []
                  = False, otherwise

a s t        = [l:mm | l:ll <- s; m:mm <- t; l = m]

no s t       = [], if res ~= []
             = [[Value "true"]], otherwise
                       where res = a s t


intersect s t = s -- (s -- t)

union s t = s ++ ( t -- s)

noun_and s t = union s t

verb_and s t = f
               where f ents = s ents ++ t ents


term_or g h = r
              where r s = (g s) ++ (h s), if g s ~= []  \/ h s ~= []
                        = [], otherwise


term_and p q = r
               where r s = [], if p s = []  \/ q s = []
                         = (p s) ++ (q s) , otherwise



make_transitive_verb rel p = mkset [x : t| (x, s) <- collect rel; t <-
(p s); p s  ~= []]


collect []            = []
collect ((x:t):r)     = (x,t:[s | (a:s)<- r; a = x]):
                             collect [l:f | (l:f) <- r; l ~= x]


discover_rel =
[[(Person "hall"), (Moon "phobos"), (Time 1873), (Implement "with telescope")],
[(Person "hall"),  (Moon "deimos"), (Time 1875), (Implement "with telescope")],
[(Person "kuiper"), (Moon "uranus"),(Time 1860), (Implement "with telescope")],
[(Person "galileo"),(Moon "europa"),(Time 1820), (Implement "with telescope")]]


orbit_rel = [[(Moon "deimos") ,(Planet "mars")],
             [(Moon "phobos") ,(Planet "mars")],
             [(Planet "mars") ,(Sun "sol")],
             [(Planet "earth") ,(Sun "sol")]]

discovered = discover

discover   = make_transitive_verb discover_rel

is_discovered_by = make_transitive_verb (invert discover_rel)


orbit = make_transitive_verb orbit_rel

orbits = orbit

is_orbited_by = make_transitive_verb (invert orbit_rel)

invert rel = [ (y:x:ys):s |(x:y:ys):s <- rel]

how_many s t = #(intersect s t )

which = intersect

what x = x

does x =x

is = does

when x = x

did x = x

who x = x
```

Below we present an illustrative example query processor, based on our semantics in Miranda, which can answer various questions about authors and books:

```
entity == [char]
time == num

things   ::= Name       entity
             | Book       entity
             | Place      entity
             | Time       time
             | Value      entity

book  = [[Book "Hamlet"],          [Book "Merchant of Venice"],
         [Book "Rage of Angels"], [Book "If Tomorrow Comes"]]
books = book

author = [[Name "shakespeare"], [Name "sidney"]]
authors =author

shakespeare l        = [ (a:as) | (a:as) <- l; a = (Name
"shakespeare")]
```

```
sidney l             = [ (a:as) | (a:as) <- l;  a = (Name "sidney")]
hamlet l             = [ (a:as) | (a:as) <- l;  a = (Book "Hamlet")]
merchant_of_venice l = [ (a:as) | (a:as) <- l;  a = (Book "Merchant of Venice")]

rage_of_angels l    =  [ (a:as) | (a:as) <- l;  a = (Book "Rage of Angels")]
if_tomorrow_comes l =  [ (a:as) | (a:as) <- l;  a = (Book "If Tomorrow Comes")]


every s t = s, if subset s t
          = [], otherwise
a s t     = [l | l <- s; m <- t; l = m]
no s t    = [], if res ~= []
          = [Value "true"], otherwise
            where res = a s t

intersect s t = [ l | l<-s; m<-t; n<- m; member l n ]

first_element (e:es)= e

union    s t = s ++ ( t -- s)
subset   x y = (x -- y) = []
noun_and s t = union s t
verb_and s t = f
               where f ents = s ents ++ t ents

term_or g h = r
              where r s = (g s) ++ (h s), if g s ~= []  \/ h s ~= []
                        = [], otherwise
term_and p q = r
               where r s = [], if p s = []  \/ q s = []
                         = (p s) ++ (q s) , otherwise


make_transitive_verb rel p = mkset [x : t| (x, s) <- collect rel;
                                    t <-(p s); p s  ~= []]

collect []           = []
collect ((x:t):r)    = (x,t:[s | (a:s)<- r; a = x]):
                             collect [l:f | (l:f) <- r; l ~= x]

written_rel =
[[(Name "shakespeare"),(Book "Hamlet"),            (Time 1573),(Place "England")],
 [(Name "shakespeare"), (Book "Merchant of Venice"),(Time 1575),(Place England")],
 [(Name "sidney"),      (Book "Rage of Angels"),    (Time 1950),(Place "USA")],
 [(Name "sidney"),      (Book "If Tomorrow Comes"), (Time 1940),(Place "USA")]]


wrote = write
write  = make_transitive_verb written_rel
was_written_by = make_transitive_verb (invert written_rel)


died = [[(Name "shakespeare"), (Time 1620)],
        [(Name "sidney") ,     (Time 0)]]
born = [[(Name "shakespeare"), (Time 1530)],
        [(Name "sidney"),      (Time 1920)]]
lived =  [[(Name "shakespeare"),(Place "England")],
          [(Name "sidney") ,    (Place "USA")]]

lives = lived
live= lives


invert [] =[]
invert ((x:y:ys):es) = (y:x:ys): invert es

how_many s t = #(intersect s t )

which s t    =  [l | l <- s; m <- t; n <- m; member l n]
what x       = [a | (a:b:c:s)<- x]

does x = x
is     = does

when l = [ u | v<- l; u <- v; a_time u]

a_time (Time x) = True
a_time   any    = False

where_does = where_did
where_did l = [ u | v<- l; u <- v; a_place u]

a_place (Place x) = True
a_place   any     = False


did x = x
who x = [ a | (a:s) <- x]
```

Appendix B – A Survey

A Survey on: Use of Montague and Montague-like Compositional Semantics in Natural Language Database Query Processing

Introduction:

This is a survey on the use of Montague and Montague-like compositional semantics in natural-language database-query processing. In section 1 of the survey compositional semantics is introduced. Composition semantics for natural language is described in section 2. Section 3 contains semantics in parsing natural-language, and natural-language interfaces to databases are described in section 4. Finally, in section 5 of the survey, Compositional Semantics for Natural Language Database Queries is described

1. Compositional Semantics

Compositional Semantics, abbreviated in this survey to CS, is defined as a functional dependence of the meaning of an expression on the meaning of its parts. It is called compositional semantics because of the crucial part played by the principle of compositionality: that the meaning of the whole sentence is composed from meanings of its parts. The books [Schmidt, 1986] & [Stoy, 1997] are good introductions to compositional semantics.

2. Compositional Semantics for Natural Language

2.1 General introduction to Computational Linguistics

Computational Linguistics (CL) originated from the Machine Translation Research of the '50s and '60s. The study of computer processing, understanding and generation of human language is known as Computational Linguistics (CL). Computational linguistics is sometimes regarded as a subfield of artificial intelligence. In different applications such as machine translation, speech recognition, information retrieval, intelligent web searching and intelligent spelling checking, techniques from computational linguistics are used. Computational linguistics is devoted exclusively to the design and analysis of natural-language processing systems.

The paper [Blackburn and Bos, 2003] gives a good introduction to the Computational Semantics of Natural Language. This paper introduces the basics of natural-language semantics. It describes first-order logic, lambda calculus and underspecified representations such as scope ambiguities (e.g. John advertised one house on every street) and Montague's approach. More general information on computational linguistics can be found in [Lewis and Carl, 1985] and [Tore, 2002].

2.2 Meaning of Words

Semantics is concerned with the meaning of words and how they combine to form sentence meanings. There are many ways of representing word meanings but one way, which has proven to be one of most useful, is in the field of machine translation involving associating words with semantic features, which correspond to their sense components. The book [Dowty, 1979] on Word meaning and Montague Grammar is a good introduction to areas related to meaning of words.

[Thomason, 1991] talks about some possible problems in lexical semantics, which the author thinks are both exciting and challenging and which can be solved by cooperative research between linguists and computer scientists.

[Thomason, 2001] proposed an approach, the logical approach, which they claim has never produced a very satisfactory account of word meaning but is successful in the semantic interpretation of syntactic structure. For example the natural way to define 'x is water soluble' is as follows:

If x were put in some water, then x would dissolve in the water.

The definition of 'water-soluble' is obtained by using eventualities in place of times (This formula uses more or less standard formalization techniques in event-centered semantics, for example [Push(e) ^ Past(e) ^ Pusher(e) = Charlie ^ Pushee(e) = Piano] is used to represent Charlie pushed the piano .)

.

$$\forall x\ [\ \text{water-soluble}(x) \leftrightarrow\ \ \forall e_1 \forall y\ [\ \text{put\_in}(e_1) \wedge \text{movee}(e_1) = x \wedge \text{container}(e_1) = y \wedge \text{water}(y)] \rightarrow \exists e\ [\ \text{Dissolving}(e) \wedge \text{dissolvee}(e) = x \wedge \text{medium}(e) = y\ ] \wedge \neg \text{Ab}(e)] \rightarrow \exists\ e_2\ [\text{culmination}(e) = e_2 \wedge \text{dissolved}(e_2)\ \wedge \text{disolvee}(e_2\,) = x \wedge \text{medium}(e_2) = y\ ]]$$

In words: x is water-soluble if and only if necessarily if an event $e1$ of putting x in a quantity of water occurs then $e1$ is the inception of a dissolving eventuality e involving the same x and quantity of water, which unless something abnormal about e will culminate in a state in which x is dissolved.

An extension to Montague's framework is proposed and some of its applications in the semantics of words are illustrated in [Thomason, 2002].

2.3 Montague and Montague-style Semantics and extensions

Model-theoretic semantics of natural language is a way of analyzing the meanings of NL expressions. Richard Montague introduced the technique in two classical papers entitled Universal grammar [Montague 1974] and The proper Treatment of Quantification in Ordinary English [Montague, 1970], which is known as PTQ. Universal Grammar, which is a predominantly theoretical treatise, refers to the branch of mathematics called universal algebra from which the main techniques were adopted. PTQ, on the other hand, applies these theoretical principles to 'ordinary English'. Grammars based upon Montague's PTQ are called Montague grammars.

A Montague grammar is a grammar for a particular fragment of natural language which consists of three components: the syntax which is a syntactic analysis of the expressions of the fragment, the translation translating natural language into a logical language and the model theory or the semantics, and a (model-theoretic) interpretation of the expressions of the logical language

Montague-style semantics (see Dowty, Wall and Peters, 1981) has been used in natural-language processing. Montague Semantics has been one of the most influential theories in the semantics of natural languages in the tradition of truth-conditional, model-theoretic and intensional semantics. A Montague grammar is a theory of the semantic effects of composition.

Montague Semantics can be implemented and has been used as a semantic basis in a number of implemented systems for natural language querying [e.g. Clifford 1990, Frost and Launchbury 1989, Frost and Boulos 2002].

[Frost and Launchbury, 1981], [Frost and Saba 1990] and [Frost and Boulos, 2002] describe an efficient implementation of Montague's semantics in a set-theoretic framework [details are given in section 5.2]

[Groenendijk and Stokhof, 1990] propose a new logical system as the semantic component of a Montague–style grammar that extends the compositionality of DPL (dynamic predicate logic) to the sub-sentential level. In DLP (Dynamic Predicate Logic) a sentence such as "Every farmer who owns a donkey beats it" can be translated into the formula as follows:

$\forall x[[\ \text{farmer}(x) \wedge \exists y\ [\text{donkey}(y) \wedge \text{own}(x, y)]] \rightarrow \text{beat}(x\ ,y\ )\ ]$

In DLP the above translation is equivalent to :

$\forall x \forall y\ [[\ \text{farmer}(x) \wedge \text{donkey}(y) \wedge \text{own}(x, y)] \rightarrow \text{beat}(x\ ,y\ )\ ]$

It is a continuation of their work [Groenendijk and Stokhof, 1989] on dynamic predicate logic.

A recent approach extending the classical Montague semantics can be found in [Muskens, 1995]. In his book the author presents a semantics of possibly-contradictory beliefs and other propositional attitudes.

[Malinowski, 1996] suggests semantics for illocutionary logic (Serale's and Vanderveken's), which is based on Montague's intensional logic.

[Eijck 1999] proposed that a Montague-style architecture for NL semantics provide proper treatment both of quantification and of context use and context change. In his paper, the author refers to the work done by [Groenendijk and Stokhof, 1989, 1990].

[Nelken and Francez, 2000] suggest a new semantic interpretation of interrogative NPs (noun phrases), which play an important role in driving the interpretation of wh-questions such as "which women". The authors used a formal language called Intensional Logic with Questions (ILQ) which extends Montague's IL. The authors added two operators: the interrogative operator (?) used for yes/no questions and the binding interrogative operator (?x) used for constituent questions. For example, here the authors interpret the interrogative determiner "which" as:

$[\text{ Det which }] = \lambda P \lambda Q.\ ?x\ (P(x) \wedge Q(x))$

which is similar to the standard interpretation of the determiner "a" :

$\text{a: } \lambda P \lambda Q\ \exists x\ (P(x) \wedge Q(x))$

So the meaning of the sentence "which woman kissed John" can be interpreted as follows:

[Which woman kissed John]

$= [\text{INP which woman}]\ ([_{VP} \text{ kissed John}])$

$= \lambda Q.\ ?x\ (\text{ woman}(x) \wedge Q(x))\ (\lambda y.\ \text{kiss}\ (y, \text{John}))$

$= ?x\ (\text{ woman}(x) \wedge \text{kiss}\ (x, \text{John}))$

The authors refer to some of the work done by [Eijck, 1996]

[Onet and Doina, 2001] describe the fundamentals of intensional logic and introduce some methods for treating quantitative natural sentences. Authors split quantitative sentences in three categories: definite quantity sentences (e.g. "Four women cry"), indefinite quantity sentences (e.g. "Most women cry"), restrictive quantity sentences (e.g. "Maximum five children answer") and tried to translate them in to intensional logic.

In [Cimiano, 2003] the author presents a approach to map natural-language wh-questions into F(rame)-logic queries based on Montague-style compositional semantics where semantic representation is constructed on the basis of Lexicalized Tree Adjoining Grammar LTAG-style derivation tress.

[Perez , 2003] shows how semantic interpretation and parsing of a sentence can be accomplished in a compositional way by defining semantic rules that work in a one-to-one correspondence with the syntactic ones.

Coverage of Natural-Language Semantics

Year	Authors	Work
1972	Montague R.	Natural-language semantics for noun, pronoun, intransitive verbs

1989	Frost and Launchbury	Implementation of a set-theoretic version of sub-set of Montague Semantics
1990	Clifford	Used semantics as a basis to implement systems for Natural-language querying
1990	Groenendijk and Stokhof	Proposed a system which use Montague Semantics to extend the compositionality of DLP (Dynamic Predicate Logic)
1990	Frost and Saba	Used Montague Semantics to implement natural-language interfaces to databases
2001	Onet and Doina	Extended Montague Semantics to handle quantitative natural sentences e.g. "Four women cry" etc..
2002	Thomoson	Used Montague framework in semantics of words
2002	Frost and Boulos	Implemented compositional semantics for database queries based on a set-theoretic version of Montague semantics to accommodate negation

Table: Coverage of Natural-Language Semantics

2.4 Alternative approaches to Montague Semantics

Rather than Montague semantics, other approaches have been used in natural language. [Hardt, 1996] presents a dynamic framework, a dynamic logic system, with extensions for the discourse center (a distinguished discourse entity that is the topic of a discourse), VP ellipsis (Verb Phrase ellipsis) and paycheck pronouns. (A paycheck pronoun is a pronoun, which exhibits sloppy identity, for example "Smith spent his paycheck. Jones saved it.". Here "it" is not an ordinary bound pronoun, nor is it an ordinary free pronoun.

[Shan, 2001] introduces a new variable-free dynamic semantics, which means denotational semantics for natural language where meanings of constituents are updates to information states. The author continued the work done by [Groenendijk and Stokhof, 1990]

Shan [2001] analyzed sentences such as "A man walks in the park. He whistles." For example, the author wrote e for the type of an individual, $e \to 1$ for the type of a property and $e \to e \to 1$ for the type of a two-place relation. So, the derivation of the sentence " A man walks in the park" is translated as follows:

$$\text{A}: (e \to 1) \to e = \lambda p.\ \{\ v \mid * \in p(v)\ \}$$

$$\text{Man}: e \to 1,\ \text{WITP}: e \to 1,\ \text{WITP(A(MAN))}:1$$

And, whistles denotes some property WHISTLE: $e \to 1$ and "he" denotes

$$\text{HE}: e \blacktriangleright e = \lambda v.v$$

where $\blacktriangleright$ ("in") is a new binary type constructor where type $\sigma \blacktriangleright \tau$ is like $\sigma \to \tau$ in that they may have the same models, namely functions from σ to τ.
So now "He whistles" can be derived as follows:

$$g^{\blacktriangleright}\text{(WHISTLE) (HE)}: e \blacktriangleright 1 = \lambda v\text{: WHISTLE}(v)$$

where $g^{\blacktriangleright}$ is a type-shift operation such as

$$g^{\blacktriangleright} : (\alpha \to \beta) \to (\sigma \blacktriangleright \alpha) \to (\sigma \blacktriangleright \beta) = \lambda f.\ \lambda v.\ \lambda s.\ f(v(s))$$

[Fox and Pollard 2002] present PTCT (Property Theory with Curry Typing) where a language of types joins the language of terms and well-formed formula. [Shan, 2002] characterizes the similarity between several semantics accounts for interrogatives, focus, intensionality, variable binding & quantifications by using monads. A monad is a structure from category theory.

[Bernardi, 2003] describe a logical system, which has the ability to compute the semantics of both declaratives and interrogative sentences. For example, the author analysed sentences such as:

Q: Did Tarantino direct Titanic?	$\lambda Y(Y((\text{direct titanic})\ \text{tarantino}))$
A: No	$\lambda p\ \neg p$
Q(A) By twice beta-reduction	$\neg((\text{direct titanic})\ \text{tarantino})$

The authors also considered "what" as an example:

Q: what did Cameron direct?	$\lambda Y(Y\lambda\ x((\text{direct } x)\ \text{cameron}))$
A: Titanic	$\lambda P\ P(\text{titanic})$
Q(A):By twice beta-reduction	$((\text{direct titanic})\ \text{cameron})$

3. Semantics in Parsing Natural Language

Semantic parsing is a difficult problem in natural-language analysis [Hirst, 1987]. During sentence analysis, the question of the appropriate interaction of syntax and semantics has been of interest for a long time. The early work on semantic parsing was done in 70's [Siklossy 1972; Reeker, 1976] with emphasis on cognitive modelling of human language learning and on discovering mechanisms for language acquisition.

According to Warren [1982], a complete, well-defined context in which these questions can be considered is provided by Montague grammar with its fully formalized syntax and semantics. [Warren, 1982] describes how to reduce the combinatorial explosion of syntactic ambiguity by using semantics during parsing in Montague grammar.

In [Lang and Hirschman 1988] the authors show how parsing can be improved through interactive acquisition of semantic information.

[Mosny, 1995] proposes an approach to extract constraints, which are explicitly or implicitly provided by a semantic part of the natural language interface to a database, from the semantic description of the database domain and incorporate them into information directly accessible to the parser.

[Da-Silva, Seabra and Siqueira, 1995] propose a parser that performs syntactic and semantic analysis, simultaneously as in Montague Grammar, of assertions, which are related to Space Science and are expressed in a restricted form of natural language.

[Chan, 1997] shows how semantic parsing can be formulated as a sequence of processes in which multiple sources of knowledge are incorporated.

[Miller, Fox, Ramshaw and Weischedel , 2000] introduce a statistical, context-free probabilistic parser for information extraction which shows a significant increase in parsing accuracy .

[Lappoon et al, 2000] propose a method for learning semantic parses, which are systems for mapping natural language to logical forms that integrate logic-based and probabilistic methods. [Lappoon et al, 2000] also present a method for integrating statistical and relational techniques for the automated acquisition of NLI's from training examples. They also claim that their approach is more robust than a purely logical approach.

4.Natural-Language Interfaces to Databases (NLIDBs)

4.1 Overview of Natural-Language Interfaces to Databases

A natural-language interface to a database is a system that allows the user to access information stored in a database by typing requests expressed in some natural language such as English. The first natural-language interfaces to databases appeared in the late sixties and early seventies. According to [Androutsopoulos, 1995], the best-known NLIDB at that period was LUNAR [Woods, 1972], a natural-language interface to a database containing chemical analyses of moon rocks. Some other NLIDBs developed at that time were RENDEZVOUS, LADDER, PLANTES and PHILIQA1 [description of and references for all of these systems can be found in Androutsopoulos, 1995]. Marjorie and Burger describe some of the problems in natural language and database management involved in natural-language interface development [Marjorie and Burger 1983]. More NLIDBs were developed in eighties and nineties.

In recent years, there have been a significant number of papers on NLIDBS published each year and NLIDBS continue to evolve, adopting advances in the general natural- language processing field, exploring architectures that transform NLIDBs into reasoning agents, and integrating language. [Androutsopoulos, 1995 (which is a good introductory paper on NLIDBs)], talks about the history of NLIDBs, some advantage and disadvantage of NLIDBs and also compares NLIDBs to formal query languages, form-based interfaces, and graphical interfaces.

Focus on the central process of translating the natural-language questions into database queries has also been investigated by some researchers [e.g. Copestake and Jones 1990]. Different approaches have been applied to NLIDBs. Demers in his thesis, introduces a lexicalist approach, which is based on unification grammars to database NLI's along a small-scale example [Demers, 1996]. The author claims that the solution proposed to this approach is not only feasible but also provides reasonable complexity and processing time for unambiguous words and expressions.

Many techniques have been developed to translate natural-language questions into database queries. [Filipe and Mamede, 2000] mainly focus on the translation stage, translating user questions first into a logic language and then into Structured Query Language (SQL) [more details and examples of SQL-type interfaces are given in the next section], which is then processed by a database-management system to return answers to the questions.

4.2 SQL-type Interfaces

A lot of natural-language interfaces that have been developed are based on an SQL-type approach. [Hasting, 1991] describes the design and implementation of an SQL-based speech-recognition database-query system.

[Androutsopoulos, 1995] talks about using a language called TSQL2 in a natural-language interface. The paper [Androutsopoulos, 1995] focuses on the TSQL2 in a natural-language interface for temporal databases and also in some point on the semantics of TSQL2. For example the question " On how many Mondays was John at University of Windsor in 2000? " can be expressed as:

Select Snapshot Count (Distinct d.*)

```
From &year_month_day (Period) As d, student_visits (Period) as t
Where d.year = 2000
AND d.day_name= "Monday"
AND VALID(t) OVERLAPS  VALID(d)
AND t.student='John'
AND t.school='University of Windsor'
```

Assuming that the calendric table and year_month_day and student_visits tables are available.

[Reis and Mamede, 1997] present the Edite system, which is a natural-language interface to databases, and explore the advantage of joining natural-language processing with the expressiveness of graphical interfaces. Edite, a natural-language front-end for relational databases, is multi-lingual (Portuguese, French, English, Spanish). It is capable of answering written questions related to tourism by transforming them into SQL queries. The answer can be a list of resources, text, images or graphics depending of the questions. At present, the database contains 53000 tourism resources, arranged on 253 distinct types, which corresponds to 209 tables. This paper refers to the work done by [Androutsopoulos, Ritchie, Thanisch, 1993].

[Stratica, 2002] talks about a natural language processor for querying Cindi, which is also an SQL-based system.

A reliable natural-language interfaces to household appliances which is also an SQL-based interface is described in [Yates and Etzioni, 2003].

[Popescu, Etzioni and Kautz 2003] introduces a theoretical framework, which is the foundation for the fully implemented Precise NLI and proved that Precise guarantees a map for each question to the corresponding SQL query, for a broad class of semantically-tractable natural-language questions.

4.3 Other approaches to Natural-Language Interfaces

There are different approaches to natural-language interfaces. [Ryan and Root, 1988] describe some application-specific issues in developing of natural-language interfaces.

A fully-statistical approach to a natural-language interface, which consists of three stages of processing: parsing, semantic interpretation and discourse, is described in [Scott and David 1996]. All of the stages are modeled as a statistical process, which are integrated, resulting in an end-to-end system that maps input utterances into meaning-representation frames.

A deductive object-oriented approach in the development of natural-language interfaces that uses a deductive object-oriented database (DOOD) is described in [Werner and Yahiko 1997]. The authors follow the approach of [Rymon, 1993] and refer to [Androutsopoulos, Ritchie, and Thanisch, 1995] in the paper.

5 Compositional Semantics for Natural-Language Database Queries

5.1 Introduction to Semantics in Databases

Issues related to database semantics played an important role in the early days of database research and most of the database conferences were dominated by the papers discussing database models, conceptual design, integrity constraints and normalization. Semantics of databases and information systems can be based on approaches, which have been developed and successfully used by different communities such as the logic community who are working on constraint problems, induction, non-classical semantics, the database-theory community who are working on constraints, and the AI community who are working on logic and reasoning, deduction, agents etc.

According to [Teskey, 1987] semantic models developed by linguists have not had any significant impact on information retrieval.

[Kalita, Jones and McCalla 1986] describes the detailed design and implementation of a system, which generates summary responses to queries of a relational database.

[Ranta, 1999] describes a database-query system based on the Grammatical Framework which was demonstrated using a database of restaurants which runs in seven European languages and the system can be modified in various levels like changing basic grammatical structures into other structures.

In [Hausser, 2001a] the author presents a new approach where the spatio-temporal location of propositional content is not specified precisely within a Cartesian system of space and time coordinates instead it is characterized cognitively by the order of direct observations entering the database of a cognitive agent. The sequence of propositions which serves as the spatial landmarks are structured by observations of the environment and temporal landmarks are structured by observations of cyclical events. In database semantics, like all other inference, which navigate through the concatenated propositions, spatio-temporal inferences are handled.

[Hausser, 2001b] describes database semantics as a declarative model of a cognitive agent, which is called a SLIM machine and which functionally integrates the procedures of natural-language interpretation, conceptualization and production. No one appears to have referred to this work at this point of time.

5.2 Database interface based on Montague's Approach

There has not been much research on building database interfaces based on Montague's Semantics. [Frost and Launchbury, 1989] describe how in a functional-programming language, natural-language parsers and interpreters can be implemented. Frost and Launchbury refer to the book by Dowty, Wall and Peters (1981) but make no reference to any previous work on the use of Montague semantics in database query processing. It appears that Frost and Launchbury were amongst the first to use Montague semantics in database query processing.

It would also appear that Frost and Launchbury were the first to use a set-theoretic based implementation of Montague semantics. For example instead of interpreting 'every' like in Montague as:

$$[\text{every}] = \lambda p \lambda q\ [\ \forall p(x) \rightarrow q(x)\ ]$$

Frost and Launchbury used:

$$[\text{every}]_{FL} = \lambda p\ \lambda q\ \ p \subseteq q$$

According to [Yonezaki and Enomoto 1980], Richard Montague's Intensional Logic (IL), which describe semantics of natural language, can be useful to the theory of databases in designing database systems which handle historical data and provide a formal description of database semantics.

[Frost and Saba 1990] implemented some of the concepts of Montague that can be used in natural-language interface to databases. The database interface is implemented in a higher-order functional programming language and the semantic calculation is achieved through higher-order functional application.

[Lapalme and Lavier 1990] showed how a larger part of Montague Semantics can be implemented in a pure higher-order functional programming language.

[Frost and Boulos, 2002] describe an implementation of a compositional semantics for database queries, based on a set-theoretic version of Montague semantics, which accommodates phrases that include the word 'no'. The approach is based on an extended set theory in which 'negative' phrases denote infinite sets represented in complement form.

5.3 Question-Answering

The question-answering systems developed in the 1970's were complex AI-based systems that converted a natural-language query into a knowledge-base query. Those systems then searched in the knowledge base for an answer and returned the results in natural language. Constructing and maintaining those knowledge bases was a great problem and those systems were not scalable. The LUNAR system (Woods, 1977) is one of the examples of those systems. Recently triggered by the Text Retrieval Conference (TREC) Question Answering Track (Voorhees, 2001) there has been an increase in research on text-based question answering.

According to [Main and Benson, 1983] denotational semantics can be used as a specification technique for question-answering programs and implementation of the principle of compiler design was suggested as principle of question answerer design.

There is an interesting paper [Zweigenbaum, 2003] in which question answering is used in biomedicine for natural language question answering.

[Duclaye and Yvon, 2003] discussed several methods on how to improve question-answering systems. The authors presented an unsupervised methodology starting with one single positive learning example for automatically learning paraphrases and which is able to filter out the invalid potential paraphrases extracted during the acquisition steps using an EM-based validation. The authors claim that these paraphrases are useful to improve the results of their question-answering system.

[Katz and Lin, 2003] describe how to improve precision in question answering by selectively using relations.

5.4 Predicate-logic -based Approaches

Much research has been done on predicate-logic-based approaches for building natural-language database interfaces. [Rayner, 1993] in his Ph.D thesis discusses abductive equivalential translation and its application to natural-language database interfacing.

5.5 Approximate answer from cooperative sources

As databases and information systems often do not explicitly attempt to cooperate with their users, they are sometimes hard to use. Direct answers may not always be the best answer to database and knowledge-base queries. On the other hand, a more-useful and less-misleading answer to a user may be an answer with extra or alternative information. [Gaasterland, Parke and Minker, 1992] describe intelligent information systems, which are able to exhibit cooperative behaviour.

[Pankowski, 1999] talks about semantics of approximate answers in cooperative database systems.

5.6 Semantics of Dialogues

A computer system and a human user work cooperatively via a natural-language interface to achieve a certain goal in task-oriented dialogues. A typical example of task-oriented dialogues are information-querying interactions where the system reports information about e.g. bus schedules on the basis of certain input-parameters of the user.

Most of the current task-oriented dialogue systems interpret user utterances by directly mapping them onto parameters that represent the questions the user has to answer. [Malte Gabsdi, 2001] in his master's thesis talked about interpreting questions and answers in a prototype dialogue system.

5.7 Natural-Language Interfaces to Temporal Databases

Natural-language database interfaces have been the subject of interest in the natural language processing community since the 1960s. Users are able to access information stored in database through NLDBs by simple formulating requests in natural language.

Most existing NLDBs are designed to interface to database systems provided very little facilities for manipulating time-dependent data. Most NLDBs also provide very little temporal support. Temporal database systems are becoming increasingly interesting in the database community. These temporal database systems are intended to store and manipulate information not only about the present, but also about the past and future.

The work of Clifford and Warren [1983] is one of the first attempts to incorporate a concept of time in database.

[Clifford and Warren, 1983] has discussed that formal logic has made important contribution in understanding and specification of the semantics of database. Authors showed that relational database model could be extended to incorporate the concept of historical relations as well as database and also shown how ILs (reformulated IL to include *s* as a basic type) can provide a semantic theory for this database concept. In this paper the authors also suggested as interesting aspect in defining the translation of English questions into ILs, where the authors interpret English statements as database commands. For example, the authors interpreted the statement ‘ John earns 30k’ as a command to record this as a fact in the database with the time-stamp taken from the system clock when made by an authorized user.

In [Hirst, 1983] the author proposed a new approach to semantic interpretation based on the semantic formalism of Richard Montague. In this approach author claim that their semantics are compositional by design and strongly typed like Montague and they replace Montague's semantic objects and truth conditions with the elements of the frame language Frail and added a word sense and case slot disambiguation system. They claim that their approach to semantic interpretation is superior to previous approaches. For example a single noun phrase the book can be interpreted as (the ?x (book ?x)), which is a Frail frame statement. And a descriptive adjective correspond to a lot-filler pair from example red is represented by (color=red), so the red book would have semantic interpretation (the ?x (book ?x(color=red))). Similarly the sentence "Nadia bought the book from a store in the mall " will be interpreted as

(a ?u (buy ?u (agent = (the ?x (thing ?x (propername= "Nadia"))))
(patient = (the ?y (book ?y))) (source = (a ?z (store ?z (location =
(the ?w (mall ?w))))))

In [Clifford, 1988] the author examines the connection between the semantics of historical databases and the semantics of natural language querying and through a common view of the semantics of time link them together. [Clifford, 1988] demonstrated the use of QE-III, a formally defined English database query language whose semantics and pragmatic theory are based on a Montague type semantics and discussed the issues on providing both semantics and pragmatic interpretation for question within a model-theoretic framework. For example questions in English Query Language QE-III can be handled in the following way:

Who is Peter's manager?

which can be interpreted as:

$$\lambda u\ \exists x\ [\ MGR'(now)(x)\ \Lambda\ x(now) = u\ \Lambda\ AS\text{-}1(Peter,x)]$$

In [Hinrichs, 1988], the author argued that a logical semantics for temporal expressions could provide sufficient representations for natural-language inputs to an interface such as JANUS, a natural language understanding and generation system under joint development by BBN Labs and ISI. The author demonstrated that if narrow scopes are given to tense quantifiers that will enable to provide adequate scope relation with respect to natural-language quantifiers and to interpret such NPs relative to a given discourse context. The author also demonstrated that how in English the narrow scope of tense results in a fully compositional syntax and semantics of tensed sentences.

In their paper (which is a good introductory paper on temporal Databases), Androutsopoulos, Ritchie and Thanisch, in 1998 suggest a new framework for constructing natural language interface for temporal database as at that point of time most of the natural language database interfaces designed had very limited facilities for manipulating time-dependent data and didn't support temporal linguistic mechanisms. The authors refer to the work done by [Clifford and Warren, 1983] in temporal databases.

[Claire, 1990] describes the implementation of formal semantics as described in Keena and Faltz *Boolean Semantics for Natural Language for Natural Language*. The author claims that his implementation avoids the intermediate step of translating Natural Language into a formal language such as an extended version of predicate calculus which makes his implementation free of the problems related to the syntax of such a language like binding the variable and resolving scope ambiguities however which has disadvantage that every denotation (i.e. semantic value) requires to be explicitly and accurately represented in a database.

In [Kabanza, St'evenne, and Wolper, 1990] the authors present a framework, which is an extension of classical relational database, for describing, storing and reasoning about infinite temporal information and this framework represents infinite temporal information by generalized tuples which are defined by linear repeating points and constraints on these points. Authors prove that relations formed from generalized tuples are closed under the operations of relational algebra.

In his doctoral thesis [Nelken, 2001] suggests the design of a natural language interface to temporal databases, based on translating natural-language temporal questions into SQL/Temporal, which is a recent temporal database query language .The interface is based on two stage translation process, where in first stage question are translated into a two-sorted first-order logic over temporal interval and in second stage logical formulae is translated into SQL/Temporal.

In other paper [Nelken, 2001] the author continues his work in temporal databases and presents a Natural Language Interface to temporal database controlled by novel based on translating natural language questions into temporal database query language, which is done using Type-Logical Grammar framework. For example consider the NL question: During which year did Mary work in marketing?

The meaning of the sentence is constructed as:

$(\text{year}(I) \wedge \exists J\, (\text{work}(\text{mary}, \text{marketing}, J) \wedge J \subseteq \text{past} \wedge J \subseteq I))$

Which can be translated into the following SQL/Temporal query:

```
NonSequenced Validtime
Select distinct a0.c As c1
From work' As a1.year' As a0
Where Validtime(a0) contains
Validtime (a1)
And a1.c1 = 'mary'
And A1.c2 = 'marketing'
And period (TimeStamp 'beginning', TimeStamp 'now') contains Validtime (a1)
```

Coverage involving temporal databases in Natural-language interfaces

Years	Authors	Work
1983	Clifford and Warren	Were first to incorporate a concept of time in databases
1983	Hirst	Proposed a approach to semantic interpretation based on the Montague semantics.
1988	Clifford	Examines the connection between the semantic of historical database and the semantic of Natural language querying through the semantics of time

1988	Hinrichs	Explains how logical semantics for temporal expressions provide sufficient representations for natural-language inputs to an interface.
1990	Claire	Describes implementation of formal semantics as in Keena and Faltz
1990	Kabanza, Stevenne and Wolper	Presents framework for describing, storing and reasoning about infinite temporal information.
1998	Androutsopoulos, Ritchie and Thanisch	A good introductory paper on temporal database
2001	Nelken	Describes the design of natural-langauge interface to temporal database based on translating natural-langauge temporal questions to SQL/temporal

Table: Coverage of NL interfaces to temporal databases

Conclusion:

There hasn't been much work done in recent years on the use of Montague semantics in natural-language database query processing. Since the development of Montague Grammar a few new semantic theories [e.g., Groenendijk and Stokhof, 1991] have been developed either to augment Montague Grammar itself or as alternate theories to deal with some problem not dealt within the original definitions. One of the researchers in Computer Science Michael Beeson stated ' I still think Montague semantics could be developed further, but as far as I know, those who are doing natural-language processing aren't using it.'

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

Appendix-i

Bibliography

[1] Androutsopoulos I., Ritchie G.D., Thanisch P. Time, Tense and Aspect in Natural Language Database Interfaces. Natural Language Engineering, 4(3), pp. 229-276, Cambridge Univ. Press, Sept. (1998)

[2] Androutsopoulos I., Ritchie G. D., and Thanisch P. Experience using tsql2 in a natural language interface. In: Proceedings of the International Workshop on Temporal Databases, Zurich, pp. 113-132 September (1995).

[3] Androutsopoulos, I. and Ritchie G.D. and Thanisch P. Natural Language Interfaces to Databases--an introduction. Journal of Language Engineering, Vol 1, No 1, pp. 29-81, (1995)

[4] Bainbridge R. I. , Montagovian Definite Clause Grammar , Proc. of the 2nd EACL, Geneva, Switzerland, pp. 25-34, 1985

[Keywords: Compositional Semantics, Definite Clause Grammar, Friedman Warren Algorithm, Intensional Logic, Montague Grammar, Natural Language Processing, PROLOG]

[5] Bernardi R. and Moot R., Generalized Quantifiers in declarative and interrogative sentences. In J. Bos and M. Kohlhase (eds.) Logic Journal of IGPL Vol. 11, N. 4, July (2003).

[6] Bertossi L. E., Katona G., Schewe K. D., Thalheim B. Semantics in Databases, Second International Workshop, Dagstuhl Castle, Germany, January 7-12, 2001, Revised Papers. Springer 2003

[7] Bettini C., Wang S. X., Bertino E., Jajodia S. Semantic assumptions and query evaluation in temporal databases, ACM SIGMOD Record, v.24 n.2, p.257-268, May 1995

[8] Blackburn P. and Bos J. Computational Semantics for Natural Language. Course Notes for NASSLLI, Indiana University, (2003)

[9] Bos J. and Gabsdil M. First-Order Inference and the Interpretation of Questions and Answers. Communication Research Centre (HCRC).

[10] Chan S.W.K. Semantic Parsing as an Energy Minimization Problem, IEEE International Conference on Intelligent Processing Systems, Beijing, October 1997.

[11] Clifford J., Warren D. S., Formal Semantics for time in databases, ACM Transaction on Database Systems (TODS), v.8 n.2 p.214-254, June 1983

[12] Clifford J. Natural Language Querying of Historical Databases, Computational Linguistics 14(4), 1988

[13] Clifford J. Formal Semantics and Pragmatics for Natural Language Querying. Cambridge University Press, 1990

[14] Cimiano P. Translating Wh-Questions into F-Logic Queries In: Proceedings of 2nd CoLogNET-ElsNET Symposium. 2003

[15] Copestake A. and Jones K.S. Natural Language Interfaces to Databases. Knowledge Engineering Review, Volume 5, Number 5, 1990.

[16] Copestake A. and Jones K. S. Inference in a Natural Language Front End for Databases. University of Cambridge Computer Laboratory Technical report No. 163, (1990)

[17] Creary L. G. and Pollard C. J. A computational semantics for natural language. In Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, pages 172--179, (1985).

[18] Dalmas T., Leidner J. L., Webber B., Grover C., and Bos J. Generating annotated corpora for reading comprehension and question answering evaluation An EACL workshop, April (2003).

[19] Da-Silva Julia, R.M., Seabra J.R., Siqueira I.S., An intelligent parser that automatically generates semantic rules during syntactic and semantic analysis, Systems, Man and Cybernetics, 1995. 'Intelligent Systems for the 21st Century'., IEEE International Conference on , Volume: 1 , 22-25 Pages:806 - 811 vol.1 Oct. (1995)

[20] Delmonte R. Getaruns: a hybrid system for summarization and question answering. An EACL workshop, April (2003).

[21] Demers N.P. A Lexicalist Approach to Natural-Language Database Front-Ends. Master's Thesis, University of Ottawa (1996)

[22] Dowty D., Wall R., and Peters S. Introduction to Montague Semantics, Dordrecht, Holland: Reidel, (1981).

[23] Duclaye F., Yvon F., and Collin O. Learning paraphrases to improve a question-answering system. An EACL workshop, April (2003).

[24] Eijck J. V. The proper treatment of context in NL. In Paola Monachesi, editor, Computational Linguistics in the Netherlands (1999)

[25] Filipe P. P. and Mamede N. J. Databases and Natural Language Interfaces. V Jornada de Engenharia de Software e Bases de Dados (JESBD'2000), Valladolid, Spain, November (2000)

[26] Fox C., Lappin S. and Pollard C. First-Order, Curry-Typed Logic for Natural Language Semantics. in S. Wintner (ed.), Proceedings of the Seventh Workshop on Natural Language Understanding and Logic Programming, Copenhagen, pp. 175-192. (2002)

[27] Frost R. A. and Launchbury J. Constructing natural language interpreters in a lazy functional language. The Computer Journal, 32(2): 108--121, April (1989)

[28] Frost R. A. and Saba W. S. A database interface based on Montague's approach to the interpretation of natural language. International Journal of Man-Machine Studies, 33(2): 149-176, (1990).

[29] Frost R. A. and Boulos P. An Efficient Compositional Semantics for Natural-Language Database Queries with Arbitrarily-Nested Quantification and Negation. Lecture Notes In Computer Science Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, pp. 252 – 267, (2002)

[30] Gaasterland T., Godfrey P. and Minker J. An Overview of Cooperative Answering. Journal of Intelligent Information Systems", vol 1, no. 2, pp. 123-157, (1992).

[31] Gabsdil M. Interpreting Questions and Answers in a Prototype Dialogue System. Master's thesis, Universität des Saarlandes, Saarbrücken. (2001)

[32] Gadia S. K. A Homogeneous Relational Model and Query Languages for Temporal Databases. ACM Trans. Database System 13(4): 418-448(1988)
[Keywords: Historical database, relational calculus, relational model, temporal data, temporal databases, time, tuple calculus]

[33] Greenwood M. and Gaizauskas R. Using a named entity tagger to generalise surface matching text patterns for question answering. An EACL workshop, April (2003).

[34] Groenendijk, J. & Stokhof, M., 1989, `Dynamic predicate logic', Amsterdam: ITLI, to appear in Linguistics and Philosophy (1989)

[35] Groenendijk J. and Stokhof M. Dynamic Montague Grammar. Faculty of Mathematics and Computer Science, Roeterssraat, Amsterdam, Holland (1990)

[36] Gunter C. A. Semantics of Programming Languages: Structures and Techniques. Foundations of Computing. MIT Press, (1992).

[37] Hardt D. Centering in dynamic semantics. In COLING-96. Copenhagen, (1996).

[38] Hasting J. D. Design and Implementation of a Speech Recognition Database Query System, M.S. Department, University of Wyoming (1991)

[39] Hausser R., Database semantics for natural language. Artificial Intelligence, Vol. 130, Issue 1, pp. 27-74, July (2001)

[40] Hausser R. "Spatio-Temporal Indexing in Database Semantics," in A. Gelbukh (ed)., 2001

[41] Hinrichs E. W. Tense, Quantifiers, and Contexts, Computational Linguistics, Vol. 14 No. 2, June 1988

[42] Hirst G. A Foundation for semantic interpretation, Proceedings of the 21st Annual Meeting, Association for Computational Linguistics, Cambridge, Mass., June 1983, 64--73.

[43] Hirst G. Semantic interpretation and the resolution of ambiguity. Studies in Natural Language Processing, Cambridge University Press. (1987)

[44] Introduction to Computational Linguistics Lecture Notes, Natural Language Processing at Massachusetts Institute of Technology, Spring (2003)

[45] Kabanza F., St'evenne J.M., and Wolper P. Handling infinite temporal data. In Ninth ACM Symposium on Principles of Database Systems, pages 392--403, Nashville, Tennessee, Apr. 1990.

[46] Kalita J. K, Jones M. L, McCalla G. I. Summarizing natural language database responses. Computational Linguistics, Vol. 12 Issue 2, April (1986).

[47] Käppel D. Morphological oriented and Fault tolerant Recognizing of template based natural Language Questions on combinatorial Question Space (2002).

[48] Katz B. and Lin J. Selectively Using Relations to Improve Precision in Question Answering. Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering, April (2003).

[49] Knowles S. A Natural Language Database Interface for SQL-Tutor. Hours Project, 1999

[50] Landsbergen S. P. J. Adaptation of Montague Grammar to the Requirements of Question-answering, Proceedings of the 8th conference on Computational linguistics, Tokyo, Japan, pp 211-212, 1980

[51] Lang F. M., Hirschman L. Improved portability and parsing through interactive acquisition of semantic information, Proceedings of the second conference on Applied natural language processing Austin, Texas Pages: 49 - 57, (1988)

[52] Lapalme, G. and Lavier, F. (1990) Using a functional language for parsing and semantic processing. Publication 715a, Department d'informatique et recherché operationelle, Universite de Montreal.

[53] Lappoon R. T. Integrating Statistical and Relational Learning for Semantic Parsing: Applications to Learning Natural Language Interfaces for Databases. University of Texas, Department of Computer science, (2000)

[54] Lappoon R. T. and Raymond M. Automated Construction of Database Interfaces: Integrating Statistical and Relational Learning for Semantic Parsing. Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 133-141, Hong Kong, October, (2000)

[55] Litkowski, K. C. Question-Answering Using Semantic Relation Triples, in Voorhees, E. M. and Harman, D. K. (eds) Information Technology: The Eighth Text REtrieval Conference (TREC-8), NIST Special Publication 500-246. Gaithersburg, MD: National Institute of Standards and Technology, pp. 349-56, (2000).

[56] Main M. G., Benson D. B. Denotational Semantics for ``Natural" Language Question-Answering Programs. American Journal of Computational Linguistics 9(1): 11-21 (1983)

[57] Malinowski J. Montague style semantics for illocutionary logic. Matematyka V, Joanna Grygiel ed., Proceedings of Czestochowa Pedagogical University, pp. 76 – 82, (1997)

[58] Marjorie T., and Burger J. Problems in natural language interface to DBMS with examples from EUFID. ACL Proceedings, Conference on Applied Natural Language Processing, pp. 3—16, (1983)

[59] Miller S., Fox Heidi, Ramshaw L., Weischedel R. A novel use of statistical parsing to extract information from text , Source ACM International Conference Proceeding Series archive, Proceedings of the first conference on North American chapter of the Association for Computational Linguistics table of contents
Seattle, Washington pages: 226 - 233, (2000)

[60] Mollá D., Schwitter R., Rinaldi F., Dowdall J., and Hess M. NLP for answer extraction in technical domains. An EACL workshop, April (2003).

[61] Montague R. The proper treatment of quantification in ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, Approaches to Natural Language. Proceedings of the 1970.

[62] Mosny M., Semantic Information Preprocessing for Natural Language Interfaces to Databases. Meeting of the Association for Computational Linguistics, pp. 314-316, (1995)

[63] Muskens R. Meaning and Partiality. European Association for Logic, Language and Information (Folli), Studies in Logic, Language and Computation, 1995.

[64] Nelken R. and Francez N., Bilattices and the Semantics of Natural Language Questions, Technical Report LCL 9801, Laboratory for Computational Linguistics, the Technion. (1998)

[65] Nelken R. and Francez N. The Algebraic Semantics of Interrogative NPs. The Algebraic Semantics of Interrogative NPs. Journal Grammars, Vol. 3, N 2/3, pages 259-273, (2000)

[66] Nelken R. and Francez N. Querying Temporal Databases Using Controlled Natural Language. In proceedings of Coling (2000)

[67] Nelken R. Questions, Time and Natural Language Interfaces to Temporal Databases. PhD thesis (2001)

[68] Nyberg, E., Mitamura T., Carbonell J., Callan J., Thompson K. C., Czuba K., Duggan M., Hiyakumoto L., Hu N., Huang Y., Ko J., Lita L., Murtagh S., Pedro V. and Svoboda D., The JAVELIN Question-Answering System at TREC 2002, Proceedings of TREC 11, November (2002).

[69] Onet A., Doina T. Intensional Logic Translation for Quantitative Natural Language Sentences. (colaborare cu A.Onet), Studia Universitatis "Babes-Bolyai", Seria Informatica, no1, pp 41-54, (2001)

[70] Pankowski T. Semantics of approximate answers in cooperative database systems. Proc. of Int. Conf. on Computational Intelligence on Modeling, Control and Automaton, CIMCA' 99, Vienna, February 17-19, (1999)

[71] Partee H. Formal semantics and the lexicon. Lecture Notes Formal Semantics, Lecture 4, RGGU, March 7 (2003)

[72] Perez, R.D. Constructive semantics for extensional PTQ, Proceedings of the Fourth Mexican International Conference on, Pages: 33 – 39 8-12 Sept. 2003

[73] Popescu M. A., Etzioni O. and Kautz H., Towards a Theory of Natural Language Interfaces to Databases. IUI (2003)

[74] Ranta A. A database query system based on GF (Grammatical Framework). XRCE Grenoble June (1999).

[75] Rayner M., Abductive Equivalential Translation and its application to Natural Language Database Interfacing. Ph.D. thesis, Royal Institute of Technology, Stockholm, September (1993).

[76] Reeker, L. H. The computational study of language acquisition. In Yovits, M., & Rubinoff, M. (Eds.), Advances in Computers, Vol. 15, pp. 181--237. Academic Press, New York. 1976

[77] Reis P., Mamede N., Matias J. Edite -- A Natural Language Interface to Databases: a New Dimension for an Old Approach in "Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism", ENTER' 97, Edinburgh, Scotland (1997).

[78] Rosner M. and Johnson R. (edited by). Review of Computational Linguistics and Formal Semantics, Cambridge University Press, pp. 321 (1992).

[79] Ryan K. L., Root R. and Olawsky D., Application-Specific Issues in Natural Language Interfacer Development for a Diagnostic Expert System. Proc. of the Second Conference on Applied Natural Language Processing,Austin, TX,pp. 109-114, (1988).

[80] Schmidt D. A., Denotational Semantics: A Methodology for Language Development, Allyn and Bacon, Newton, MA, (1986).

[81] Scott M., Stallard D., Bobrow R. and Schwartz R., A Fully Statistical Approach to Natural Language Interfaces. Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, Morgan Kaufmann Publishers, San Francisco, pp. 55-61, (1996)

[82] Shan C. Monads for natural language semantics. Proceedings of the 2001 European Summer School in Logic, Language and Information student session, ed. Kristina Striegnitz, pp. 285-298 (2001)

[83] Shan C. A variable-free dynamic semantics. Proceedings of the 13th Amsterdam Colloquium, ed. Robert van Rooy and Martin Stokhof, pp. 204-209 (2002)

[84] Siklossy, L. Natural language learning by computer. In Simon, H. A., & Siklossy, L. (Eds.), Representation and meaning: Experiments with Information Processing Systems. Prentice Hall, Englewood Cliffs, NJ. 1972

[85] Stratica N., Kosseim L. and Desai B.C. (2002) A Natural Language Processor for Querying Cindi . In Proceedings of International Conference Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet (SSGRR 2002s), L'Aquila, Italy July (2002).

[86] Stoy J., Denotational Semantics: the Scott-Strachey approach to Programming Language Theory, MIT Press, (1977).

[87] Teskey F. N. Enriched knowledge representation for information retrieval. Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval November (1987)

[88] Thomason R. H. Knowledge Representation and Knowledge of Words. Lexical Semantics and Knowledge Representation: Proceedings of a Workshop Sponsored by the Special Interest Group on the Lexicon of the Association for Computational Linguistics, Association for Computational Linguistics", Somerset, New Jersey, pp.1-8, (1991)

[89] Thomason R. H. Non-Monotonic Formalisms for Natural Language Semantics. Linguistics Department, University of Pittsburgh, Pittsburgh. August (2002).

[90] Thomason R. H. Formalizing the Semantics of Derived Words. Linguistics Department, University of Pittsburgh (2001)

[91] Tore A. The Understanding Computer. Lecture notes on Natural Language Interfaces course at NTNU, (2002)

[92] Vanderhoeft C. An Implementation Of Formal Semantics In The Formalism Of Relational Databases. COLING 1990: 377-382

[93] Wallace M. Communicating with Databases in Natural Language. Horwood, Chichester, (1984).

[94] Warren D. S. and Friedman J. Using Semantics in Non-Context-Free Parsing of Montague Grammar. American Journal of Computational Linguistics Vol 8, N 3-4, pp. 123-138, (1982)

[95] Winiwarter W. and Kambayashi Y. DOA - The Deductive Object-Oriented Approach to the Development of Adaptive Natural Language Interfaces Abstract, British National Conference on Databases, pp. 137-138, (1997)

[96] Woods W., Kaplan R., and Webber B., The Lunar Sciences Natural Language Information System: Final Report, BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, 1972.

[97] Yates A. and Etzioni O. and Weld D. Reliable natural language interfaces to household appliances. IUI-03, (2003).
[Keywords: Natural language interface, database, appliance, planner.]

[98] Yonezaki N. and Enomoto H. Database system based on intensional logic, COLING-80", pp. 220-227, (1980)

[99] Zaenen A. and Uszkoreit H. Language Analysis and Understanding. In Joseph Cole Ronald A., Mariani, Hans Uszkoreit, Annie Zaenen, and Victor
Zue, editors, Survey of the State of the Art in Human Language Technology,
chapter 3. Cambridge University Press, (1996).

[100] Zelle J. M. and Mooney R. J. Learning to Parse Database Queries Using Inductive Logic Programming, Proceedings of the 14th National Conference on Artificial Intelligence, AAAI Press/MIT Press, Portland, OR, pp. 1050-1055, (1996).

[101] Zweigenbaum Pierre, Question answering in biomedicine Natural Language Processing for Question Answering. An EACL workshop, April (2003).

[102] http://www.ercim.org/publication/Ercim_News/enw26/zampolli.html

[103] http://www.linguistlist.org/issues/2/2-524.html#1

Appendix-ii

Annotated Bibliography

[1] Androutsopoulos, I. and Ritchie G.D. and Thanisch P. Natural Language Interfaces to Databases--an introduction. Journal of Language Engineering, Vol 1, No 1, pp. 29-81, (1995)
[Androutsopoulos, Ritchie and Thanisch's paper in 1995, is an introductory paper on natural language interfaces to databases which talks about history of NLIDBs, some advantages and disadvantages of NLIDBs and also compares NLIDBs to formal query languages, form-based interfaces and graphical interfaces. The first natural-language interfaces to databases appeared in the late sixties and early seventies According to the authors, the best-known NLIDB at that period was LUNAR [Woods, 1972], a natural-language interface to a database containing chemical analyses of moon rocks. Some other NLIDBs developed at that time were RENDEZVOUS, LADDER, PLANTES and PHILIQA1 [description of and references for all of these systems can be found in Androutsopoulos, Ritchie and Thanisch, 1995].]

[2] Androutsopoulos I., Ritchie G. D., and Thanisch P. Experience using tsql2 in a natural language interface. In: Proceedings of the International Workshop on Temporal Databases, Zurich, pp. 113-132 September (1995).

[In this paper, the authors talk about using a language called TSQL2 in a natural-language interface. The paper focuses on the TSQL2 in a natural-language interface for temporal databases and also in some point on the semantics of TSQL2. For example the question " On how many Mondays was John at University of Windsor in 2000? " can be expressed as:

```
Select Snapshot Count (Distinct d.*)
From &year_month_day (Period) As d, student_visits (Period) as t
Where d.year = 2000
AND d.day_name= "Monday"
AND VALID(t) OVERLAPS  VALID(d)
AND t.student='John'
AND t.school='University of Windsor'
```

Assuming that the calendric table and year_month_day and student_visits tables are available.]

[3] Androutsopoulos I., Ritchie G.D., Thanisch P. Time, Tense and Aspect in Natural Language Database Interfaces. Natural Language Engineering, 4(3), pp. 229-276, Cambridge Univ. Press, Sept. (1998)

[In their paper (which is a good introductory paper on temporal Databases) in 1998, Androutsopoulos, Ritchie and Thanisch, suggest a new framework for constructing natural language interface for temporal database as at that point of time most of the natural language database interfaces designed had very limited facilities for manipulating time-dependent data and didn't support temporal linguistic mechanisms. The authors refer to the work done by [Clifford and Warren, 1983] in temporal databases.]

[4] Bernardi R. and Moot R., Generalized Quantifiers in declarative and interrogative sentences. In J. Bos and M. Kohlhase (eds.) <u>Logic Journal of IGPL</u> Vol. 11, N. 4, July (2003).

[In this paper authors describe a logical system, which has the ability to compute the semantics of both declaratives and interrogative sentences. For example, the author analysed sentences such as:

Q: Did Tarantino direct Titanic?	$\lambda Y(Y((\text{direct titanic})\ \text{tarantino}))$
A: No	$\lambda p\ \neg p$
Q(A) By twice beta-reduction	$\neg((\text{direct titanic})\ \text{tarantino})$

The authors also considered "what" as an example:

Q: what did Cameron direct?	$\lambda Y(Y\lambda\ x((\text{direct } x)\ \text{cameron}))$
A: Titanic	$\lambda P\ P(\text{titanic})$
Q(A):By twice beta-reduction	$((\text{direct titanic})\ \text{cameron})$]

[5] Blackburn P. and Bos J. Computational Semantics for Natural Language. Course Notes for NASSLLI, Indiana University, (2003)

[The paper [Blackburn and Bos, 2003] gives a good introduction to the Computational Semantics of Natural Language. This paper introduces the basics of natural-language semantics. It describes first-order logic, lambda calculus and underspecified representations such as scope ambiguities (e.g. John advertised one house on every street) and Montague's approach.]

[6] Clifford J., Warren D. S., Formal Semantics for time in databases, ACM Transaction on Database Systems (TODS), v.8 n.2 p.214-254, June 1983

[In this paper Clifford and Warren has discussed that formal logic has made important contribution in understanding and specification of the semantics of database. Authors showed that relational database model could be extended to incorporate the concept of historical relations as well as database and also shown how ILs (reformulated IL to include *s* as a basic type) can provide a semantic theory for this database concept. In this paper the authors also suggested as interesting aspect in defining the translation of English questions into ILs, where the authors interpret English statements as database commands. For example, the authors interpreted the statement ' John earns 30k' as a command to record this as a fact in the database with the time-stamp taken from the system clock when made by an authorized user.]

[7] Clifford J. Natural Language Querying of Historical Databases, Computational Linguistics 14(4), 1988

[In this paper the author examines the connection between the semantics of historical databases and the semantics of natural language querying and through a common view of the semantics of time link them together. The author demonstrated the use of QE-III, a formally defined English database query language whose semantics and pragmatic theory are based on a Montague type semantics and discussed the issues on providing both semantics and pragmatic interpretation for question within a model-theoretic framework. For example questions in English Query Language QE-III can be handled in the following way:

Who is Peter's manager?

which can be interpreted as:

$\lambda u\, \exists x\, [\, MGR'(now)(x) \wedge x(now) = u \wedge AS\text{-}1(Peter,x)]\,]$

[8] Claire G. Dynamic Semantics and VP-Ellipsis. JELIA 1990: 251-266, (1990)

[[Claire, 1990] describes the implementation of formal semantics as described in Keena and Faltz *Boolean Semantics for Natural Language for Natural Language.* The author claims that his implementation avoids the intermediate step of translating Natural Language into a formal language such as an extended version of predicate calculus which makes his implementation free of the problems related to the syntax of such a language like binding the variable and resolving scope ambiguities however which has disadvantage that every denotation (i.e. semantic value) requires to be explicitly and accurately represented in a database.]

[9] Cimiano P. Translating Wh-Questions into F-Logic Queries In: Proceedings of 2nd CoLogNET-ElsNET Symposium. 2003

[In this paper the author presents a approach to map natural-language wh-questions into F(rame)-logic queries based on Montague-style compositional semantics where semantic representation is constructed on basis of Lexicalized Tree Adjoining Grammar LTAG-style derivation tress. For example, who owns a company? can be translated as follows:

$?\text{-}\exists\, Y\ Y\text{: company} \wedge X\, [\text{ own} \rightarrow Y\,]\, .\,]$

[10] Demers N.P. A Lexicalist Approach to Natural-Language Database Front-Ends. Master's Thesis, University of Ottawa (1996)
[In this thesis the author introduces a lexicalist approach, which is based on unification grammars to database NLI's along a small-scale example. The author claims that the solution proposed to this approach is not only feasible but also provides reasonable complexity and processing time for unambiguous words and expressions]

[11] Filipe P. P. and Mamede N. J. Databases and Natural Language Interfaces. V Jornada de Engenharia de Software e Bases de Dados (JESBD'2000), Valladolid, Spain, November (2000)
[In this paper the authors mainly focus to the translation stage, translating user questions first into a logic language and then into Structured Query Language (SQL) , which is that processed by a database management system to return answer to the question.]

[12] Duclaye F., Yvon F., and Collin O. Learning paraphrases to improve a question-answering system. An EACL workshop, April (2003).

[Duclaye and Yvon, 2003] discussed several methods on how to improve question-answering systems. The authors presented an unsupervised methodology starting with one single positive learning example for automatically learning paraphrases and which is able to filter out the invalid potential paraphrases extracted during the acquisition steps using an EM-based validation. The authors claim that these paraphrases are useful to improve the results of their question-answering system.

[13] Frost R. A. and Launchbury J. Constructing natural language interpreters in a lazy functional language. The Computer Journal, 32(2): 108--121, April (1989)

[In this paper the authors describe how in a functional programming language, language parsers and interpreters can be implemented. Frost and Launchbury refer to the book by Dowty, Wall and Peters (1981) but make no reference to any previous work on the use of Montague semantics in database query processing. It appears that Frost and Launchbury were amongst the first to use Montague semantics in database query processing.

It would also appear that Frost and Launchbury were the first to use a set-theoretic based implementation of Montague semantics. For example instead of interpreting 'every' like in Montague as:

$$[\text{every}] = \lambda p \lambda q\ [\ \forall p(x) \rightarrow q(x)\ ]$$

Frost and Launchbury used:

$$[\text{every}]_{FL} = \lambda p\ \lambda q\ \ p \subseteq q\ ]$$

[14] Frost R. A. and Saba W. S. A database interface based on Montague's approach to the interpretation of natural language. International Journal of Man-Machine Studies, 33(2): 149-176, (1990).
[In this paper the authors implement some of the concepts of Richard Montague that can be used in Natural Language Interface to databases .The database interface is implemented in a higher-order functional programming language and the semantic calculation is achieved through higher-order functional application.]

[15] Frost R. A. and Boulos P. An Efficient Compositional Semantics for Natural-Language Database Queries with Arbitrarily-Nested Quantification and Negation. Lecture Notes In Computer Science Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, pp. 252 – 267, (2002)
[In this paper the authors describes implementation of a compositional semantics based on a set-theoretic version of Montague semantics for a small Natural Language Query processor. A compositional semantics for phrases that include the word 'no' is developed based on an extended set theory in which 'negative' phrases denote infinite sets represented in complement form.]

[16] Groenendijk J. and Stokhof M. Dynamic Montague Grammar. Faculty of Mathematics and Computer Science, Roeterssraat, Amsterdam, Holland (1990)

[In this paper the authors propose a new logical system as the semantic component of a Montague –Style grammar that extends the compositionality of DPL (dynamic predicate logic) to the subsentential level. In DLP (Dynamic Predicate Logic) a sentence such as "Every farmer who owns a donkey beats it" can be translated into the formula as follows:

$\forall x[[\text{farmer}(x) \wedge \exists y [\text{donkey}(y) \wedge \text{own}(x, y)]] \rightarrow \text{beat}(x ,y)]$

In DLP the above translation is equivalent to :

$\forall x \forall y [[\text{farmer}(x) \wedge \text{donkey}(y) \wedge \text{own}(x, y)] \rightarrow \text{beat}(x ,y)]$

It is a continuation of their work [Groenendijk and Stokhof, 1989] on dynamic predicate logic.]

[17] Hardt D. Centering in dynamic semantics. In COLING-96. Copenhagen, (1996).

[In this paper the author presents a dynamic framework, a dynamic logic system, with extensions for the discourse center (a distinguished discourse entity that is the topic of a discourse), VP ellipsis (Verb Phrase ellipsis) and paycheck pronouns. (A paycheck pronoun is a pronoun, which exhibits sloppy identity, for example "Smith spent his paycheck. Jones saved it.". Here "it" is not an ordinary bound pronoun, nor is it an ordinary free pronoun.]

[18] Hausser R., Database semantics for natural language. Artificial Intelligence, Vol. 130, Issue 1, pp. 27-74, July (2001)

[In this paper authors describes database semantics as a declarative model of a cognitive agent which is called a SLIM machine and which functionally integrates the procedures of Natural language interpretation, conceptualization and production. No one appears to have referred to this work at this point of time.]

[19] Hausser R. "Spatio-Temporal Indexing in Database Semantics," in A. Gelbukh (ed)., 2001

[In this paper Hausser presents a new approach where the spatio-temporal location of propositional content is not specified precisely within a Cartesian system of space and time coordinates instead it is characterized cognitively by the order of direct observations

entering the database of a cognitive agent. The sequence of propositions which serves as the spatial landmarks are structured by observations of the environment and temporal landmarks are structured by observations of cyclical events. In database semantics, like all other inference, which navigate through the concatenated propositions, spatio-temporal inferences are handled.]

[20] Hinrichs E. W. Tense, Quantifiers, and Contexts, Computational Linguistics, Vol. 14 No. 2, June 1988

[In [Hinrichs, 1988], the author argued that a logical semantics for temporal expressions could provide sufficient representations for natural-language inputs to an interface such as JANUS, a natural language understanding and generation system under joint development by BBN Labs and ISI. The author demonstrated that if narrow scopes are given to tense quantifiers that will enable to provide adequate scope relation with respect to natural-language quantifiers and to interpret such NPs relative to a given discourse context. The author also demonstrated that how in English the narrow scope of tense results in a fully compositional syntax and semantics of tensed sentences.]

[21] Hirst G. A Foundation for semantic interpretation, Proceedings of the 21st Annual Meeting, Association for Computational Linguistics, Cambridge, Mass., June 1983, 64--73.

[In [Hirst, 1983] the author proposed a new approach to semantic interpretation based on the semantic formalism of Richard Montague. In this approach author claim that their semantics are compositional by design and strongly typed like Montague and they replace Montague's semantic objects and truth conditions with the elements of the frame language Frail and added a word sense and case slot disambiguation system. They claim that their approach to semantic interpretation is superior to previous approaches. For example a single noun phrase the book can be interpreted as (the ?x (book ?x)), which is a Frail frame statement. And a descriptive adjective correspond to a lot-filler pair from example red is represented by (color=red), so the red book would have semantic interpretation (the ?x (book ?x(color=red))). Similarly the sentence "Nadia bought the book from a store in the mall " will be interpreted as

(a ?u (buy ?u (agent = (the ?x (thing ?x (propername= "Nadia"))))
(patient = (the ?y (book ?y))) (source = (a ?z (store ?z (location =
(the ?w (mall ?w)))))))]

[22] Kabanza F., St'evenne J.M., and Wolper P. Handling infinite temporal data. In Ninth ACM Symposium on Principles of Database Systems, pages 392--403, Nashville, Tennessee, Apr. 1990.
[In this paper the authors present a framework, which is an extension of classical relational database, for describing, storing and reasoning about infinite temporal information and this framework represents infinite temporal information by generalized tuples which are defined by linear repeating points and constraints on these points. Authors prove that relations formed from generalized tuples are closed under the operations of relational algebra.]

[23] Lappoon R. T. and Raymond M. Automated Construction of Database Interfaces: Integrating Statistical and Relational Learning for Semantic Parsing. Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 133-141, Hong Kong, October, (2000)
[In this paper the authors present a method for integrating statistical and relational techniques for the automated acquisition of NLI's from training examples. They also claim that their approach is more robust than a previous purely logical approach.]

[24] Main M. G., Benson D. B. Denotational Semantics for ``Natural'' Language Question-Answering Programs. American Journal of Computational Linguistics 9(1): 11-21 (1983)
[According to Main and Benson in 1983, denotational semantics can be used as a specification technique for question-answering programs & for implementation the principle of compiler design was suggested as principle of question answerer design.]

[25] Mosny M., Semantic Information Preprocessing for Natural Language Interfaces to Databases. Meeting of the Association for Computational Linguistics, pp. 314-316, (1995)

[In this paper the author propose an approach to extract constraints, which are explicitly or implicitly provided by a semantic part of the NLID, from the semantic description of the database domain and incorporate them into information directly accessible to the parser.]

[26] Nelken R. and Francez N., Bilattices and the Semantics of Natural Language Questions, Technical Report LCL 9801, Laboratory for Computational Linguistics, the Technion. (1998)

[In this paper authors propose a novel semantics theory of NL questions which is composed of a compositional translation method into a formal logical meaning representation language in a Montagovian framework.]

[27] Nelken R. and Francez N. The Algebraic Semantics of Interrogative NPs. The Algebraic Semantics of Interrogative NPs. Journal Grammars, Vol. 3, N 2/3, pages 259-273, (2000)

[In the paper "the algebraic Semantics of Interrogative NPs" authors Nelken and Francez, in 2000, suggest a new semantic interpretation of interrogative NPs, which play an important role in driving the interpretation of wh-questions such as " which women". The authors used a formal language called Intensional Logic with Questions (ILQ) which extends Montague's IL. The authors added two operators: the interrogative operator (?) used for yes/no questions and the binding interrogative operator (?x) used for constituent questions. For example, here the authors interpret the interrogative determiner "which" as:

$$[\ \text{Det which}\ ] = \lambda P \lambda Q.\ ?x\ (P(x) \wedge Q(x))$$

which is similar to the standard interpretation of the determiner "a" :

$$\text{a: } \lambda P \lambda Q\ \exists x\ (P(x) \wedge Q(x))$$

So the meaning of the sentence "which woman kissed John" can be interpreted as follows:

[Which woman kissed John]

= [INP which woman] ([$_{VP}$ kissed John])

= $\lambda Q.\ ?x\ (\text{woman}(x) \wedge Q(x))\ (\lambda y.\ \text{kiss}\ (y, \text{John}))$

= $?x\ (\text{woman}(x) \wedge \text{kiss}\ (x, \text{John}))$]

[28] Nelken R. and Francez N. Querying Temporal Databases Using Controlled Natural Language. In proceedings of Coling (2000)

[In this paper authors present Natural Language Interface to temporal database controlled by novel based on translating natural language questions into temporal database query language, which is done using Type-Logical Grammar framework.]

[29] Nelken R. Questions, Time and Natural Language Interfaces to Temporal Databases. PhD thesis (2001)

[In his doctoral thesis, Nelken suggest the design of a natural language interface to temporal databases, based on translating natural language temporal questions into SQL/Temporal, which is a recent temporal database query language .The interface is based on two stage translation process, where in first stage question are translated into a two-sorted first-order logic over temporal interval and in second stage logical formulae is translated into SQL/Temporal.]

[30] Nelken R. and Francez N. Querying Temporal Databases Using Controlled Natural Language. In proceedings of Coling (2000)

[In other paper [Nelken, 2000] the author continues his work in temporal databases and presents a Natural Language Interface to temporal database controlled by novel based on translating natural language questions into temporal database query language, which is done using Type-Logical Grammar framework. For example consider the NL question: During which year did Mary work in marketing?

The meaning of the sentence is constructed as:

$(\text{year}(I) \wedge \exists J\ (\text{work}(\text{mary}, \text{marketing}, J) \wedge J \subseteq \text{past} \wedge J \subseteq I))$

Which can be translated into the following SQL/Temporal query:

NonSequenced Validtime

Select distinct a0.c As c1

From work' As a1.year' As a0

Where Validtime(a0) contains

Validtime (a1)

And a1.c1 = 'mary'

And A1.c2 = 'marketing'

And period (TimeStamp 'beginning', TimeStamp 'now') contains Validtime (a1)]

[31] Onet A., Doina T. Intensional Logic Translation for Quantitative Natural Language Sentences. (colaborare cu A.Onet), Studia Universitatis "Babes-Bolyai", Seria Informatica, no1, pp 41-54, (2001)
[In this paper authors describe the fundamentals of intensional logic and introduce some methods for treating quantitative natural sentences. Authors split quantitative sentences in three categories: definite quantity sentences (e.g. "Four women cry"), indefinite quantity sentences (e.g. "Most women cry"), restrictive quantity sentences (e.g. "Maximum five children answer") and tried to translate them in to intensional logic.]

[32] Popescu M. A., Etzioni O. and Kautz H., Towards a Theory of Natural Language Interfaces to Databases. IUI (2003)
[In this paper authors introduce a theoretical framework, which is foundation for the fully implemented Precise NLI and proved that Precise guarantees to map each question to the corresponding SQL query, for a broad class of semantically tractable Natural Language Questions.]

[33] Ranta A. A database query system based on GF (Grammatical Framework). XRCE Grenoble June (1999).
[In his paper, Ranta describes a database query system based on Grammatical Framework which was demonstrated concerning a database of restaurants which runs in seven European languages and the system can be modified on various level like changing basic grammatical structure into other structure.]

[34] Reis P., Mamede N., Matias J. Edite -- A Natural Language Interface to Databases: a New Dimension for an Old Approach in "Proceeding of the Fourth International Conference on Information and Communication Technology in Tourism", ENTER' 97, Edinburgh, Scotland (1997).

[In this article the authors present the Edite system which is a Natural Language Interface to Database and the system explore the advantage of joining natural language processing with the expressiveness of graphical interfaces. Edite, a natural-language front-end for relational databases, is multi-lingual (Portuguese, French, English, Spanish). It is capable of answering written questions related to tourism by transforming them into SQL queries. The answer can be a list of resources, text, images or graphics depending of the questions. At present, the database contains 53000 tourism resources, arranged on 253 distinct types, which corresponds to 209 tables. This paper refers to the work done by [Androutsopoulos, Ritchie, Thanisch, 1993].]

[35] Scott M., Stallard D., Bobrow R. and Schwartz R., A Fully Statistical Approach to Natural Language Interfaces. Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics, Morgan Kaufmann Publishers, San Francisco, pp. 55-61, (1996)

[A fully-statistical approach to a natural-language interface, which consists of three stages of processing: parsing, semantic interpretation and discourse, is described in [Scott and David 1996]. All of the stages are modeled as a statistical process, which are integrated, resulting in an end-to-end system that maps input utterances into meaning-representation frames.]

[36] Shan C. Monads for natural language semantics. Proceedings of the 2001 European Summer School in Logic, Language and Information student session, ed. Kristina Striegnitz, pp. 285-298 (2001)

[In his paper, Shan characterizes the similarity between several semantics accounts for interrogatives, focus, intensionality, variable binding & quantifications by using monads. A monad is a structure from abstract algebra, category theory.]

[37] Shan C. A variable-free dynamic semantics. Proceedings of the 13th Amsterdam Colloquium, ed. Robert van Rooy and Martin Stokhof, pp. 204-209 (2002)
[In his paper, in 2002, Shan introduces a new concept variable free dynamic semantics, which means denotional semantics for Natural Language where meanings of constituents are updates to information states. Shan [2002] analyzed sentences such as "A man walks in the park. He whistles." For example, the author wrote e for the type of an individual, $e \rightarrow 1$ for the type of a property and $e \rightarrow e \rightarrow 1$ for the type of a two-place relation. So, the derivation of the sentence " A man walks in the park" is translated as follows:

$$A: (e \rightarrow 1) \rightarrow e = \lambda p.\ \{\ v \mid * \in p(v)\ \}$$

$$\text{Man}: e \rightarrow 1,\ \text{WITP}: e \rightarrow 1,\ \text{WITP}(\text{A}(\text{MAN})):1$$

And, whistles denotes some property WHISTLE: $e \rightarrow 1$ and "he" denotes

$$\text{HE}: e \blacktriangleright e = \lambda v.v$$

where $\blacktriangleright$ ("in") is a new binary type constructor where type $\sigma \blacktriangleright \tau$ is like $\sigma \rightarrow \tau$ in that they may have the same models, namely functions from σ to τ.
So now "He whistles" can be derived as follows:

$$g \blacktriangleright (\text{WHISTLE})\ (\text{HE}): e \blacktriangleright 1 = \lambda v: \text{WHISTLE}(v)$$

where $g \blacktriangleright$ is a type-shift operation such as

$$g \blacktriangleright : (\alpha \rightarrow \beta) \rightarrow (\sigma \blacktriangleright \alpha) \rightarrow (\sigma \blacktriangleright \beta) = \lambda f.\ \lambda v.\ \lambda s.\ f(v(s))]$$

[38] Thomason R. H. Formalizing the Semantics of Derived Words. Linguistics Department, University of Pittsburgh (2001)
[In this paper the authors propose an approach ,the logical approach, which they claim has never produced a very satisfactory account of word meaning but is successful in the semantic interpretation of syntactic structure. For example the natural way to define 'x is water soluble' is as follows:

If x were put in some water, then x would dissolve in the water.

The definition of 'water-soluble' is obtained by using eventualities in place of times (This formula uses more or less standard formalization techniques in event-centered semantics, for example [Push(e) ^ Past(e) ^ Pusher(e) = Charlie ^ Pushee(e) = Piano] is used to represent Charlie pushed the piano .)

.

$\forall x$ [water-soluble(x) $\leftrightarrow$ $\forall e_1 \forall y$ [put_in(e_1) $\wedge$ movee(e_1) = x $\wedge$ container(e_1) = y $\wedge$ water(y)] $\rightarrow$ $\exists e$ [Dissolving(e) $\wedge$ dissolvee(e) = x $\wedge$ medium(e) = y] $\wedge$ $\neg$Ab(e)] $\rightarrow$ $\exists$ e_2 [culmination(e) = e_2 $\wedge$ dissolved(e_2) $\wedge$ disolvee(e_2) = x $\wedge$ medium(e_2) = y]]

In words: *x* is water-soluble if and only if necessarily if an event *e*1 of putting *x* in a quantity of water occurs then *e*1 is the inception of a dissolving eventuality *e* involving the same *x* and quantity of water, which unless something abnormal about *e* will culminate in a state in which *x* is dissolved.]

[39] Warren D. S. and Friedman J. Using Semantics in Non-Context-Free Parsing of Montague Grammar. American Journal of Computational Linguistics Vol 8, N 3-4, pp. 123-138, (1982)

[According to Warren [1982], a complete, well-defined context in which these questions can be considered is provided by Montague grammar with its fully formalized syntax and semantics. [Warren, 1982] describes how to reduce the combinatorial explosion of syntactic ambiguity by using semantics during parsing in Montague grammar.]

[40] Yonezaki N. and Enomoto H. Database system based on intensional logic, COLING-80", pp. 220-227, (1980)

[According to Yonezaki and Enomoto, Richard Montague's Intensional Logic (I L), which describe semantics of natural language, can be useful to the theory of database in designing database systems which handles historical data and provide a formal description of database semantics.]

VITA AUCTORIS

Maxim Roy was born in 1978 in Leningrad, Russia. He graduated from Chittagong University School and College in Bangladesh in 1997. From there he went on to the University of Windsor where he obtained B.S.C.(Honours) in Computer Science in 2002. He is currently a candidate for the Master's degree in Computer Science at the University of Windsor and hopes to graduate in Winter 2005.