SEMANTIC QUERY LANGUAGE FOR TEMPORAL GENEALOGICAL TREES

Innopolis University

Thesis submitted to The Innopolis University in conformity with the requirements for the degree of Bachelor of Science.

presented by

Evgeniy Gryaznov

supervised by

Manuel Mazzara

Date

SEMANTIC QUERY LANGUAGE FOR TEMPORAL GENEALOGICAL TREES

To my parents and close relatives. This work would't be possible without their help.

Contents

List of Tables

List of Figures

Abstract

abstract ...

Chapter 1

Introduction

Chapter 2

Literature Review

The purpose of this chapter is to give a brief survey of academic literature with respect to our thesis. There are two major areas of Computer Science that are related to out research: Knowledge Representation (KR) and Natural Language Processing (NLP). We will examine them one by one and highlight some of the most important works.

2.1 Knowledge Representation

The field of Knowledge Representation is concerned with how the knowledge about our physical world can be stored, managed and utilized by computers. KR owes its existence to the more general field of Artificial Intelligence, which prompted the study of encoding information about the physical reality into an intelligent system in such a way that it can be used by that system to solve complex problems.

The main presupposition of the whole field of KR is that, in order for an intelligent agent to resolve a difficult problem, it needs an access to some form of a knowledge specific to a particular domain and that knowledge should be stored inside the agent. This presupposition is now widely known as the *Knowledge Representation Hypothesis*:

> Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge.

This original formulation of the hypothesis is due to Smith [1].

Brachman and Levesque outlined the hypothesis in their article *Expressiveness and tractability in Knowledge Representation and Reasoning* [2]. The authors argue that the trade-off between *expressiveness* of a knowledge-based system and its *tractability* (i.e. the ability to reason correctly) is intrinsic to every such system, and can only be partially solved. According to them, it is impossible to implement a knowledge-based system that will be both highly expressive and completely tractable.

Moreover, the whole enterprise of encoding knowledge directly into an agent is proven to be useful only in domain-specific applications, such as the famous *block world* [3] domain. A solution with at least partially hard-coded knowledge is tremendously difficult to scale. Since the dawn of Machine Learning, the plausibility of the hypothesis is continuously challenged. Indeed, it is questionable whether we can say about a neural network that it stores knowledge in the weights of its neurons. Thus, today the field of KR does not enjoy that much popularity because the main focus of the AI community has shifted to other areas, such as Deep Learning.

The most prominent advances in the KR field were the development of Description (Terminological) and Temporal logics and the formulation of the concept of Ontology. They all are of great importance to our research, so we will survey them one by one.

2.1.1 Ontologies

The word "ontology", derived from the Greek word meaning "the study of being", was unwarrantably borrowed by computer scientists from the namesake branch of philosophy concerned with the nature of reality. Despite being a useful coinage in informatics, not all philosophers are content with such state of affairs [4].

The term "ontology" in Computer Science refers to the mechanism by which reality is compartmentalized into a strictly-defined categories only to be read by machines later. According to Josephson et al. [5] an ontology comprises a body of knowledge about a particular domain of interest. However, we must distinguish between an abstract conceptualization of a particular domain and a concrete instantiation of it. The latter is usually implied when the plural word "ontologies" is used.

The typical ontology consists of:

1. A finite set of *concepts* (a.k.a. nodes, classes). Represents entities of a domain.

2. A finite set of *properties* (a.k.a. attributes, slots, roles). Represents what can be asked of a concept.

3. A finite set of *relationships* between concepts.

4. A finite set of logical *constraints*, which put the boundaries around what can and cannot be stored in an ontology.

At the first sight, the description of an ontology highly resembles that of a database. Indeed, it is true that every database can be seen as a special case of an ontology, but not vise-versa. As noted in [2], the power of ontologies lies not in what can be said in them, but exactly *what can be left unsaid*. For example, suppose we want to store the birth date of our grandfather, but all we know about him is that he won a medal fighting in WWII. Then we are forced, using a database, to left the `birth_date` field empty, thus loosing the knowledge of his

heroic deed. But, we can eloquently express this knowledge in some ontological lisp-like language as:

```
(set birth_date (father (father me)) (during WWI))
```

Later we can use this fact to reason about our ancestor more efficiently.

Ontologies find their natural application in the context of our thesis. Since the original formulation of a concept, a lot of software has been developed to manage ontologies, including such systems as Protege, In4j and others. These systems have already been heavily used in the variety of different fields.

For instance, Tan Mee Ting [6] designed and implemented a genealogical ontology using Protege and evaluated its consistency with Pellet, HermiT and FACT++ reasoners. He showed that it is possible to construct a family ontology using *Semantic Web* [7] technologies with full capability of exchanging family history among all interested parties. Hovewer, he did not adress the issue of navigating the family tree using kinship terms.

An ontology can be used to model any kind of family tree, but the problem arises when a user wants to query his relatives using kinship terms. No standard out-of-the-box ontology query language is able to articulate statements such as in our example above. Although an ontology can be tailored to do so, it is not in any way a trivial matter. Maarten Marx [8] addressed this issue, but in the different area. He designed an extension for XPath, the first order node-selecting language for XML.

Catherine Lai and Steven Bird [9] described the domain of linguistic trees and discussed the expressive requirements for a query language. Then they presented a language that can express a wide range of queries over these trees, and showed that the language is first order complete. This language is also an extension of XPath.

Artale et al. [10] did a comprehensive survey of various temporal knowledge representation formalisms. In particular, they analysed ontology and query languages based on the linear temporal logic LTL, the multi-dimensional Halpern-Shoham interval temporal logic, as well as the metric temporal logic MTL. They

note that the W3C standard ontology languages, OWL 2 QL and OWL 2 EL, are designed to represent knowledge over a static domain, and are not well-suited for temporal data.

Modelling kinship with mathematics and programming languages, such as LISP, has been an extensive area of research. Many people committed a lot of work into the field, including Bartlema and Winkelbauer, who investigated [11] the structure of a traditional family and wrote a simple program that assigns fathers to children. Their main purpose was to understand how this structure affects fertility, mortality and nuptiality rates. Although promising, small step has been made towards designing a language to reason about kinship. Also, their program cannot express temporal information.

Another prominent attempt in modelling kinship with LISP was made [12] by Nicholas Findler, who examined various kinship structures that exist in literature and combined them together to create a LISP program that can perform arbitrary complex kinship queries. Although his solution is culture-independent, he did not take the full advantage of LISP as a programming language, and because of that it is impossible to express queries which are not about family interrelations. In contrast, our system does not suffer from that restriction.

More abstract, algebraic approach was taken [13] by D. W. Read, who analysed the terminology of American Kinship in terms of its' mathematical properties. Specifically, he invented an algebraic system, a semigroup, isomorphic to the one particular type of kinship: American Kinship, which refers to the terminology used by English speakers. His algebra clearly demonstrates that a system of kin terms obeys strict rules which can be successfully ascertained by formal methods.

Periclev et al. developed [14] a LISP program called `KINSHIP` that produces the guaranteed-simplest analyses, employing a minimum number of features and components in kin term definitions, as well as two further preference constraints that they propose in their paper, which reduce the number of multiple componential models arising from alternative simplest kin term definitions conforming to one feature set. The program is used to study the morphological and phono-

logical properties of kin terms in English and Bolgarian languages.

According to many authors [13] [14] there have been many attempts to create adequate models of kinship based on mathematics, but these models are either lacking management of temporal information or use only a limited subset of their programming language. This is not the case with our system.

2.1.2 Temporal and Description Logics

The usage of logic in the field of KR is motivated by its excellence in such areas as mathematics and computer science in general. The early researches in KR saw the unharvested power of logic – especially first-order logic – as a main component in any intelligent system. Subsequent works showed that FOL can provide semantics for specific kind of KR structures: *frames* [15]. Later, Brachman and Levesque proved [2] that we do not need the *whole* FOL for that purpose, but only certain fragments of it. Moreover, different fragments of FOL have different expressive power and tractability. Thus, a research began under the label *terminological systems*, only to be later renamed to *Description Logic* when the main focus was shifted to the properties of underlying logical systems.

Description Logic finds its application in the context of this thesis as a natural formalism for family trees. However, as expressive as any DL can be, formulating the concept of time requires adding another modal operator. Any logic which handles time is knows as *temporal logic*. Philosophers have tried to put time into a coherent framework since Aristotle, in the 20th century mathematicians and computer scientists proposed various formalisms, among which was Allens' *interval algebra* [16] and the temporal logic of Shoham [17]. Thus, what we need in this thesis is an amalgamation of a description and temporal logic.

The first successful attempt at integrating two logics is due to Schmiedel [18]. He combined the DL in the tradition of KL-ONE [19], Shohams' [17] temporal logic and Allens' [16] algebra into one unifying framework. The main features of his formalism are the complete preservation of original DL and the use of

lisp-like syntax for expressing roles, concepts and time entires.

The application of temporal logic to graphs, relational databases and ontologies is also a heavily-invested subject. Barcelo and Lubkin examined [20] several temporal logics over unranked trees and characterized commonly used fragments of first-order (FO) and monadic second-order logic (MSO) for them. They also considered MSO sibling-invariant queries, that can use the sibling ordering but do not depend on the particular one used, and captured them by a variant of the μ-calculus with modulo quantifiers.

Alexander Tuzhilin and James Clifford defined [21] a temporal algebra that is applicable to any temporal relational data model supporting discrete linear bounded time. This algebra has the five basic relational algebra operators extended to the temporal domain and an operator of linear recursion. They showed that this algebra has the expressive power of a safe temporal calculus based on the predicate temporal logic with the "until" and "since" temporal operators.

Perry in his dissertation [22] highlighted that even in state-of-the-art ontological query languages, such as OWL, expressing the concept of time is an arduous task. He augmented the *Resource Description Framework* with temporal RDF graphs and extended the W3C-recommended SPARQL query language to support these new structures.

An adequate representation of time is the holy grail among researchers in the field of ontology development. Baratis et al. [23] designed and implemented *TOQL*: a high-level SQL-like language which is capable of expressing temporal queries. They motivate the need for such a language by noting that conveying the concept of time using classical languages, such as OWL, is proven to be difficult, although feasible. They also developed an application that supports translation and execution of TOQL queries on temporal ontologies combined with a reasoning mechanism based on event calculus.

2.2 Natural Language Processing

The main goal of Natural Language Processing (NLP) field is to invent, study and implement algorithms and techniques that help a computer understand an ordinary language, such as English, Russian, French or Swahili.

A lot of research in NLP is dedicated to the problem of querying a relational database in some natural language. Since the early developments, a substantial progress has been achieved. For instance, Jeremy Ferrero et al. proposed and implemented [24] a solution to query any database, irrespective of its' schema, in virtually any natural language. They showed that it supports more operations that most of the other translators. They tested their program on English and French languages.

Another similar attempt was made [25] by Norouzifard et al. They implemented an expert system using Prolog to transform a sentence in a natural language to SQL. Chaudhari [26] presented a light weight technique of converting a natural language statement into equivalent SQL statement.

Nelken et al. took [27] a step further and presented a novel controlled NL interface to *temporal databases*, based on translating NL questions into *SQL/Temporal*, a temporal database query language. They noted that their translation method is considerably simpler than previous attempts in this direction.

2.3 Conclusions

In this literature review we surveyed several major field in Computer Science. We showed that each of these fields has been advanced considerably over the last half-century, especially the domain of ontologies. However, we did not find a research that would satisfy all of the following criteria:

1. Employ either a temporal ontology or a temporal database to store knowledge of temporal family relations.

2. Propose a solution for effective navigation in a genealogical tree via kinship terms.

3. Design and implement a text parser for querying temporal ontologies in natural language.

Although there were articles that partially fulfill some of these requirements, none of them satisfied all. Moreover, since our solution is specifically tailored to work only on family trees, we can conclude that its' performance is better than any other general one. This entails novelty of our work.

Chapter 3

Methodology

3.1 General Considerations

The study of kin structures has its roots in the field of anthropology. Among the first foundational works was Henry Morgans' *magnum opus* "Systems of Consanguity and Affinity of the Human Family" [28], in which he argues that all human societies share a basic set of principles for social organization along kinship[1] lines, based on the principles of **consanguinity** (kinship by blood) and **affinity** (kinship by marriage). At the same time, he presented a sophisticated schema of social evolution based upon the relationship terms, the categories of kinship, used by peoples around the world. Through his analysis of kinship terms, Morgan discerned that the structure of the family and social institutions develop and change according to a specific sequence. He was the first to recognize and record six kin structures that are present in numerous societies and cultures around the world.

Bearing in mind the vast variety of possible options, we are going to settle on just one specific kin structure, which we shall call *traditional kinship*. We will devote the entire chapter to describing and modelling this structure.

Following Henry Morgan, we recognize two primary types of family bonds:

[1] Recall that in this thesis, the word "kinship" includes relatives as well as in-laws

marital (affinity) and parental (consanguinity). These bonds define nine basic kin terms: *father, mother, son, daughter, husband, wife, parent, child and spouse*. Observe that combining them in different ways will yield all possible kinship terms that can and do exist. For instance, *cousin* is *a child of a child of a parent of a parent* of a particular person. Another example: *mother-in-law* is just *a mother of a spouse*.

Let us introduce several useful definitions:

Definition 1. We call a kinship term **abstract** iff it can refer to relatives of different sex. For instance, the word "parent" is an abstract kinship term, because it refers to a mother as well as to a father. Other well known examples: *cousin, spouse, sibling* and *child*.

Definition 2. In contrast, a **concrete** kin term refers only to relatives of the same sex, e.g. *brother, aunt* and *nephew*.

Definition 3. A **dyadic** kin term express the relationship between individuals as they relate to one another symmetrically, e.g. if I am your cousin (sibling), then you are also my cousin (sibling). The few, and uncommon, English dyadic terms involve in-laws: *co-mothers-in-law, co-fathers-in-law, co-brothers-in-law, co-sisters-in-law, co-grandmothers*, and *co-grandfathers*.

Definition 4. An **ego** is a focal point of a genealogy, i.e. it is a person from whose point of view we will describe all other people using kin terms.

Now Let us represent a traditional christian family tree as a special type of *ontology* with its' own concepts, attributes, relations and constraints. Concepts are people in a family, their attributes are: *name, birth date, birthplace, sex* and relations are parental and marital bonds with a wedding date.

Together with the everything stated above, we have the following cultural constraints imposed on our genealogy:

1. Each person can have any finite number of children.

2. Each person can have at most two parents of different sex.

3. Each person can have at most one spouse of different sex.

4. A spouse cannot be a *direct relative*, i.e. a sibling or a parent. In other words, direct incest is prohibited.

When considering those prerequisites one should bear in mind that we deliberately focused only on rules, taboos and customs of one particular culture, namely American culture in the sense of Read [13]. Under different assumptions and in further studies, these conditions can be relaxed and revisited.

Apart from these four, here are two additional temporal constraints that express the interrelation between birth and wedding dates:

1. No one can marry a person before he or she was born, i.e. a wedding date can only be strictly after a birth date of each spouse.

2. A parent is born strictly before all of his (her) children.

Due to the general nature of these two constraints, they are always true in every genealogy and therefore can be safely assumed in our work.

Every genealogy that meets these six requirements we shall call a **traditional family tree**. As the name "tree" suggests, we can indeed view this structure as a graph with its vertices as people and edges as bonds. Observe that every kinship term corresponds exactly to a *path* between ego and specified relative. Under such a view, kin term becomes a set of instructions, telling how to get from the starting vertex `A` to the end vertex `B`. For example, consider the term *mother-in-law*. What is it if not precisely a *directive*: "firstly, go to my spouse, then proceed to her mother". The wonderful thing is that, due to the nature of kinship terminology, we can *compose* them together to create new terms, even those which don't have their own name. This simple observation that we can see kinship terms as paths in a family tree underpins our entire thesis.

Now, if we want to efficiently query a traditional family tree, we need to further investigate the mathematical features of the language of kinship terms. In the next sections we explore a formal model of kinship language, its syntax

and semantics. Then we conclude this chapter with an examination of various approaches of tackling the concept of time. We also discuss possible augmentations to our model and the problem of *term reduction*.

3.2 Formal Language of Kinship

Here we present our attempt to model the language of traditional American, in the sense of Read [13], kinship terminology. There are three main characteristics that define every formal language: its syntax (spelling, how words are formed), semantics (what does particular word mean) and pragmatics (how a language is used). In order to describe it we must expound the first two.

3.2.1 Syntax

We use Backus-Naur Form to designate the syntax for our formal language. Let Σ be the set of six basic kinship terms: *father, mother, son, daughter, husband, wife*. Then we can express the grammar as follows:

$$term ::= \Sigma \mid (term \cdot term) \mid (term \vee term) \mid (term)^{-1} \mid (term)^{\dagger}$$

The first operation is called *concatenation*, second – *fork*, third – *inverse* and the last – *dual*. We denote this language by $\mathcal{L}$.

Here are some examples of ordinary kinship terms expressed in our new language. Note that we deliberately omit superfluous parentheses and the composition sign for the sake of simplicity:

- Parent is $father \vee mother$.
- Child is $son \vee daughter$.
- Brother is $son(father \vee mother)$.
- Sibling is $(son \vee daughter)(father \vee mother)$.
- Uncle is $son(father \vee mother)(father \vee mother)$.

- Daughter-in-law is $daughter \cdot husband$
- Co-mother-in-law is $mother(husband \vee wife)(son \vee daughter)$

From these examples you can see the real power of this language – the power to express all possible used as well as not used kinship terms that can and do exist. Now the important step towards solving our main goal, developing a language for managing temporal genealogies, is to assign meaning to these words. From now on we distinguish between *artificial* kinship terms, i.e. well-formed terms of our formalization, and *natural* kin terms used in ordinary English. By referring to just terms, we mean the former, if nothing else is stated.

3.2.2 Semantics

Let Σ^* stand for the set of all possible kin terms generated from the basis Σ using the previously defined syntax. Let $\mathcal{G} = (V, E)$ be a traditional family tree with V as a set of its vertices (people) and E as a set of its edges (bonds). Moreover, because $\mathcal{G}$ is traditional, every person from the set V have the following attributes:

- A father. We will denote him as $father(p)$, a function that returns a *set* containing at most one element.
- A mother. We will denote her by $mother(p)$.
- A set of his or her children: $children(p)$.
- A set of his or her sons:

$$son(p) = \{c \mid c \in children(p) \wedge Male(p)\}$$

- A set of his or her daughters:

$$daughter(p) = \{c \mid c \in children(p) \wedge Female(p)\}$$

- A spouse: $spouse(p)$.

- A husband:

$$husband(p) = \{s \mid spouse(s) \wedge Male(p)\}$$

- A wife:

$$wife(p) = \{s \mid spouse(s) \wedge Female(p)\}$$

Due to the constraints stated in 3.1, result-set of *father*, *mother*, *spouse*, *husband* and *wife* can contain at most one element.

Now we are ready to introduce **Denotational Semantics** for Σ^*. This name was chosen because it highly resembles namesake semantics of programming languages. Note that we regard kinship terms as *functions* on subsets of V. Each function takes and returns a specific subset of all relatives, so its type is $f : \mathcal{P}(V) \to \mathcal{P}(V)$.

We proceed by induction on the syntactic structure of $\mathcal{L}$. Let t be an element of Σ^*, then:

1. If $t \in \Sigma$, then $[\![t]\!] = F(t)$, where $F(t)$ assigns to each basic kin term its corresponding function from the list 3.2.2.

2. Term concatenation is a composition of two functions:

$$[\![(t_1 \cdot t_2)]\!] = [\![t_1]\!] \circ [\![t_2]\!]$$

3. Fork is a set-theoretic union of results of its sub-functions:

$$[\![(t_1 \vee t_2)]\!] = p \mapsto [\![t_1]\!](p) \cup [\![t_2]\!](p)$$

4. Term inverse is exactly the inverse of its function:

$$[\![t_1^{-1}]\!] = [\![t_1]\!]^{-1}$$

The *dual* operator (†) is more difficult to define. We want it to mean exactly the same as the term, where the gender of each its basic sub-term is reversed, e.g. dual of "uncle" is "aunt", dual of "brother" is "sister" and so on. Here we can use induction once again:

1. If $t \in \Sigma$, then $[\![t]\!] = D(t)$, where $D(t)$ is a basic term of opposite sex.

2. Dual is distributive over concatenation, i.e. dual of concatenation is a concatenation of duals:

$$[\![(t_1 \cdot t_2)^\dagger]\!] = [\![(t_1^\dagger \cdot t_1^\dagger)]\!]$$

3. Dual is distributive over forking:

$$[\![(t_1 \vee t_2)^\dagger]\!] = [\![(t_1^\dagger \vee t_2^\dagger)]\!]$$

4. Inverse commutes with dual:

$$[\![(t^{-1})^\dagger]\!] = [\![(t^\dagger)^{-1}]\!]$$

Observe that we also have the distributivity of concatenation over forking. This semantics allows us to efficiently navigate any family tree.

3.3 Term Reduction

Our artificial language has a problem: its too verbose. Indeed, to encode such ubiquitous kin terms as "uncle" or "great-nephew" one must use quite lengthy phrases that are hard to write and read. It is therefore important to have some sort of reduction mechanism for our language that will shorten long terms into a small set of common kinship relations to aid their understanding by a user.

Firstly, let us analyse the problem. We have the following mapping ω be-

tween Σ^* and the set of English kinship terms $\mathcal{W}$

$$
\begin{aligned}
\mathit{son}(\mathit{father} \vee \mathit{mother}) &\mapsto \text{brother} \\
\mathit{daughter}(\mathit{father} \vee \mathit{mother}) &\mapsto \text{sister} \\
\mathit{father}(\mathit{father} \vee \mathit{mother}) &\mapsto \text{grandfather} \\
\mathit{mother}(\mathit{father} \vee \mathit{mother}) &\mapsto \text{grandmother} \\
\mathit{son}(\mathit{son} \vee \mathit{daughter}) &\mapsto \text{grandson} \\
&\vdots \\
\mathit{father}(\mathit{son} \vee \mathit{daughter})(\mathit{wife} \vee \mathit{husband})(\mathit{son} \vee \mathit{daughter}) &\mapsto \text{co-father-in-law}
\end{aligned}
$$

This dictionary allows us to effectively translate between kin terms of our artificial language $\mathcal{L}$ and their usual English equivalents. We can also view this mapping as a *regular grammar* in the sense of Chomsky hierarchy [29]. However, note that we strictly prohibit mixing these two collections and therefore we deliberately avoid using words from the RHS in the LHS, because otherwise the grammar will lose its regularity and become *at least* context-free, making the problem even more challenging. Let us define another function on top of ω that will replace the first sub-term $u \subset t$ in a term $t \in \Sigma^*$:

$$\Omega_u(t) = t[u/\omega(u)]$$

Here the only change in meaning of $t[u/\omega(u)]$ is that the substitution takes place only once.

Now the task can be stated thusly: given a term $t \in \Sigma^*$ find its shortest (in terms of the number of concatenations) translation under Ω, i.e. which sub-terms need to be replaced and in what order.

This problem can be easily reduced to that of of finding the desired point in the tree of all possible substitutions. Moreover, this point is actually a *leaf*, because otherwise it is not the shortest one. But the latter can be solved by just searching for this leaf in-depth. Unfortunately, the search space grows

exponentially with the number of entries in the dictionary ω, thus making the naive brute-force approach unfeasible.

Here we propose a heuristic greedy algorithm A.1 that, although does not work for all cases, provides an expedient solution to the reduction problem in $O(n^2)$ time. Firstly, it finds the longest sub-term u that exists in the dictionary ω, then divides the term into two parts: left and right from u, after that it applies itself recursively to them, and finally it concatenates all three sub-terms together.

Now let's analyse the time complexity of this algorithm:

Theorem 1. *The execution time of the algorithm listed in A.1 belongs to* $\Theta(n^2)$.

Proof. Let $T(n)$ be the execution time of the algorithm, where n stands for the number of concatenations in a kin term. First of all, observe that $T(n)$ obeys the following recurrence:

$$T(n) = 2\,T(n/2) + O(n^2) \tag{3.1}$$

Indeed, we make a recursive call exactly *two* times and each call receives roughly the half of the specified term. During execution the function passes through two nested cycles, so one call costs us $O(n^2)$.

Secondly, we use the **Master Method** from the famous book *Introduction to Algorithms* [30] by Cormen et al. In our case $a = 2$, $b = 2$, and $f(n) = O(n^2)$. Observe, that if we take ϵ to be any positive real number below one: $0 < \epsilon < 1$, then $f(n) = \Omega(n^{\log_b a + \epsilon}) = \Omega(n^{1+\epsilon})$.

Let us show that $f(n)$ satisfies the *regularity* criterion: $af(n/b) \leqslant cf(n)$ for

some constant $c < 1$. Indeed, just pick $c = 1/2$:

$$\begin{aligned} 2f(n/2) &\leqslant cf(n), \\ 2\frac{n^2}{4} &\leqslant cn^2, \\ \frac{1}{2}n^2 &\leqslant cn^2, \\ \frac{1}{2}n^2 &\leqslant \frac{1}{2}n^2 \end{aligned}$$

Thus, we can use the third case from the Master Method, which tells us that $T(n) = \Theta(n^2)$. □

3.3.1 Pursuing Confluence

Current greedy approach listed in A.1 has one major disadvantage: like any other greedy algorithm it can fail to choose a correct reduction path between two terms with equal amount of concatenations. We can alleviate this by augmenting our rewriting system, based on ω dictionary, with a feature called *confluence*, also known as *Church-Rosser* property:

Definition 5. An abstract term rewriting system is said to possess **confluence**, if, when two terms N and P can be yielded from M, then they can be reduced to the single term Q. Figure B.1 depicts this scenario.

Not only we can fix our reduction algorithm by introducing this property, but also we can improve the time complexity, making it linear.

One way to achieve confluence is to attach a single kinship term to any possible path in a family tree. Observe, that English kinship terms have a specific pattern that we can exploit. All relatives who are distant enough from ego have the following structure of their kin term:

$$n^{th} \text{ cousin } m^{th} \text{ times removed}$$

In-laws also have their own pattern, where the ending "-in-law" is appended to

a valid consanguine kinship term. However, this applies only to people, who are linked together by only one nuptial bond. For instance, there is no single term for a husband of ego's wife's sister. These relations can be accounted for by *prefixing* "-in-law" with an ordinal, which shows the number of marital bonds that one should pass in order to go to such person. Under this representation, last example will receive the term "brother *twice*-in-law". Generalizing that scheme we will get a pattern that looks like the following:

$$\langle \text{Consanguine kinship term} \rangle k^{th} \text{ times-in-law}$$

We can also view this as an attribution of a distinct *natural number* to every vertex with ego as an *offset*, thus imposing a natural ordering on the set of all vertices. This assignment can be made in such a way that reducing a kinship term n will correspond *exactly* to the calculation of n from some arithmetic expression like $5 \cdot (2+3)+4$, thus providing a **translation** between the language of all valid arithmetic expressions and our formal language of kinship $\mathcal{L}$.

However, it is not the topic of this thesis, so we are leaving it to the considerations of future researches.

3.4 Incorporating Time

Now the only matter that is left to address is an adequate representation of time. Historically, there are two main approaches for modelling time: point-based and interval-based. The former treats time as a single continuous line with distinguished points as specific *events*, and the latter uses *segments* of that line to represent time entries, which is more famous. For instance, it was used in Allen's interval algebra [16]. For the sake of simplicity we chose the former approach, because it can easily imitate intervals by treating them as endpoints of a line segment.

Not only we want to talk about different events by modelling them as points on a line, but also we want to orient ourselves on that line, i.e. to know where

we are, which events took place in the past and which will happen in the future. Thus, we need to select exactly one point that will stand for the present moment and call it "now". Then all point to the left will be in the past, and all point to the right will be in the future. Also, notice that any set with total ordering on it will suffice, because the continuous nature of a line is redundant in point-based model. Collecting everything together, we have the following formalisation of time:

$$\mathcal{M} = \langle T, now, \leqslant \rangle$$

Where T is a non-empty set with arbitrary elements, $now \in T$, and $\leqslant$ is a total ordering relation on T.

Within this model we can reason about which event comes *before* or *after*, what events took place in the past or in the future, and so on.

When considering family trees it's necessary to define only five predicates:

1. $Before(x, y)$ is true iff $x < y$.
2. $After(x, y)$ is true iff $x > y$.
3. $During(x, s, f)$ is true iff $s \leqslant x \leqslant f$.
4. $Past(x)$ is true iff $Before(x, now)$.
5. $Future(x)$ is true iff $After(x, now)$.

Those relations are the basis from which all other operations on $\mathcal{M}$ can be defined. It is also interesting to note that, since any ordering relation generates a *topology* over its structure, we can speak about time in terms of its topological properties.

This observation concludes our methodology chapter.

Chapter 4

Implementation

...

Chapter 5

Evaluation and Discussion

...

Chapter 6

Conclusion

...

Bibliography

[1] B. C. Smith, "Reflection and semantics in a procedural language," Ph.D. dissertation, MIT, Cambrige, 1982.

[2] H. J. Levesque and R. J. Brachman, "Expressiveness and tractability in knowledge representation and reasoning," *Comput. Intell.*, vol. 1, no. 3, pp. 78–93, 1987.

[3] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," Ph.D. dissertation, MIT, Cambrige, 1971.

[4] H. Morowitz, "The plural of 'ontology' is 'confusion'," *Wiley Periodicals*, vol. 17, no. 6, 2012.

[5] B. Chandrasekaran and J. R. Josephson, "What are ontologies, and why do we need them?" *IEEE Intelligent Systems*, 1999.

[6] T. M. Ting, "Building a family ontology to meet consistency criteria," Master's thesis, University of Tun Hussien, 2015.

[7] J. B. F. B. T. Furche and S. Schaffert, "Web and semantic web query languages a survey."

[8] M. Marx, "Xcpath the first order complete xpath dialect."

[9] C. Lai and S. Bird, "Querying linguistic trees," 2009.

[10] A. A. R. K. A. K. V. R. F. Wolter and M. Zakharyaschev, "Ontology-mediated query answering over temporal data: A survey," *24th International Symposium on Temporal Representation and Reasoning*, vol. 1, no. 1, pp. 1–37, 2017.

[11] J. Bartlema and L. Winkelbauer, "Modelling kinship with lisp; a two-sex model of kin-counts," *IIASA Working Papers*, vol. WP-96-069, no. 1, p. 48, 1961.

[12] N. V. Findler, "Automatic rule discovery for field work in antropology," *Computers and the Humanities*, vol. 26, no. 1, pp. 285–292, 1992.

[13] D. W. Read, "An algebraic account of the american kinship terminology," *Current Antropology*, vol. 25, no. 49, pp. 417–429, 1984.

[14] V. Periclev and R. E. Valdes-Perez, "Automatic componental analysis of kinship semantics with a proposed structural solution to the problem of multiple models," *Anthropological Linguistics*, vol. 40, no. 2, pp. 272–317, 1998.

[15] M. Minsky, "A framework for representing knowledge," 1974.

[16] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, 1983.

[17] Y. Shohan, "Temporal logic in ai: Semantical and ontological considerations," *Artificial Intelligence*, vol. 33, no. 1, pp. 89–104, 1987.

[18] A. Schmiedel, "A temporal terminological logic," *AAAI-90 Proceedings*, vol. 1, no. 1, pp. 640–645, 1990.

[19] R. J. Brachman and J. G. Schmolze, "An overview of kl-one knowledge representation system," *Cognitive Science*, vol. 9, no. 1, pp. 171–216, 1985.

[20] P. B. L. Libkin, "Temporal logic over unranked trees," 2008.

[21] A. Tuzhilin and J. Clifford, "A temporal relational algebra as a basis for temporal relational completeness," *Proceedings of the 16th VLDB Conference*, 1990.

[22] M. S. Perry, "A framework to support spatial, temporal and thematic analysis over semantic web data," Ph.D. dissertation, University of Georgia, 2008.

[23] E. B. E. G. P. S. B. N. Maris and N. Papadakis, "Toql: Temporal ontology query language."

[24] B. Couderc and J. Ferrero, "Fr2sql : database query in french," *22eme Traitement Automatique des Langues Naturelles*, 2015.

[25] F. D. M. S. M.H.Shenassa, "Using natural language processing in order to create sql queries," *Proceedings of the International Conference on Computer and Communication Engineering*, 2008.

[26] P. P. Chaudhari, "Natural language statement to sql query translator," *International Journal of Computer Applications*, vol. 82, no. 5, 2013.

[27] R. Nelken and N. Francez, "Querying natural language databases using controlled natural language," 2001.

[28] L. H. Morgan, *Systems of consanguity and affinity of the human family.* Smithsonian Institution, 1870.

[29] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 1, pp. 113–124, 1956.

[30] T. H. C. C. E. L. R. L. R. C. Stein, *Introduction to Algorithms*, 3rd ed. The MIT Press, 2009.

Appendix A

Pseudocode Listings

Algorithm A.1: Kinship Term Reduction

```
1   input: kinship term t.
2   note: ω is a dictionary of kinship terms.
3   note: Functions "leftPart(t, u)" and "rightPart(t, u)" return the sub-term of t
4   note: from the left of sub-term u or from the right respectively.
5   note: Function "subterm(t, i, j)" returns the
6   note: sub-term of the kinship term t between indices i and j.
7   output: reduced kin term.
8   function shorten(t)
9   begin
10      maxShortenableSubterm ← empty
11      currentSubterm ← empty
12      for i ← 0 to length(t) do
13      begin
14          for j gets length(t) − i to 0 do
15          begin
16              currentSubterm ← subterm(t, i, j)
17              if length(currentSubterm) > length(maxShortenableSubterm)
18                  and ω(currentSubterm) is not empty
19              then
20                  maxShortenableSubterm = currentSubterm
21          end
22      end
23      return shorten(leftPart(t, maxShortenableSubterm))
24             · ω(maxShortenableSubterm)
25             · shorten(rightPart(t, maxShortenableSubterm))
26  end
```

Appendix B

Figures

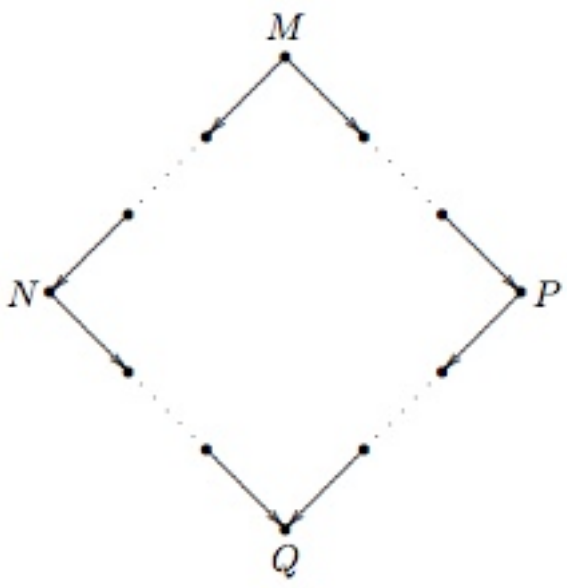

Figure B.1: Confluence in a term rewriting system.