Evaluating Natural Language Access to Relational Databases

Mårten Forsmark

December 20, 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Michael J. Minock
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

Natural language interfaces were once considered a very promising research area but today the area is regarded with mistrust. Other interfaces dominate today's design space and there are few applications that use natural language interfaces. This master's thesis tries to determine if the mistrust of natural language interfaces is justified or if there is a need for natural language interfaces. One natural language interface will be particularly scrutinized, namely STEP. STEP is an ongoing research project that provides natural language access to relational databases. Rigorous tests and comparative studies has been performed on this interface to determine whether natural language interfaces is on par with other interfaces.

Contents

Chapter 1

Introduction

This report is a 20 credit master's thesis in computer science, developed at Umeå University, at the Department of Computing Science. The goal of the thesis is to evaluate natural language interfaces and in particular the Schema Tuple Expression Processor (STEP) system[16]. The original idea for this master's thesis was proposed by Michael Minock, who is also my supervisor and the developer of STEP.

The STEP system is an ongoing research project and the focus has been to see how well it performs and to see if it can close the so called *conceptual gap* between the information desired and the method used to obtain it. To evaluate STEP an application using Swedish history as it's domain has been developed and assessed.

Some of you are probably wondering what a natural language interface is. A Natural Language Interface (NLI) is a user interface that interprets user requests expressed in a natural language (e.g. English). One of the most common uses for this kind of interface is as a Natural Language Interface to a Database (NLIDB) where it will transform the given expression into a formal query language, typically the Structured Query Language (SQL). The main benefit would be that the user would need little or no training to perform rather advanced tasks.

During this thesis all references to Natural Language Interfaces will assume that the user requests are written on a standard computer keyboard, hence issues related to voice recognition and similar technologies will not be addressed. Furthermore this thesis is mainly based on published papers and the author lacks personal experience with using most of the referenced natural language interfaces. When a system feature is mentioned the cited document states that the system has that feature and it is not implied that other systems lack this feature.

I will proceed by presenting some background information on NLIs are presented together with the benefits and drawbacks of using one. This involves discussing how natural language interfaces can have features such as expressing negations and quantifications in a concise manner, referring objects not current on screen and using techniques such as underspecified sentences. Some problems with natural language interfaces will be discussed these include portability, anthropomorphization[1] and how to clarify the boundaries of the conceptual and linguistical coverage to the users. Techniques to ease or perhaps even overcome these problems will also be discussed.

We will then move on to explain the basic idea behind the phrasal lexicon (see section 2.4) which STEP is built on. Following this a descriptions of how STEP work and what

[1]Attribution of human motivation, characteristics, or behavior to inanimate objects.

distinguishes it from other interfaces is presented.

To test STEP and it's capabilities a test application has been written using Swedish history as a domain. A short description of how this application was developed and how a Finite State Transducer (FST) was constructed to better handle temporal queries will be given.

When this is done we can finally move on to the performed studies and the conclusions that can be drawn from them. The studies will mainly focus on testing the coverage of the application, the general software requirements and a comparison with a standard form-based interface.

Chapter 2

Background

This chapter provides a closer look on the brief history of natural language interfaces and discusses some of the advantages and disadvantages with using a natural language interface. When this is done the notion of a phrasal lexicon will be presented together with the natural language interface STEP.

2.1 The History of Natural Language Interfaces

The very basis for automated natural language processing could be said to date back to the early 1900s and to the reconstruction of mathematical reasoning using logic. This kind of research was manifested in the work by Frege, Russell, Wittgenstein and others. Their work suggested that natural languages was a formal system amenable to automatic processing[26].

Prototype applications that tried to parse natural language started to show up as early as the late sixties[2] but they were mostly bound to a very specific database and application. The most noteworthy of these was LUNAR[31] which was a natural language interface to a database with chemical analyses of rocks from the moon. LUNAR had a finite state machine that described all the possible inputs it could parse. This worked fairly well for this specific application and created a hype around natural language processing. It was commonly believed in the research community that natural language processing was restricted enough to be feasible and it would soon become the de facto standard for computer interfaces. A researcher at IBM said as late as 1978 that "Natural language understanding can't be that far in the future. Little children learn in just a few years."[32] Time has proven this researcher wrong.

Although many research hours has been spent and commercial system have been available for decades, the practical usage of natural language interfaces is not very widespread. It seems that companies mistrust NLIs and that they lack knowledge or conviction about their benefits over other types of interfaces[22]. This mistrust has enabled other types of interfaces to dominate the design space of today, like keyword search and form based interfaces. But is this mistrust justified or are there actual benefits of using NLIs?

2.2 Advantages with Natural Language Interfaces

There are several reasons as to why it would be advantageous to have a natural language interface. The most obvious one would be that the user of the system would not need to learn any artificial language to be able to communicate with the computer. This would benefit companies that do not need to educate all the employees and it would also be ideal for the occasional end-user that has a very limited computer experience.

Some researchers would say that achieving a natural language interface is just a partial goal because when we have a computer that can understand natural language, then the completion of many researchers dream is not so far away. This dream is to have HAL9000-like[1] interfaces, where you can have a conversation with your computer. Many believe that this kind of interface is not so far into the future since there are inventions like quite capable grammar checkers, voice recognition programs and language translation tools. This is a misconception since these inventions is not by far good enough for natural language applications and that kind of interface is still a pipe dream[13].

Is there any other reasons to have natural language interfaces other than to achieve some far-fetched pipe dream? The interfaces used today seem to work well, don't they? Even though graphical and form-based interfaces are effective in many areas they have their limitations. Some of these limitations include the ability to reference objects not visible on the screen, temporal relations and difficulties expressing negation and quantification in a concise manner (e.g. *Which department has no programmers?*, or *Which companies supply every department?*)[2, 6]. Luckily these are the strengths of natural language and as you can see the questions are concise. Of course these questions can be expressed in other interfaces or in formal query languages like SQL but they will be more complex and tedious to manually write.

Another advantage of having natural language interfaces is the possibility to support underspecified expressions like anaphoric and elliptical expressions[6]. An anaphoric expression is an expression that refers back to a previous expression in a natural language discourse. For example: "Martin died. *He* was not old." The word he refers to Martin, and is described as an anaphoric reference to Martin. Anaphora can be very useful to allow the user to refine a query. The following example of the usefulness of with anaphora is gathered from [27].

>*Is there a ship whose destination is unknown?*
- `yes`
>*What is it?*
- `What is the ship whose destination is unknown?`
- `Saratoga`

An elliptic expression or an ellipsis refers to situations in which sentences are abbreviated by leaving out parts of them that are to be understood from the context. For example, if someone asks "What is your name?" and the reply is "John Smith" then this can be viewed as an elliptical form of the full sentence "My name is John Smith". This can be quite useful since the standard user would probably appreciate if they need to write less to operate the interface. A interesting natural language understanding project that often uses elliptical sentences is the NICE[2]-project and their NICE fairy-tale game[5]. The NICE project focuses on a promising area for natural language

[1]An artificial intelligence from the book *2001 - A Space Odyssey*, by Arthur C. Clark
[2]Natural Interactive Communication for Edutainment

understanding, namely edutainment[3]. An ordinary input sequence in the fairy-tale game can look like this.

>Cloddy Hans, please pick up the axe.
- `OK.`
(The character picks up the axe)
>Now the hammer.
- `Will do.`
(The character picks up the hammer)

To humans it is obvious that the second command given by the user implies that the character Cloddy Hans should pick up the hammer, since we are aware of the context. Natural language understanding systems need to be aware of the context as well, in this case using elliptic expressions to realize that it is directed towards Cloddy Hans and that the user wants him to pick up the hammer.

Anaphoric and elliptical expressions are usually not possible to express in formal query languages, graphical and form-based interfaces because they try not to be context sensitive. This is because context free applications are much easier to develop and there exist strict rules for such behavior.

All these advantages sound nice but are they useful in every day applications? Unfortunately there have been very few long term studies, however Walker and Whittaker conclude that natural language interfaces can be better than menu-based systems. The subject in the study that preferred the natural language interfaces wanted to retrieve information with many different levels of granularity, especially when in conjunction with coordinated expressions, such as conjunctions[29]. This was often used to join two sets and thus creating a new ad-hoc set they could operate on, which strongly resembles how it would be done in SQL. The subjects had some of the advantages of SQL and this combined with the possibility to express complex queries in simple ways caused some subjects to prefer the natural language interface over the standard menu-based system, even though the natural language interface was much more prone to errors. This study and other similar studies suggests that the benefits of natural language interfaces are not simply of theoretical value but also useful in every day applications.

2.3 Disadvantages with Natural Language Interfaces

The first thing to decide is whether any *natural* language interfaces actually exist. NLIs today can only describe a limited subset of a natural language, since natural languages are too vast and continuously changing. It is easy to argue that a subset of a natural language is no longer a natural language and thus no *true* natural language interface exist today.

The problem of natural language interfaces not being able to handle an entire natural language has other side effects as well, even if the user is aware of this limitation. Implementations of natural language interfaces often only range over a restricted domain and only tries to cover the natural language needed for this domain. The subset of natural language an application operate over is called its linguistical coverage. Where the linguistical coverage of an application ends is not often obvious to the user[6, 25]. Users find it difficult to understand and remember what kind of questions the system

[3]The fusion between education and entertainment

can answer. It is very easy for users to make an assessment of a system based on a single question and get a false expectation of the system. If the user asks what they believed to be a simple query, but outside of the systems domain, he will likely get a “Unable to answer query”-message. Since it was an *easy* query studies show that he will more likely believe that the system is faulty rather than assume that the query was outside of the domain[7]. This is a real problem and there is no absolute solution to this. STEP tries to solve this by showing the user example queries that shows valid queries within the domain. This should in theory give the users an better estimation of where the domain boundaries are drawn.

Formal query languages, form-based and graphical interfaces usually do not suffer from these problems, since the domain are more clearly displayed. There have been attempts to combine form-based interfaces and natural language interfaces to eliminate this disadvantage[25] but it seems to have had little impact on the commercial and scientific communities.

Another problem is that when a natural language interface fails it is often unclear whether it was a conceptual or linguistic failure. This often causes users to rephrases questions even though they are outside the systems conceptual coverage, because they believe it is outside its linguistic coverage[25]. Of course this also applies the other way around. The worst thing that can happen when a failure occur is not that the system does not answer but that it returns an incorrect answer. This has led to many misinterpretations. Luckily this problem can be solved by paraphrasing the given question back to the user. This will show how the system has interpreted the question and let the user deduct whether it was the desired question. STEP[16] takes this even further and uses paraphrasing to resolve ambiguity as well. Paraphrasing queries is an important part of any practical natural language interface over databases which unfortunately often is neglected[16].

When a system paraphrases the questions back or try to keep some kind of dialogue with the user, another problem occur. Users could anthropomorphize the computer and believe that it has common sense, ability to deduct facts or even attribute it with *intelligence*. This leads to unrealistic expectations of the systems capabilities and when these are not fulfilled the user will be disappointed and believe that the interface is flawed[8, 23].

These problems can be somewhat relieved by clearly letting the user know that it possesses no intelligence or by letting the system have some deductive abilities. This means that the system tries not simply parse the question but also to act as a intelligent agent. The natural language interface does not simply use the database as its only source for knowledge and adds knowledge about the user or the surrounding world. There could be some kind of rules that carries out reasoning based on the raw data in the database. Often some kind of logic-system is incorporated which performs this reasoning.

One of the more practical problems is that natural language interfaces often require tedious and lengthy configuration. Older systems were so specific that they could not change domains nor which natural language it used. Still today there is often a fair amount of time involved in setting up a natural language interface for a new domain.

To be able to set up a natural language interface for a new domain one needs to fulfill a few basic requirements. The most obvious requirements are that when configuring the system knowledge about the new domain and the natural language on which the interface is to operate is needed.

These requirements are often fulfilled but are usually not enough to satisfy the system. Since each domain has its own demands and part of the system must be configured to the new domain. Most systems try to keep this part small and well defined to facilitate the rewriting, unfortunately not all of them suceed. A third requirement is thus that we need a programmer familiar with how to configure the system. This is often the hardest requirement to fulfill since configuring a system is not always the easiest of tasks.

There are systems which attempt to eliminate the need for a programmer to rewrite code to set up a new domains. One example of this is the natural language interface Parlance[2]. In Parlance the end-users are responsible for expanding the system. The end-users could extend the capabilities of the interface by using natural language. An example of this follows.

```
define:  a rich employee is an employee which earns over 100K per year
```

In this example the end-user has created a rule that defines a group which fulfill a certain criteria. In the future the end-user will only need to use the name of the group instead of specifying it every time. This is not the only configuration task that can be left to the end-users. The system can let the users modify the database by allowing inserts, updates and deletes via the natural language interface. This would let the database be altered by end-users with expertise in the domain, but possessing fairly limited computer experience to extend or alter the domain. If one looks one step even further it could perhaps even be possible to change the database structure and phrasal lexicon via the natural language interface.

There is another way to work around the domain portability and domain boundary problems and that is to have a open domain. A open domain is a domain that contains every possible domain, i.e. a superset of all domains. Open domain systems obviously do not suffer from boundary problems since there are no boundaries, but they have other problems. Since open domain systems claim to work independently of domain these interfaces must handle all possible phrases in the natural language used, which currently is an unobtainable demand for every known natural language[24]. Even if open domain systems could parse every valid sentence in a natural language they would have a much harder time resolving ambiguious queries than a closed domain system. This is quite obvious since open domain systems operate on a larger domain and can not use any context knowledge specific to the domain. Because of these problems open domain system are usually more of theoretical use than practical.

It is not hard to find researchers that point out a need to have a knowledge base to facilitate natural language processing[9, 14, 21]. For a system operating on a closed domain this is quite obvious since it will greatly improve the disambiguation and parsing. There are of course benefits for open domain system as well but where does one find a knowledge base that knows everything? There are those who believe that these problems and the problems with open domain systems, can be solved by using the enormous general knowledge base CYC[4]. Using CYC for natural language processing may sound like a great idea but it is not without it's problems[15]. A more promising area to use CYC is to add it as a supporting program to closed domain application. It's function would be to provide better responses than *"Unable to answer query"*, when a query goes outside the applications conceptual coverage.

Porting between domains is not the only issue in natural language processing. One would like to be able to port an application between different natural languages. To translate between two natural languages is not an easy task. Even though there exists

[4] `http://www.cyc.com/cyc/technology/whatiscyc`

some fairly good translation programs these are far from fulfilling the demand of a natural language interface. Most systems and research carried out assumes that the only natural language used is English. To modify an existing natural language interface is not an easy task since it may be based on assumptions of the natural language used. Often it involves to rewrite the syntax rules, the semantic rules and the lexicon. Even though its a difficult task to port natural language interfaces there are successful examples e.g. there is a Swedish version of the interface CLE[1].

2.4 What is a Phrasal Lexicon?

The basic idea behind the phrasal lexicon is that our utterances are formed by repetition, modification and concatenation of previously known phrases consisting of more than one word[4]. In other words we simply stitch together pieces of text we have heard before. One do not usually think about it but when you start to analyze natural language one finds surprisingly many idioms, allusions, slogans, cliches and even lines from popular song lyrics or TV-shows. This theory is quite the opposite of theories were utterances are combinatorially created by from words or even smaller units.

The phrasal lexicon is simply a set of sentence templates that is coupled to elementary expressions of a tuple calculus class. These sentence templates corresponds to the likely phrases that users give the interface. The phrases will vary between different domains but there are some generic phrases that can be reused when porting an application to another domain.

To give examples of what a phrase may look like I took a few phrases from this section and put them in table 2.1. The phrases listed in the table may not be ideal for a natural language interface for databases but clearly shows the uses of a phrasal lexicons in natural language understanding.

Table 2.1: Some phrases found in this section

the basic idea	in other words	stitch together
quite the opposite	to give examples of	may look like
may not be	but clearly shows	the uses of

2.5 STEP

The STEP system is a natural language interface that provides an interface to relational databases. Provided with a database and a phrasal lexicon over a restricted domain, STEP processes natural language in a bidirectional way, meaning that it parses queries in a natural language and attempts to answers them in the same natural language. The only natural language which STEP currently tries to support is English, but some initial experiments has been carried out on Swedish.

The database can be any relational database while the phrasal lexicon must be written, in CLISP, using some syntax specific to the STEP application. This phrasal lexicon is compiled into a subsumption hierarchy.

A subsumption hierarchy is an ontology[5] in which the parent subsumes it's children. A concept A subsumes a concept B if the definitions of A and B logically imply that members of B must also be members of A. To exemplify this we can use an example from natural language. A *person that has children* subsumes *person that has a son*, since the more general description (subsumer) subsumes the more specific description (subsumee) of the same sort. Since the subsumee is a member of it's subsumer it also inherits information and attributes from it. This means if you are told that *People that have children are happy*, then it follows that *People that have a son are happy* as well.

When this subsumption hierarchy is created each sentence template from the phrasal lexicon is sorted into this subsumption hierarchy. Since ontologies try to describe the semantic relationships between objects, this can be seen as a semantical sorting of the templates.

The core representation of all data in STEP is a class of tuple calculus. Tuple calculus is actually first order logic with some 'syntactic sugar' on top to allow variables to range over entire tuples instead of the i-th position of a given predicate. There exists a direct translation from the tuple calculus into first order logic and SQL as well[16].

To enable natural language understanding an inverse search process is performed when receiving user input. In this case inverse search refers to a search where you search for something that could generate the input. This specific inverse search is a state-space search and it tries to find the best fitting tuple calculus query which would have generated text matching the input sentence. Since there is a direct translation between tuple calculus and SQL this is equivalent to searching for the database query the user is trying to express. For more information on how the natural language understanding works see [16].

This type of parsing within the phrasal lexicon is called knowledge based parsing. In STEP a cost is attached to the parsing. To cope with variations and errors in a query the input can be *fudged* into something more likely to be parsed, with an additional cost. When the parse deviates too far from the entries in the phrasal lexicon the cost will rise. If the cost for a possible match reaches above a specific threshold it will be paraphrased back to the user to confirm that the correct query is answered. If the cost reaches a second threshold the system states that there are no possible parses for the given query. To enable more possible parses the synonym finder Word-Net is used to match words that are not known by the phrasal lexicon.

When a parse has been found STEP processes the information by issuing satisfiability queries to the SPASS theorem prover. SPASS only operates on first order logic, but since the query is in tuple calculus a direct translation can be made. When a query is deemed appropriate an answer pattern is retrieved from the phrasal lexicon. This answer pattern is constructed by combining the queries immediate parent in the subsumption hierarchy and combining it with other linguistic resources. This may also cause an extension of the query so it must fetch additional relevant information. This answer pattern dictates how the results will presented to the users. Finally the query is translated into SQL, once again with a direct translation, and issued to the database. The result from the SQL-queries are inserted in the answer templates and displayed to the user. For more detailed information on how the natural language generation is performed see [18].

[5]A network of the semantic relationships that describe how objects relate to one another

Chapter 3

Method and Approach

This chapter provides a closer look on how the test application was developed and how a natural language interface should be evaluated. This application and some of the evaluation methods will later be used to assess the STEP interface.

3.1 Creating the Database

The first step to creating a natural language interface for a database is to create a queriable database. The first step to create a queriable database is to decide which domain to work on. Since history can be a complex domain to represent and since Swedish history has been one of my hobbies for a long time this domain was selected. It is a good thing that history is complex since one of the requirements for natural language interfaces over restricted domains to be able to outperform other types of interfaces[17]. The vision for the application was to have a system where users interested in Swedish history easily could query over dates, events, persons and determine what happened at approximately the same time. A domain based on history would also test how suitable the STEP interface is to handle temporal aspects, an area previously unexplored.

To facilitate the construction of the application I contacted the Swedish History Society and asked to get a copy of their huge database to work from. Everything looked fine until I received the database. The database was a real mess since no one maintained it and I quickly realized that it would be better to create one of my own, even though it would contain much less information. To simplify the writing of the database, the free online encyclopedia Wikipedia[1] was partially parsed and used as input into the database. Since this is a lengthy process and not the main focus of the thesis the database only contains a very limited amount of data, but enough to test the application properly.

To design the database I first tried to imagine what kind of questions I would like the interface to answer. Unfortunately some of these questions soon turned out to be impossible to express in a database, because the facts were too vague, entry specific, disputed or imprecise. Every record-based information model assumes vertical and horizontal homogeneity, this means that each record should contain the same fields and that any given field should contain the same kind of information. But when there is considerable variation of what facts represent each member in the set more information must be stored to describe the items accurately.

[1]`http://www.wikipedia.org`

A new field could be made to handle this information but when the number of fields in a record is almost as many as the number of entries, where most entries have null values, it becomes untenable. New tables instead of fields could be added to avoid having many null values, but then instead of the null fields we get an untenable situation with redundant data. This can be seen as one of the limitations of the record-based information model[12] and more information about it can be found in section 5.2.

To solve these problems with the record-based information model the easy way out was taken, some data within the database model was simply deemed unqueriable and scrapped. The rest of the questions was used as a foundation for an EER-diagram[2] to model the database schema. The final version of this diagram can be seen in appendix A. The initial design lacked and had abundant tables so some tables were added and removed during the iterative development of the database.

The database in which the application data is stored is PostgreSQL[3]. The choice of using PostgreSQL was quite easy since it is the database I am most familiar with and at the time it was the only database that STEP supported. Since then ODBC[4] has been integrated with STEP so other relational databases are supported[20].

Since the main objective of this thesis was not to build a complete database over Swedish history the design focused on not trying to span too many areas. The main tables then became important foreign `powers`, `rulers`, important `locations`, important `persons` and important `events` in Swedish history. No information was entered from the time before Gustav Vasa, since older information seemed to vague to be of any concrete use.

3.2 Expanding the Temporal Capabilities of STEP

It is virtually impossible to create a database over history without using any representation of when something occurred. Since STEP did not have any kind of support for temporal queries what so ever, this was an area that required some development. STEP was expanded so it was possible to specify what values it should treat as dates instead of a string. For the sake of simplicity the only format of dates that STEP supports internally is the ISO 8601 date format [5]. This date format is both unambiguous and is easily parsed if needed. ISO 8601 is the recommended format of writing dates in SQL.

Since there are several other formats of expressing a date and a period of time can be referenced on several levels of granularity there was a need for more. A function that could convert any known time format, on any level of granularity into a date of the ISO 8601 standard was developed.

Since it was believed that a Finite State transducer (FST) would be beneficial to STEP it was decided to perform this translation with a FST. A FST can be described as a finite state machine, but with the ability to produce other output than it's input. This means that it translates the given input from the input language into a valid string in the output language. A general FST was constructed for STEP that allows the construction of an FST by giving the grammar of the input language and what to produce as output. Some basic operations on FST was also constructed such as taking the disjunction or product of two FSTs.

[2]Extended Entity-Relationship
[3]`http://www.posgresql.org`
[4]Open Database Connectivity
[5]`http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html`

This particular FST accepts a language that can have several ambiguous translations such as the date `2/3/2005` which in US-style means 3rd of February and 2nd of March in European style. To solve any problems that can arise from this a non-deterministic FST was used. This means that the FST will, given a valid string in the input language, accept it and produce all known valid interpretations of that string.

The task of writing an FST and a lexicon for all known formats and for several levels of granularity proved much harder than first believed. The main problem is that there are simply far too many ways of expressing a period of time and it is not always obvious how to translate them into a date. Therefore it was decided to ease the requirement of handling different levels of granularity and instead focus on how to simply parse most known formats of a date. Another problem that came up was the problem with translating an interval into dates, such as *1700th century* or *1945*. There simply is no way for the application to know that it is an interval instead of a single date. This problem was instead solved in the phrasal lexicon by allowing the user to specify intervals more specific such as *between 1945-01-01 and 1945-12-31*. A better solution that is left for the future is to specify a internal format interval that simply contains two dates. A new FST lexicon handling the interval input could be created and used instead of the FST for dates, this would solve these problems. Something that is even harder is how to allow references to time that are relative to the current date, such as *yesterday 200 years ago*. How to solve this in a general way is left for the future.

As previously stated this FST accepts most formats of dates but not all of them. Some known formats that are not supported are the ones that specifies a year and a number of days into that year. Fortunately this format is not commonly used in natural language. A list of most supported formats can be seen in table 3.1 and a graphical display of the FST can be seen in appendix C.

Table 3.1: The date formats supported by the finite state transducer

Plain text	January 18, 1999
Plain text	18th Jan 1999
ISO-8601	1999-01-18
ISO-8601	19990118
US date format	1/18/1999
European date format	18/1/1999
German date format	18.01.1999

3.3 Writing the Phrasal Lexicon

The next step in developing the test application was to write a phrasal lexicon. The obvious way to do it was to start out with the questions that were used to prototype the database and work from there. Most of these queries were quite easy to incorporate into the phrasal lexicon.

After writing a fair amount of simple queries, more complex queries were needed but then a limitation in the phrasal lexicon was encountered.

The first problem that emerged was when one pattern is a subset of or semantically equivalent to another pattern. When semantically sorted one pattern becomes the parent of another or in the same node of the search tree. When searching this tree to determine the how to generate the natural language output both queries will be matched. For exemple this let us say that we would like to have the two queries, *"Give countries that have had control over the region Skåne"* and *"Give country that controlled Skåne in 19000101"*. The semantic difference of these two queries is that the later limits the selection even further by giving a specific date. This means that the first query will become the parent node of the other query. The problem occurs when a query is asked and STEP finds two possible matches for the answer. This is not surprising since the latter query is a special case of the first. The output will then be garbled and look something like this.

>Give country that controlled Skåne in 16590101

```
- According to the database, the country that have had control over Skåne,
that had control over Skåne in 16590101 is Sweden.
```

To fix this problem one has to make them more semantically different. This was made by adding an extra dummy condition to one of the queries which alters the semantics of the query in such a way that it will still yield the same results. This is done by testing an attribute that is completely unrelated. This is somewhat tricky and not always solvable. Some work on the problem with multiple matching of output has been done in later versions of STEP but as far as I know, no complete solution that does not also introduce other problems has been found. I am confident that this problem will be solved in later releases of STEP.

Later on another problem was discovered while performing pattern matching. It seemed like some of the questions were hard to give an answer to without making the database contain exponentially more tables with redundant information. This problem was because the only two logical operators in the pattern matching process of the phrasal lexicon are *and* and *atomic negation*. With these logical operators we can not construct all the other logical operators e.g. *or* and *not*, hence it is limited in what it can express.

As an example of this we can use the example of contemporary persons. We would like to have a list of famous persons that lived at the same time as Gustav Vasa from a database table that contains the persons name, date of birth and date of death. It can be a bit bewildering but to be a contemporary of Gustav Vasa one must be born before and to have died after Gustav Vasa was born *or* born before and to have died after Gustav Vasas death. Since *or* is not possible to create and there is no operator that unions data sets, this can not easily be done in the phrasal lexicon. Of course one could write a function that is in a layer closer to the user that recognizes this specific query, makes both of the queries separately and unify the results, but that is not how it should be done. It breaks almost every principle of good programming and it only solves this special case, not the general problem.

To solve this problem the query was split into two separate queries, one for the earlier contemporaries, when the person was born before Gustav Vasa and later contemporaries that died after Gustav Vasa. When a query is performed on which the contemporaries of Gustav Vasa are the disambiguation engine matches both the younger and the older contemporaries and then prompts the user which one to choose. This means that the user will never be able to get all the contemporaries in one query and must take the union of the result sets themselves. Also notice that it is possible for someone to be both a younger contemporary and an older contemporary. This is not a perfect solution but it enables the users get the desired data.

When the phrasal lexicon covered the intended areas the test application was deemed *good enough* to be submitted to testing. It was finally time to start evaluating the interface.

3.4 How to Evaluate a Natural Language Interface

It is not easy to evaluate a natural language interface since many different criteria may be employed and there is no consensus in how it should be carried out. Suggestions on how it should be evaluated include techniques such as black-box testing, glass-box testing, pen and paper-testing, statistical evaluation of the number of correctly answered queries, random query generation, adequacy testing, separate component testing, use case scenarios, test suites, habatability studies, comparative studies with other interfaces and several variations of what can only be described as ad-hoc testing[10, 11, 19, 28, 30]. It seems that there are almost as many ways to evaluate a system that there are systems. One thing that is true is that two natural language systems are probably incomparable since they are aimed at achieving different capabilities. Where they aim at the same capabilities, one may exhibit better performance while the other experiments with methods of greater potential. It is often possible to tune a system specifically for a explicit criteria, thereby disguising its real shortcomings[11].

Since there is not enough time to construct a long term evaluation, such as a habatability test other methods had to be found. Among the most common and for this application suitable criteria to evaluate a natural language interface are the ones suggested in [30], namely coverage, learnability, general software criteria and comparison with other interfaces.

Coverage is a comparison between the set of inputs that the system can and should be able to handle. Learnability is a measurement on how well new users are able to identify coverage limitations and operate within them while carrying out a task. The general software criteria is concerned with a wide area of measurements like speed, size, portability, modifiability, installation, maintenance, etc. Comparisons with other interfaces are made to see how suited a system is for a specified domain. The evaluation is often based on parameters such as how long it takes for a user to get a correct answer to a task or the number of queries per task.

Chapter 4

Testing the Application

This chapter will present how and why the studies were performed and why some of them was not performed. The results of these studies will also be presented to facilitate the drawing of conclusions.

4.1 Coverage

Coverage is a comparison between the set of inputs that the system can and should be able to handle. A system should at least cover it's specified domain.

4.1.1 The Study

To be able to ensure that the phrasal lexicon has good coverage and to see where the language went beyond the capabilities of the lexicon a *Wizard of Oz* test was performed.

A Wizard of Oz test is a test were subjects are led to believe that they are using a natural language interface but in reality they are only using a simulated system. The system is simulated by replacing the system with a person that answers the queries instead. This person, called the wizard, ensures that the frailty of the current system does not affect the subjects. Wizard of Oz studies also works around the problem with written tests where the users tend to write differently than if they are using a interactive interface. The input that the users gave to the simulated interface is later given to the real interface to find relevant information not covered by the interface.

In this study the subjects were told that they were interacting with a natural language interface but a chat program was hidden behind the graphical interface and a wizard was simulating the database and the rest of the application. The wizard tries to answer the given queries as well as possible and informs the user if they are leaving the specified domain. Since the answers from the wizard is not of interest in this study they will be omitted from this report. The queries the subject submitted to the system was analyzed to find questions, data and phrases not covered by the prototype system.

4.1.2 The Subjects

The test of this interface was done on a group of 42 people with various backgrounds. Most of them had English as their native language but some of them merely stated that they had decent knowledge of English. The age of the subjects ranged approximately from ten to fifty years old and they all had very different experiences in using computers. None of the subjects claimed to have had any experience with natural language interfaces what so ever. To get a wider perspective of how the domain was used both people with very little knowledge of Swedish history and those with vast knowledge participated. This study was performed by first giving them a very brief demonstration of the interface and then letting people ask anything they would like to know about Swedish history.

4.1.3 The Results

The test resulted in 361 different queries that were not covered by the actual application. These failed questions can be divided into five different categories, linguistical errors, out of domain errors, underspecified expression errors, unlisted synonym errors and conjunction errors.

Excerpts from what the users gave as input in the Wizard of Oz study can be found in appendix B.

Linguistical errors

The largest category, with 128 failed queries, was because the user made linguistical errors. The questions were not correctly parsed because people misspelled words or made grammatical errors such as using the wrong preposition. This category was initially slightly larger but some of what was first believed to be linguistical errors turned out to be shortcomings of the phrasal lexicon. When this was discovered the phrasal lexicon was adjusted accordingly. It is not very difficult to draw the conclusion that the STEP system would benefit greatly from having a spell and grammar checker.

Outside domain errors

The second most common reason to why a query was not answered correctly is because it was outside of the given domain. 112 queries were outside of the domain, and about half of these were not related to Swedish history at all. The queries that were relevant to the domain but not covered were questions that are not suited for database application of this type. This could be because there are several answers or the answer is so vague that it is hard to describe. To give an example of these problems we will discuss a question that appeared in the Wizard of Oz test and how the STEP responds to the input.

>*"How much older is Sweden than the U.K.?"*
```
-"The query 'How much older is Sweden than the U.K.' does not parse."
```

This is not an easy question to answer since it is very hard to determine when Sweden was born. Are we talking about the nation[1], state[2] or the country[3] Sweden? Even if we know which one of it we are referring to it is not easy to answer since there are almost as many opinions to when Sweden became Sweden as there are Swedish historians. But if we supposedly solved this problem we still need to find a date as to when the United Kingdoms was born.

[1]A nation is an imagined community of people created by a national ideology
[2]A state is a geographic political entity possessing political sovereignty
[3]A country is a geographical area and an independent political entity

Unfortunately history is a domain with many of these kinds of questions and hence is an ill suited domain for the type of database application developed. The best way to solve this problem would probably be to complement the STEP system with an encyclopedia type application that is better suited to give lengthy explanations and theories that answers these difficult queries. A longer discussion on why it is hard to represent history in a database can be found in section 5.2.

There was one query outside of the domain concerning common denominators between persons that was particularly interesting. What makes it interesting is that even if it would have been within the domain this question would have been hard to answer because of limitations in SQL. The query looked like this:

>*"What connection is there between Sigismund and Karl IX"*

This query is very general and could both refer to that they were brothers, lived during the same time or that they both were involved in the battle of Stångebro. It would have been a nice feature to answer if there are any kind of relations between two entries but since SQL always relies on the user to specify the relationship when executing a query this is impossible to do in a general way. To solve this problem it could be possible to write complex stored procedures but it would likely need to specify all tables in a tedious way or try to handle it outside of the database which also suffer from the same problem. There are people that have made an effort into solving the problem of unspecified relationships by using an intermediate language between the database and the application. One of these interesting projects is the Hydra project which is a functional language that uses graphs to represent the data[3].

Underspecified sentences errors

The third most common reason, with 86 failed queries, was the use of underspecified sentences often when trying to refine a query. Refinement of a query is when the users are trying to query over a set of answer given from the previous query. Querying over the results from a previous set and the use of anaphoric and elliptical sentences are currently not supported by the STEP interface since it has no knowledge of previous queries. To give an example of how this may look I use an example that occurred in the Wizard of Oz test and how STEP responds to it.

>*"Give the persons involved in the assassination of Gustav III"*
```
-"According to the database, the 3 persons involved in Assassination of
Gustav III are Carl Fredrik Pechlin, Gustav III and Jakob Johan
Anckarström."
```
>*"When did they die?*
```
-"There is no person called they in the database"
```

At first glance one would think that this kind of feature is not important in a natural language interface since it does not reduce the amount of answers the application can provide, but this is not true since it greatly reduces the number of queries the interface accepts. Still the example above would have yielded a correct answer if the user would have written *"When did <person> die?"* instead, where person is substituted for each of the results of the previous query. This leads to that one could argue that this feature is just something for the the users convenience. Even if it is for user convenience it does not change the fact that natural languages uses many tricks, including these, just because of user convenience.

Unlisted synonyms errors

The fourth reason for failed parses, with 25 queries, was the use of unlisted synonyms and phrases. To solve this problem STEP has support for using the lexical reference program Word-net[4]. Word-net has English nouns, verbs, adjectives and adverbs organized into synonym sets, where each set represents one underlying lexical concept. When a word is not encountered in the phrasal lexicon, synonyms for this word is looked up with Word-net and tested again to see if it makes more sense.

It is somewhat surprising that this error is so uncommon. It was believed that the phrasal lexicon was going to be huge after the Wizard of Oz test but actually there where only a few additions that had to be made. It seems like that the subjects we tested used a similar set of synonyms, even though they had very different backgrounds.

Conjunction errors

The fifth reason for failed parses was the use of conjunctions in queries. Ten of the queries contained the words *and* or *or* to bind together two separate queries. The usage of these conjunctions are very tricky since they often generate ambiguous sentences which are hard to resolve. An example of this is the question *"Give the average salary of the marketing and the sales department?"* It is in this question not obviuos if the user want the average salary for each departments separatly or together.

A way to resolve these ambiguities have not been incorporated in STEP and hence it may sometimes generate an incorrect answer. How to resolve these problem is not a trivial matter and the best way would probably be to paraphrase the query back to the user for disambiguation, just like STEP handles other ambiguities.

4.2 Learnability

Learnability is a measurement on how well new users are able to identify the limitations of the system and operate within them. To perform a learnability study and actually measure how fast and well new users identify and adapt is not easy. I started to make plans on how to evaluate this but quickly found out that I do not know how to perform such a study. The few methods to perform a learnability test on natural language interfaces that I could find were all either too time consuming to fit in a master's thesis or too abstract and non-conclusive. This has led me to not perform a learnability test but I would still like to say a couple of words about how the users adapt to the STEP system. By looking at how users behaved in the comparative study (see section 4.4) it seems that if users encounter a limitation in STEP on the first use of a certain type of query they tend to avoid that kind of phrasing again. If they on subsequent uses of a type of query encounter a limitation they are not as eager to abandon this type of query, even if they discover that it only works erratically. There are still signs of decreasing usage of a type of query but not as absolute as if it failed on the first attempt.

Even though quick adaptation is a good thing, to cease completely after one query is often bad. Examples of too quick adaptation can also be found in the comparative study. The most common example is when the user mistakes one type of error for another, such

[4] `http://wordnet.princeton.edu/`

as a spelling mistake for a limitation in the coverage of the application. The user does not notice the spelling mistake and determines that the system has a limitation and falsely adapts to it.

In general it can be said that users adapt very quickly to the STEP interface but unfortunately they sometimes falsely adapt. The application is somewhat prone to errors and it seems that even lengthy use of the interface does not reduce the error rate significantly.

4.3 General software requirements

The general software criteria is concerned with a wide area of measurements like run-time speed, program size, portability, scalability, modifiability, robustness, installation, maintenance, etc. The most interesting measurement would be to compare the STEP system with other natural language interfaces, but since I have only been able to get my hands on very few other natural language systems, these tests can not be conducted with accuracy.

When comparing STEP with other natural language interfaces empirical tests suggest that STEP is more robust and uses less storage space than other applications, but is in some cases slower than other interfaces. This slower response time mainly occurs when STEP fails to parse a query and needs to rephrase it to to derive its meaning. Overall the run-time performance is reasonable, even with the optional application word-net slowing it down. Word-net may reduce the performance but in my opinion a fair trade-off for a system with better coverage. The online application also answers questions fairly quick, but if several users query the interface simultaneously there is a noticeable drop in performance. Whether this bottleneck is because of the application, database access, network or hardware limitations have not been concluded. The response time to answer a query scales logarithmically with the size of the phrasal lexicon, which is believed to be quite good but since the scalability of other natural language interfaces is not known it is hard to determine exactly how well it scales.

STEP runs on several different platforms such as Windows, LINUX, OS X and UNIX but requires CLISP compiled with FFI[5] support as well as the SPASS[6] Automated Theorem Prover. This is better than many other systems that are locked to a single operating system. STEP also allows connections to databases via ODBC[7] so a wide variety of different relational databases can be used. All together STEP can not be said to be platform independent, but it runs on the many common operating systems.

4.4 Comparison with a Form-Based Interface

Perhaps the most interesting test that could be performed on a natural language interface is to compare it to other more widely used interfaces. Will this tell us why form-based interfaces are so much more popular than natural language interfaces?

[5]Foreign Function Interface
[6] `http://spass.mpi-sb.mpg.de`
[7]Open Database Connectivity

4.4.1 The Study

The focus of this study was to measure how well a group of testsubjects were able to perform two dozen tasks using the STEP interface and a form and menu-based interface. The forms and menu-based interface was written in PHP and utilized the standard methods PHP provides. Twelve of the tasks were to be performed with the STEP interface and twelve with a standard form and menu-based interface. Which twelve questions to answer using a certain interface was randomly determined for each subject. Each set of tasks contained six easier task which could be performed with a very straight forward query and six more elaborate tasks which required more complicated methods. An representative example of an easier task would be *Using the natural language interface produce a list of all Swedish regents* while a more elaborate task could be *Using the form-based interface produce a list that only contains persons which fulfills all of the following criteria. They were/are the regent of Sweden. They were at some time involved in a major event that occurred in the city of Stockholm.*

To measure how well the subject performed a task the following criteria was observed for each of the subject.

Correctness If the subjects produced the correct results for each task

Search time How much time the subjects needed to perform each task

Queries needed The number of queries needed to perform each task

Failure rate How often the interface failed to answer a query

4.4.2 The Subjects

The selection made to test this kind of interface was done on a diverse group of people with the common denominator that they all live or have lived in a country that have English as their native language. The age of the subjects range from 19 to 68 years and their experience using computers vary greatly. 14 out of the 16 subjects were all familiar with basic computer usage such as using word processors and reading e-mail. A further 7 of them had a wider use of computers such as office applications, spreadsheets or programming. Out of these six subjects three had some experience using databases and all of these had some experience designing their own database. None of the subjects claimed to have had any experience with natural language interfaces what so ever, which also means that none of them had participated in the Wizard of Oz test. Five of the subjects in this study had extensive knowledge in Scandinavian history while the other eleven had none or very limited knowledge.

Before the actual test started they each received a brief introduction to how each of the two interfaces worked. This demonstration was very basic and mostly aimed at those with limited computer experience. During this demonstration no features or limitations what so ever were discussed.

4.4.3 The Results

The results from this study varied very much for each subject and each query and I am certain that if this test was performed with a form-based interface more optimized for this specific application the results could be quite different. Regardless of this there is no doubt that the form-based interface clearly outperformed the natural language interface for all of the simpler tasks. For the more elaborate tasks the results are not as conclusive.

For the simpler tasks all of the subjects managed to produce the correct results for both of the interfaces, which shows that even the subjects that had very little or no computer experience managed to use the two interfaces correctly. When looking at the search-time the forms-based interface was somewhere between three to five times faster than the natural language interface for most users. The cause for this can be explained with that it takes longer to type a query and that some subjects needed more than one query to perform the task. In average the subjects needed 1.08 queries for the form-based interface and 2.84 queries for the natural language interface.

The form-based interface never failed to perform a query but around one out of six queries failed in the natural language interface. All of these failures were because of spelling-mistakes from the users.

When it comes down to the more elaborate tasks the results are not as simple to interpret. Every task except three were correctly performed and these erroneous answers were solely because of subjects misunderstood the task or performed another task than they intended. Disregarding these erroneous tasks it seems like all users were equally adept in using the form-based interface and required around four to seven queries to perform a task.

When reviewing the results for the natural language interface the subjects can be divided into two equally large but quite different groups, those who performed well and those who did not. Those who performed well only required one to three queries to perform a task and did so in less or equal the time they needed with the form-based interface. What distinguishes this group from those that did not perform well was that they made complex queries instead of several simpler ones. The interface failed to answer one out of three queries and most errors were still grammatical or spelling errors. The rest of the errors were because of lacking linguistical coverage in the phrasal lexicon and the interface limited ability to handle conjunctions. Because of the ambiguous nature of conjunctions the natural language interface did at one time even give an incorrect answer to a query. Luckily the subject that received this answer noticed the unreasonable answer and rephrased his query.

The group that did not perform that well first made a lot of smaller queries which they later tried to refine into one final query. Subjects in this group needed between seven and twenty-seven queries to perform a task. They all performed their tasks much slower, averaging around ten times slower, than they had with the form-based interface. The error rate were much less than in the group that did well and the queries failed one out of seven times. Disregarding two errors because of synonyms unlisted in the phrasal lexicon all errors were because of spelling mistakes.

After the subjects had performed all tasks a few questions were asked to get their opinion on how they experienced the natural language interface. Most users thought that the natural language interface was too prone to errors and it was unclear where the limitations of the interface was. No subject believed that the natural language interface would replace the form-based interface as the de facto standard but most of them believed that it could become a valuable compliment. Especially if features such as the ability to refine queries and a spell checker was incorporated.

Those who benefited from the interface were all positive to this kind of interface since they could perform complex queries in simple ways. Some of those that did not perform so well using the natural language interface had some interesting explanations to why. One of the subjects expressed this very nicely when he said: *If I want to list* *all the contemporaries of Gustav Vasa, I don't know how to express this in a natural way.* This might suggest that some users does not think that it is natural to use natural language with a computer.

Chapter 5

Conclusions

This chapter will emphasize some of the conclusions that can be drawn. The potential and future of STEP and other natural language interfaces will also be discussed.

5.1 Will STEP Replace other Interfaces?

After having tested STEP the obvious questions is if STEP will some day replace other interfaces. The short and simplified version of the answer is no, not as the de facto standard interface. The main reason for this is that natural language interfaces perform worse than graphical and form-based interfaces in the average case.

5.1.1 Why Develop a Natural Language Interface?

But if not as the de facto standard is there a need for natural language interfaces? I believe there is still hope for natural language interfaces but mainly in specific applications that have properties where other interfaces do not suffice. Minock suggests that an application where natural language interfaces should outperform interfaces must be circumscribed, complex and practical[17] and I must agree. Circumscribed means that the domain is focused with an obvious level of detail and representable factual data. Complex means that it has numerous entries that span more than one concept and have complex properties and relationships. Practical means that there are users that find this domain interesting and would query the domain often enough to ensure an interest in developing and maintaining the application. The application developed to test history is very complex, but it seems to be impractical and is to fuzzy to be well described by a database, hence it is not circumscribed. More on why it is not circumscribed and why it is hard to represent history can be found in section 5.2. The application could, with alot of effort, become practical by adding a lot more information to the database, but in my opinion it is not worth the effort.

As stated I believe there is a need for natural language interfaces. It is just a matter of finding the right application. This however does not say if there is a need for STEP. STEP is a natural language interface that performs well when compared to those natural language interface I have had the opportunity to test. It may not have all the features some of these systems have, but it makes up for this by being much more robust. STEP is also fairly easy to port to another domain and the system can be run on several major operating systems. There are some problems with using step e.g. the inability to express some queries in the phrasal lexicon, but this is mostly because STEP is not fully developed. To better evaluate STEP one should look at the potential of STEP rather than its current performance.

5.1.2 How Could STEP be Improved?

STEP is currently solid and portable, but there are some several improvement that would be desired. *In my opinion the single practical improvement that STEP would benefit from the most is to include a spell checker.* Without a doubt linguistical errors such as spelling mistakes or the use of the wrong preposition is the most common reason to why STEP fails to parse a query. Based on the studies I believe that if STEP had a spell checker the number of failed parses could be cut in half. If the spell checker also had the ability to check grammatical errors such as omitted and redundant words or the choice of preposition it would be even better. Some demands must be made on this spell checker, it must be very fast, since it will be called often, and it must have a fairly large vocabulary, that even recognizes names, places and domain specific terminology. Unfortunately no known spell checkers, openly available today, are fast and complete enough to be integrated into STEP.

Another more conceptual feature that I believe would really benefit STEP is better support for conjunctions. Even though not many users used conjunctions during the Wizard of Oz test there were many subjects that requested that feature. They wanted to make ad-hoc sets exactly like in Walkers and Whittakers study[29]. There are some support for this in STEP but it is not a really well developed part of the system. Most of these subjects wanted to use other logical operators than *and* in their queries. This lack of logical operators is also evident when writing phrasal lexicon where all natural language queries can not be described with a single query in the phrasal lexicon.

Most subjects that did not use conjunctions in their querying instead used anaphora and/or elliptical sentences. They wanted to make queries on the result of the previous query, which is a nice feature for users to have. By being able to query the result of a previous query it to overcome some of the problems with the lack of support for conjunctions. Unfortunately this is not supported by STEP either. Even though both of these features would require a lot of effort I believe that it would be worth it. At some time one of these features should be available in STEP, preferably both.

Something I believe could lead to a greater usage of natural language interfaces is if end-users had the ability to *teach* the system via the natural language interface. Special interest groups that wishes to keep a database could use it for their online-application. Since special interest groups often have many skilled persons in their area of expertise but often lack from too few administrators a natural language interface would be really benefiting. The maintenance would be less for the administrator since users help with this task. The database would still be queriable and probably be more up to date. It could become a good compliment to wiki web applications, that works on the same principle and is popular among online special interest groups. There is two ways of *teach* the system via the natural language interface. One is to make it possible to alter the linguistic knowledge of the system and one is to alter the database contents. Altering the database contents is a feature that is under development in STEP and hopefully this will make the interface more widely used.

In addition to these there are other features that could be added to STEP. This includes features such as better temporal support, support for special characters, the usage of CYC to generate better out-of-domain answers, the ability to answer if there is any kind of connection between two attributes and the inclusion of encyclopedia application, when needed answers a query instead of the natural language generator.

These features are interesting and should probably be included with STEP some day, but these are not critical features. They should not be added before the other features are incorporated or until there is applicational need to have them.

Overall I believe that STEP is a system that works well and that has a lot of potential. There are some bugs to work out and some features to add, but other than that it is just a matter of finding an application for STEP. This application is unfortunately not history, but is any queriable database application suitable for history?

5.2 Can History be Described in a Database?

As mentioned in section 3.1 there are limitations to the record-based information model. Kent describes many of these problems in [12] and even though there have been advances in the practical applications, many problems remain. The main problem is when there is too much variation of the facts within a table. This creates a need to add more fields or tables to accurately describe the item. Kent[12] uses a table of clothing as an example for this. To us it is quite easy to argue what items belong to the set called clothing, but it is not easy to come up with a definition for a database table. There are many relevant fields but not many of them apply to more than one type of clothing. The relevant field values could be size, waist size, length, color, pattern, neck size, sleeve length, button or zipper, fabric, heel size, neckline, washable, number of pockets, cup size, waterproof, weight, casual or formal, sports ware and much more. Even though it can be broken up into several other tables and then joined together, it would not be an easy job of designing these tables and it would be far from efficient.

The history domain suffers from this problem and the fact that the facts in each field are often not iron-clad but rather a fuzzy hind-sighted interpretation of the real events. These interpretations can actually be quite far from what actually happened since history has been rewritten to suit the current political landscape better. This makes history a dynamic domain that needs continuous reconstruction rather than the static domain that one would first believe it to be. Not stopping at this, history also has very vague concepts. Take a expression such as "Sweden had control over Norway" which tells us that the nation Sweden controlled the nation or region called Norway, but not how. Was there a union between them? Was Norway a vassal state? Were Norway conquered by force? Where they simply considered a part of Sweden? Did the people of Norway regard themselves as a part of Sweden? Was it a financial control? Was it a control by political influence? Are all of these perhaps true?

I have only listed a few common interpretations of the expression *to have control over another nation*. This is a variation of homogeneity problem mentioned above, but a bit worse since there is no obvious subset, such as shoes or hats in the clothing example, that share common attributes. To make matters even worse one would like to have an easy way of continuously updating the facts since history is a dynamic domain and tables would likely be added, removed or modified quite often. This is hardly the only example of its kind when designing a history database. Nevertheless it is still possible to model this in a database but it will be a bit complex. There will be many tables in the database and a lot of triggers and other stored procedures to facilitate the maintenance of the database and the performance and maintainability of the database would be adversly affected.

There is an even harder problem regarding represention of queries and *why* something occurred as they did. It is impossible to answer this kind of causality question *completely* since that would probably require a complete model of the universe at every point in time. It is easy to argue that this is required with a little help of Edward Lorenzs butterfly effect. It is possible that the flap of a butterfly could have started a chain of events that in the end influenced the outcome of the thirty years' war and hence could be argued to be a vital part of history. Even though it was contributing factor it is still highly unlikely that someone would like to have the query *What decided the outcome of the thirty years' war?* with *A butterfly flapped its wings.*

This means we are not looking for the complete solution to all queries since it is impossible to describe everything. The question is rather if a *adequate* solution can be found. This is still not an easy problem since every major event in history seem to have at least a dozen books about them, with the authors interpretation of why something happened. Much of this is speculations and unfortunately it is not uncommon that they contradict each other. Since a database has no opinion of its own, the creator of the database will either decide what is true or all answers will have to be added to the database. Since no human is infallible the later is probably more reasonable. The best way to solve this problem with multiple contradicting answers would probably be to present the answer in the form of an encyclopedia. Encyclopedias are well suited to give lengthy explanations that can emphasize on what are certain and what are theories. To fill the database with data, much planning must go into the design of tables and it will require a lot of historical expertise.

To complete an *adequate* queriable database of history one would need alot of work with persons both skilled in history and in database management. The data would to be constantly rewritten to suit the latest theories and to more clearly define what probably happened. Because of the large number of tables it will probably be hard to ask queries over the database in SQL so a simplified interface of some kind must be constructed. Given enough time and dedication it is probably possible to have an adequate database over history, but to be perfectly honest I believe that the effort would probably be greater than the benefit.

Chapter 6

Acknowledgments

I would like to gratefully acknowledge the enthusiastic supervision of Michael Minock during this work. I thank Jonathan Edelstam for the help in performing the studies. Truly a man who helps a friend in need. I would also like to thank all those persons that gave up some of their time to participate in the studies. Without your help there would have been no studies and no thesis. David Pierce is thanked for his help with CLISP related issues and for that I was allowed to use portions of the code he had written.

I am grateful for all my friends at Umeå University, without your moral support I would probably have finished the thesis earlier, but would had much less fun. I am particularly thankful for those friends and others that read through this report and corrected all the stupid mistakes I made. My dear parents should also receive my gratitude for all their emotional and financial support.

Finally I would like to thank all friends from the non-profit association Lah-Dih-Dah that with their insane pranks made me look sane. I kept my end of the bargain so now you all know what to do.

References

[1] Hiyan Alshawi, editor. *The Core Language Engine.* ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge, England, 1992.

[2] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.

[3] Robert Ayres. The functional data model as the basis for an enriched database query language. *J. Intell. Inf. Syst.*, 12(2-3):139–164, 1999.

[4] Joseph D. Becker. The phrasal lexicon. In *TINLAP '75: Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 60–63, Morristown, NJ, USA, 1975. Association for Computational Linguistics.

[5] Johan Boye, Joakim Gustafson, and Mats Wirén. Natural Language Understanding for the NICE Fairy-Tale Game, 2004.

[6] Philip R. Cohen. The role of natural language in a multimodal interface. In *UIST '92: Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 143–149, New York, NY, USA, 1992. ACM Press.

[7] Michael T. Cox and Ashwin Ram. Introspective multistrategy learning: on the construction of learning strategies. *Artif. Intell.*, 112(1-2):1–55, 1999.

[8] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[9] Michael George Dyer. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension.* MIT Press, Cambridge, MA, USA, 1983.

[10] Julia R. Galliers and Karen Spärck Jones. *Evaluating Natural Language Processing Systems.* "Springer-Verlag", "Berlin, Heidelberg", 1995.

[11] Jerry R. Hobbs. Some Notes on Performance Evaluation for Natural Language Systems. unpublished manuscript, July 1986.

[12] William Kent. Limitations of record-based information models. *ACM Trans. Database Syst.*, 4(1):107–131, 1979.

[13] Lillian Lee. I'm sorry Dave, I'm afraid I can't do that: Linguistics, statistics, and natural language processing circa 2001. In Committee on the Fundamentals of Computer Science: Challenges, Computer Science Opportunities, and National Research Council Telecommunications Board, editors, *Computer Science: Reflections on the Field, Reflections from the Field*, pages 111–118. The National Academies Press, 2004.

[14] Wendy Grace Lehnert. *The process of question answering.* PhD thesis, Hillsdale, NJ, USA, 1977.

[15] Kavi Mahesh, Sergei Nirenburg, Jim Cowie, and David Farwell. An assessment of CYC for natural language processing.

[16] Michael Minock. A phrasal approach to natural language interfaces over databases. Technical report, Umeå, Sweden, 2005.

[17] Michael Minock. Where are the 'killer applications' of restricted domain question answering?, 2005.

[18] Michael Minock. Modular generation of relational query paraphrases. *Formal Aspects of NLG (Special Issue of the Journal of Language and Computation)*, to appear 2005.

[19] Martha Palmer and Tim Finin. Workshop on the evaluation of natural language processing systems. *Comput. Linguist.*, 16(3):175–181, 1990.

[20] Rafal Pietrowski. Natural language interface to legal databases. Master's thesis, University of Umeå, Umeå, Sweden, 2005.

[21] R. C. Schank and R. P. Abelson. Scripts, plans, goals and understanding. In A. Collins and E. E. Smith, editors, *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*, pages 190–223. Kaufmann, San Mateo, CA, USA, 1988.

[22] Vijay Sethi. Natural language interfaces to databases: MIS impact, and a survey of their use and importance. In *CPR '86: Proceedings of the twenty-second annual computer personnel research conference on Computer personnel research conference*, pages 12–26, New York, NY, USA, 1986. ACM Press.

[23] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[24] Steven P. Shwartz. Problems with domain-independent natural language database access systems. In *Proceedings of the 20th annual meeting on Association for Computational Linguistics*, pages 60–62, Morristown, NJ, USA, 1982. Association for Computational Linguistics.

[25] Harry R. Tennant, Kenneth M. Ross, Richard M. Saenz, Craig W. Thompson, and James R. Miller. Menu-based natural language understanding. In *Proceedings of the ACL-83, 21st Annual Meeting of the Association for Computational Linguistics*, pages 151–158, Cambridge, US, 1983.

[26] Richmond H. Thomason. Logicism in formalizing common sense and in natural language semantics.

[27] Bozena H. Thompson and Frederick B. Thompson. Introducing ASK, a simple knowledgeable system. In *Proceedings of the first conference on Applied natural language processing*, pages 17–24, Morristown, NJ, USA, 1983. Association for Computational Linguistics.

[28] Valerie B. Barr and Judith L. Klavans. Verification and Validation of Language Processing Systems: Is It Evaluation? In *ACL 2001 Workshop on Evaluation Methodologies for Language and Dialogue Systems*, pages 34–40, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[29] Marilyn Walker and Steve Whittaker. When natural language is better than menus: A field study. Technical Report HPL-BRC-TR-89-020, HP Laboratories, Bristol, England, 1989.

[30] Steve Whittaker and Phil Stenton. User studies and the design of natural language systems. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, pages 116–123, Morristown, NJ, USA, 1989. Association for Computational Linguistics.

[31] R.N. Kaplan Woods, W.A. and B.N. Webber. *The Lunar Sciences Natural Language Information System: Final Report*. Cambridge, MA, USA, 1972.

[32] Robert Østberg and Matthew Piziali. Natural language questioning at IBM, 1978.

Appendix A

EER-Diagram

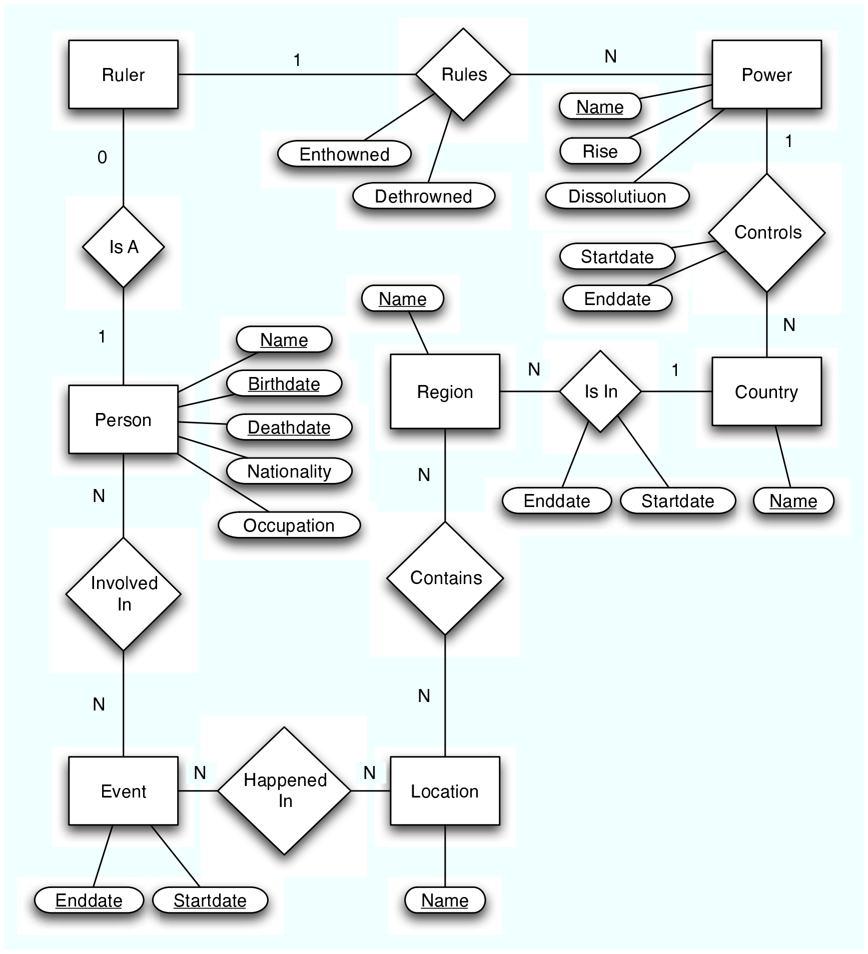

Figure A.1: A slightly simplified version of the EER-diagram

Appendix B

Excerpts from the Wizard of Oz Study

This is a excerpt from the Wizard of Oz study. This is not all of the queries but an interesting selection. This means among other things that all unrelated, immature and duplicate queries has been removed.

Show system capabilities
What can you tell me?
List database tables
How much older is Sweden then the U.K.?
Show all wars Sweden has been involved in
Show Swedish inventions
When was Sweden and Russia at war
List Key figures of Sweden
Give the persons involved in the assassination of Gustav III
When did they die?
Give the persons involved in the assassination of Gustav III except Gustav III
Show me the kings of Sweden and Norway
Who was thte first king of Sweden?
Give information on Alfred Nobel
Who was Gustav Vasa
Show events in Swedish History
When did they occur?
Lst king of Sweden
Give swayers of Sweden
Show presons born after 17000101
List persons involved in Stockholm Bloodbath
Who ruled Sweden during WWII?
Who controlled Swedish Pommerania in 18090101
Give birthdate for Gutav Vasa
Give countries that have had control over Skåne
Show events that occured in Västerås
Give contemporaries of Alexander I
What was the Battle of Ratan and Sävar?

Gvie Persons alive during Stockholm Bloodbath
Which regent swayed Sweden in 18000101
Give rulers that ruled between 17000101 nad 18000101
List nations in the database
Have the U.K. Been at war with Sweden?
Give nation that controls Swedish Pomerania
When did Sweden loose control over Finland?
How long did Sweden control Finland?
When did Finland become independent?
Where is Sweden?
Who rules Sweden?
List regions that have been a part of Sweden
List all battlefields in Sweden
List Events thta occured during the rule of Gustav Vasa
When did Stockholm Bloodbath occur??
What happened at Stockholm bloodbath?
When did the Dacke Feud occur?
Who was Nils Dacke?
Tell me about Sweden
Which persons was involved in the Thirty Years War
Only show the persons from Sweden
When did the Bernadottes become Swedish rulers?
When did Jean Baptiste Bernadotte reign in Sweden?
how long did Jean Baptiste Bernadotte reign in Sweden?
Who killed Gustaf III?
What connection is there between Sigismund and Karl IX
Give the Events in the database
When did the Dacke feud occur?
Give the date that the Dacke Feud occured.
What was the Dacke Feud?
List the persons responsible for the Dacke Feud
Give information on Nils Dcaek.
Give information on Nils Dacke.
List all Swedish reble leaders.
Give contrmporaries of Axel Oxenstierna
Who is Vladimir Putin
How does Vladimir Putin relate to Swedish History?
Give Predesessor of Karl IX
Give Predecessor of Gustav V
Give rulers of Sweden or Denmark
Exclude those born before 17000101
Include those born before 15000101
List rulers involved in an event that occured in Stockholm
Show me all important persons in Sweden
Show me all important events in Sweden
List the five latest rulers of Sweden
Give persons born on the 2nd of April
Give ruler with the shortest lifespan

Appendix C

FST

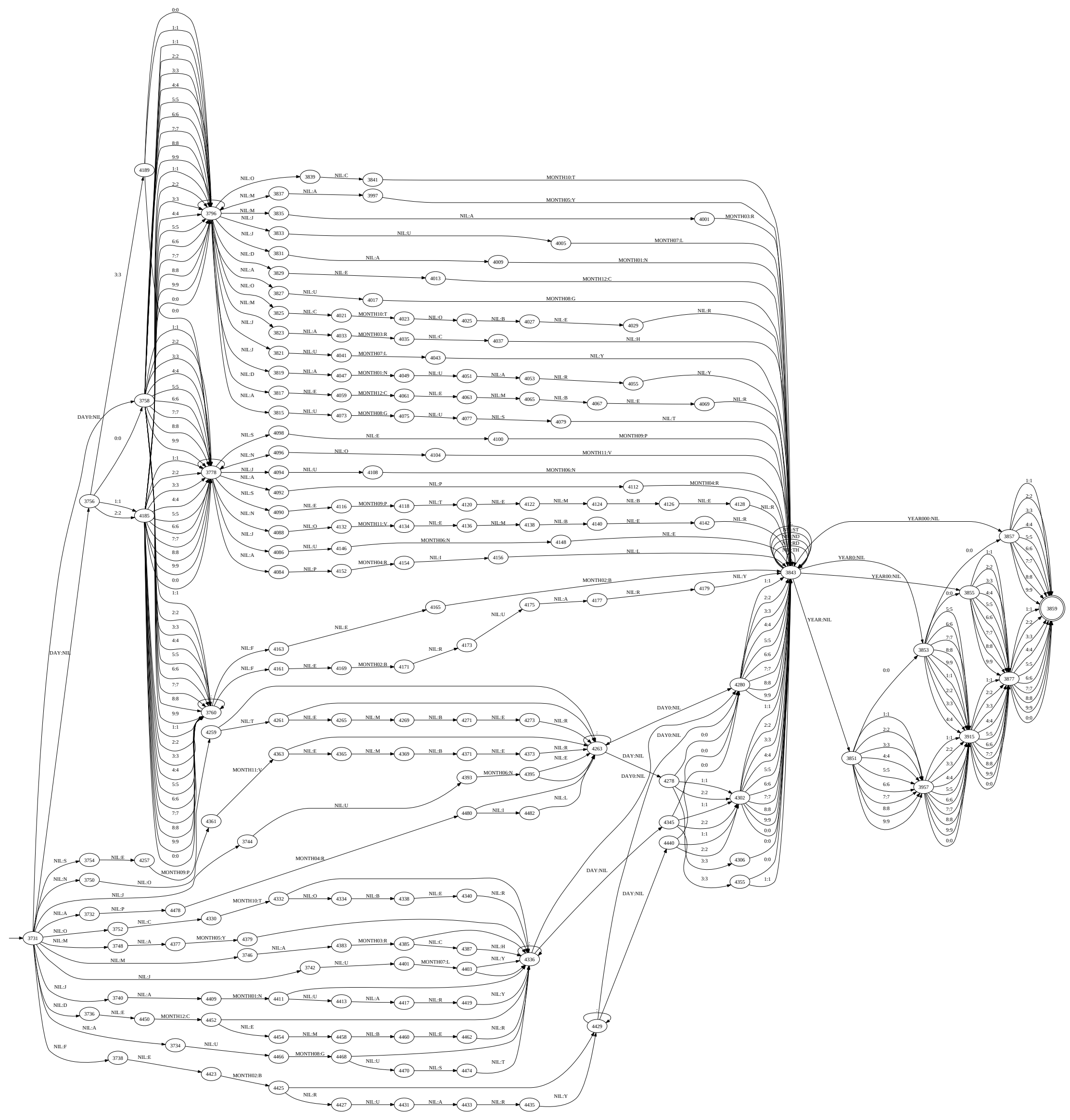

Figure C.1: A visualization of the FST parsing alphabetical date formats

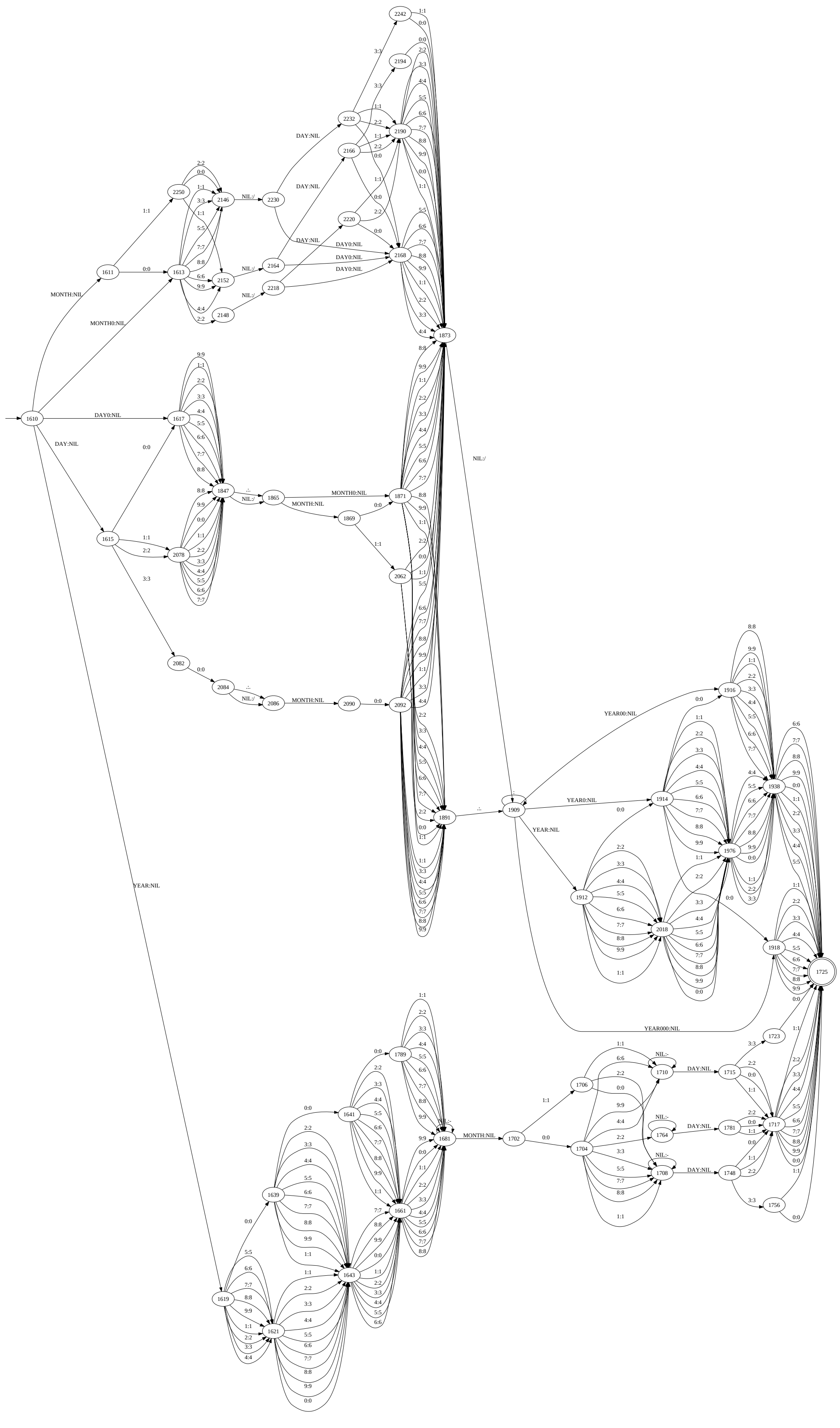

Figure C.2: A visualization of the FST parsing numerical date formats

Appendix D

Configuration of the Application

```
(set-server
:host "aristoteles.cs.umu.se"
:port 18001)

(set-db
 :name "History"

 :description
 "This database is a first small test with swedish history as a topic."

 :schema
 '((power (name VST VET) :key (name) :date(VST VET) :source (name))
   (country (name) :key(name) :source(name))
   (region (name) :key(name) :source (name))
   (location (name) :key (name) :source (name))
   (controlsCountry (power country VST VET) :date(VST VET)
    :source(power country))
   (containsRegion (country region VST VET) :date(VST VET)
    :source(country region))
   (containsLocation (region location) :key(region location)
    :source(region location))
   (person (name birthdate deathdate nationality occupation)
    :date (birthdate deathdate) :key (name birthdate deathdate)
    :source (name nationality occupation))
   (ruler (name birthdate deathdate enthrowned dethrowned power)
    :key (name birthdate deathdate power) :source(name power)
    :date(birthdate deathdate enthrowned dethrowned))
   (event (name startdate enddate) :key (name)
    :source(name) :date(startdate enddate))
   (happenedIn (event location region) :key(event location region)
    :source (event location))
   (involvedIn (eventName person birthdate deathdate)
    :date(birthdate deathdate)
    :source(eventname person)
    :key (eventName person birthdate deathdate))

   (capital (id) :dummy)
   (happenedInTheYear (id) :dummy)
   (invention (id) :dummy)
   (have (id) :dummy)
   (successor (id) :dummy)
   (predecessor (id) :dummy)
   (father (id) :dummy))

  :connection
  (odbc-connect :database "history" :user "history"
                :library "/usr/lib/libiodbc.so.2")

  :example-queries
  '("Give rulers who ruled Sweden"
    "Give rulers born after 15000101"
    "Give rulers that ruled between 15230101 and 16000101"
    "Give rulers that was involved in Stockholm Bloodbath"
    "Give events that involved Gustav Vasa"
    "Give events that occured in Stockholm"
    "Give life spans for rulers of Sweden"
    "Give life spans for rulers of Sweden born after 15000101"
    "Give life spans for rulers involved in events that occured in Stockholm"))

(create-db
:perform nil
:facts
 '((power "Sweden" "0001-01-01" "9999-12-31")
   (power "Norway" "0001-01-01" "9999-12-31")
   (power "Denmark" "0001-01-01" "9999-12-31")
   (power "Russian Empire" "0001-01-01" "1917-10-25")
   (power "Soviet Union" "1917-10-25" "1991-12-26")
   (power "Russian Federation" "1991-12-26" "9999-12-31")
   (power "Finland" "1917-12-06" "9999-12-31")
   (power "Prussia" "1701-01-18" "1947-02-25")
   (power "Germany" "1871-01-18""9999-12-31")
   (power "Poland" "0001-01-01" "9999-12-31")

   (country "Sweden")
   (country "Norway")
   (country "Denmark")
   (country "Finland")
   (country "Russia")
   (country "Prussia")
   (country "Germany")
   (country "Poland")
```

```
(region "Skåne")
(region "Västerbotten")
(region "Östergötland")
(region "Uppland")
(region "Södermanland")
(region "Västmanland")
(region "Swedish Pomerania")

(location "Västerås")
(location "Stockholm")
(location "Linköping")
(location "Sävar")
(location "Ratan")

(controlsCountry "Sweden" "Finland" "1100-01-01" "1809-09-17")
(controlsCountry "Russian Empire" "Finland" "1809-09-17" "1917-10-25")
(controlsCountry "Finland" "Finland" "1917-12-06" "9999-12-31")

(controlsCountry "Sweden" "Sweden"  "0001-01-01" "9999-12-31")

(controlsCountry "Denmark" "Denmark" "0001-01-01" "9999-12-31")

(controlsCountry "Denmark" "Norway" "1395-06-13" "1814-01-14")
(controlsCountry "Sweden" "Norway" "1814-01-14" "1905-06-07")
(controlsCountry "Norway" "Norway" "1905-06-07" "9999-12-31" )

(controlsCountry "Russian Empire" "Russia" "0001-01-01" "1917-10-25")
(controlsCountry "Soviet Union" "Russia" "1917-10-25" "1991-12-26")
(controlsCountry "Russian Federation" "Russia" "1991-12-26" "9999-12-31")

(controlsCountry "Prussia" "Prussia" "0001-01-01" "9999-12-31")
(controlsCountry "Germany" "Germany" "0001-01-01" "9999-12-31")
(controlsCountry "Poland" "Poland" "0001-01-01" "9999-12-31")

(containsRegion "Sweden" "Uppland" "0001-01-01" "9999-12-31")
(containsRegion "Sweden" "Östergötland" "0001-01-01" "9999-12-31")
(containsRegion "Sweden" "Södermanland" "0001-01-01" "9999-12-31")
(containsRegion "Sweden" "Västerbotten" "0001-01-01" "9999-12-31")
(containsRegion "Sweden" "Västmanland" "0001-01-01" "9999-12-31")
(containsRegion "Denmark" "Skåne" "0001-01-01" "1658-02-26")
(containsRegion "Sweden" "Skåne" "1658-02-26" "9999-12-31")
(containsRegion "Sweden" "Swedish Pomerania" "1648-10-24" "1814-01-14")
(containsRegion "Denmark" "Swedish Pomerania" "1814-01-14" "1815-06-08")
(containsRegion "Prussia" "Swedish Pomerania" "1815-06-08" "1871-01-18")
(containsRegion "Germany" "Swedish Pomerania" "1871-01-18" "1947-02-10")
(containsRegion "Poland" "Swedish Pomerania"  "1947-02-10" "9999-12-31")

(containsLocation "Västmanland" "Västerås")
```

```
(containsLocation "Västerbotten" "Sävar")
(containsLocation "Västerbotten" "Ratan")
(containsLocation "Södermanland" "Stockholm")
(containsLocation "Uppland" "Stockholm")
(containsLocation "Östergötland" "Linköping")

(person "Gustav Vasa" "1496-05-12" "1560-09-23"
 "Swedish" "King")
(person "Erik XIV" "1533-12-13" "1577-02-26"
 "Swedish" "King")
(person "Kristian II" "1481-07-01" "1559-01-25"
 "Swedish" "King")
(person "Gustav IV Adolf" "1778-11-01" "1837-02-07"
 "Swedish" "King")
(person "Alexander I" "1777-12-23" "1825-12-01"
 "Russian" "Tsar")
(person "Vasili III" "1505-10-27" "1533-12-03"
 "Russian" "Grand Prince")
(person "Vladimir Putin" "1952-10-07" "9999-12-31"
 "Russian Federation")
(person "Alfred Nobel" "1833-10-21" "1896-12-10"
 "Swedish" "Engineer")
(person "Johan III" "1537-12-20" "1592-11-17"
 "Swedish" "King")
(person "Sigismund" "1566-06-20" "1632-04-19"
 "Swedish" "King")

(ruler "Gustav Vasa" "1496-05-12" "1560-09-23"
 "1523-06-06" "1560-09-23"
 "Sweden")
(ruler "Erik XIV" "1533-12-13" "1577-02-26"
 "1560-09-29" "1568-09-29"
 "Sweden")
(ruler "Johan III" "1537-12-20" "1592-11-17"
 "1568-09-29" "1592-11-17"
 "Sweden")
(ruler "Sigismund" "1537-12-20" "1592-11-17"
 "1592-11-17" "1599-07-24"
 "Sweden")
(ruler "Sigismund" "1537-12-20" "1592-11-17"
 "1587-09-18" "1632-04-19"
 "Poland")
(ruler "Kristian II" "1481-07-01" "1559-01-25"
 "1520-11-01" "1523-01-20"
 "Sweden")
(ruler "Kristian II" "1481-07-01" "1559-01-25"
 "1513-07-22" "1523-01-20"
 "Norway")
(ruler "Kristian II" "1481-07-01" "1559-01-25"
    "1513-07-22" "1523-01-20"
    "Denmark")
   (ruler "Vasili III" "1505-10-27" "1533-12-03"
    "1523-06-06" "1533-12-03"
    "Russian Empire")
   (ruler "Alexander I" "1777-12-23" "1825-12-01"
    "1801-03-24" "1825-12-01"
    "Russian Empire")
   (ruler "Gustav IV Adolf" "1778-11-01" "1837-02-07"
    "1792-03-29" "1809-03-29"
    "Sweden")
   (ruler "Vladimir Putin" "1952-10-07" "9999-12-31"
    "1999-12-31" "9999-12-31"
    "Russian Federation")

   (event "Stockholm Bloodbath" "1520-11-07" "1520-11-09")
   (event "The Riksdag in Västerås" "1527-06-24" "1527-06-24")
   (event "Linköping Bloodbath" "1600-03-19" "1600-03-19")
   (event "Battle of Ratan and Sävar" "1809-08-17" "1809-08-23")
   (event "Finnish War" "1808-03-21" "1809-09-17")

   (happenedIn "Stockholm Bloodbath" "Stockholm" "Södermanland")
   (happenedIn "Stockholm Bloodbath" "Stockholm" "Uppland")
   (happenedIn "The Riksdag in Västerås" "Västerås" "Västmanland")
   (happenedIn "Linköping Bloodbath" "Linköping" "Östergötland")
   (happenedIn "Battle of Ratan and Sävar" "Sävar" "Västerbotten")
   (happenedIn "Battle of Ratan and Sävar" "Ratan" "Västerbotten")

   (involvedIn "The Riksdag in Västerås" "Gustav Vasa"
    "1496-05-12" "1560-09-23")
   (involvedIn "Stockholm Bloodbath" "Kristian II"
    "1481-07-01" "1559-01-25")
   (involvedIn "Finnish War" "Alexander I"
    "1777-12-23" "1825-12-01")
   (involvedIn "Finnish War" "Gustav IV Adolf"
    "1778-11-01" "1837-02-07")))

;; ****************** PHRASAL LEXICON *****************

(set-pl
 :english

 '((x person)
    (:head ("person") (:sing))
    (:head ("persons") (:pl))
    (:head ("key figure") (:nlu))
    (:head ("key figures") (:nlu)))

 '((x person (= x name $c1))
    (:comp ("named " $c1))
    (:comp ("called " $c1))
    (:head ($c1) () (:proper-name))
    (:answer ($c1)))

 '((x person (= x nationality $c1))
    (:comp ("that was " $c1)))

 '((x person (= x occupation $c1))
    (:comp ("that worked as an " $c1))
    (:comp ("that worked as a " $c1)(:nlu))
    (:comp ("that was a " $c1)(:nlu))
    (:comp ("that was an " $c1)(:nlu))
    (:comp ("that were a " $c1)(:nlu))
    (:comp ("that were an " $c1)(:nlu)))


 '((x person (+ (y1 y2) (involvedIn y1) (event y2)
  (= y1 person x name)
  (= y1 birthdate x birthdate)
  (= y1 deathdate x deathdate)
  (= y1 eventName y2 name)
  (*y2)))
    (:comp ("involved in " (:ref *y2)))
    (:comp ("involved in the " (:ref *y2))(:nlu))
    (:comp ("that was involved in " (:ref *y2))))

 '((x person (+ (y1 y2) (involvedIn y1) (event y2)
  (= y1 person x name)
  (= y1 birthdate x birthdate)
  (= y1 deathdate x deathdate)
  (= y1 eventName y2 name)
  (= y2 name $1)))
    (:comp ("involved in " $1))
    (:comp ("involved in the " $1)(:nlu))
    (:comp ("that was involved in " $1)))

 '((x person (> x birthdate $c1))
    (:comp ("born after " $c1))
    (:comp ("who was born after " $c1)))

 '((x person (< x birthdate $c1))
    (:comp ("born before " $c1))
    (:comp ("who was born before " $c1)))

 '((x person (> x deathdate $c1))
    (:comp ("died after " $c1))
    (:comp ("who died after " $c1)))
```

```
'((x person (< x deathdate $c1))
  (:comp ("died before " $c1))
  (:comp ("who died before " $c1)))

'((x person (+ (y) (event y)
 (<= x birthdate y startdate)
 (>= x deathdate y enddate)
 (*y)))
    (:comp ("who lived during " (:ref *y)))
    (:comp ("who was alive during " (:ref *y)))
    (:comp ("alive during " (:ref *y))))

'((x person (+ (y) (event y)
 (<= x birthdate y startdate)
 (>= x deathdate y enddate)
 (= y name $c1)))
    (:comp ("who lived during " $c1))
    (:comp ("who was alive during " $c1))
    (:comp ("alive during " $c1))
    (:comp ("who was alive when " $c1 " occured"))
    (:comp ("who lived when " $c1 " occured")))

'((x person (+ (y) (person y)
 (>= x birthdate y birthdate)
 (> y deathdate x birthdate)))
   (:head ("contemporary") (:nlu))
   (:head ("contemporaries") (:nlu))
   (:head ("who lived at the same time as") (:nlu))
   (:head ("earlier contemporary") (:sing))
   (:head ("earlier contemporaries") (:pl))
   (:head ("younger contemporary") (:nlu))
   (:head ("younger contemporaries") (:nlu)))

'((x person (+ (y) (person y)
 (>= x birthdate y birthdate)
 (> y deathdate x birthdate)
 (= y name $c1)))
   (:comp ("of " $c1)))

'((x person (+ (y) (person y)
 (<= x birthdate y birthdate)
 (> x deathdate y birthdate)))
   (:head ("contemporary") (:nlu))
   (:head ("contemporaries") (:nlu))
   (:head ("who lived at the same time as") (:nlu))
   (:head ("later contemporary") (:sing))
   (:head ("later contemporaries") (:pl))
   (:head ("older contemporary") (:nlu))
   (:head ("older contemporaries") (:nlu)))
```

```
'((x person (+ (y) (person y)
 (<= x birthdate y birthdate)
 (> x deathdate y birthdate)
 (= y name $c1)))
   (:comp ("of " $c1)))

'(((x name) person)
   (:head ("info"))
   (:head ("information")))

'(((x name) person
 (= x name $c1)
 (= x birthdate $c2)
 (= x deathdate $c3)
 (= x nationality $c4)
 (= x occupation $c5))
   (:comp ("on " $c1)(:nlu))
   (:answer ($c1 " (" $c2 " - " $c3 "), " $c4 " " $c5)))

'((x ruler)
   (:head ("swayer") (:nlu))
   (:head ("swayers") (:nlu))
   (:head ("kings") (:nlu))
   (:head ("king") (:nlu))
   (:head ("queens") (:nlu))
   (:head ("queen") (:nlu))
   (:head ("presidents") (:nlu))
   (:head ("president") (:nlu))
   (:head ("emperors") (:nlu))
   (:head ("emperor") (:nlu))
   (:head ("regents") (:nlu))
   (:head ("regent") (:nlu))
   (:head ("monarch") (:nlu))
   (:head ("monarchs") (:nlu))
   (:head ("ruler") (:sing))
   (:head ("rulers") (:pl)))

'((x ruler (= x name $c1))
   (:comp ("named " $c1))
   (:comp ("called " $c1))
   (:mod ("who was "))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))

'((x ruler (= x power $c1))
   (:comp ("who ruled " $c1))
   (:comp ("who reigned " $c1))
   (:comp ("who controlled " $c1))
   (:comp ("of " $c1))
   (:head ($c1) () (:proper-name)))

'((x ruler (+ (y1 y2) (involvedIn y1) (event y2)
 (= y1 eventName y2 name)
 (= y1 person x name)
 (= y1 birthdate x birthdate)
 (= y1 deathdate x deathdate)
 (*y2)))
   (:comp ("involved in " (:ref *y2)))
   (:comp ("that was involved in " (:ref *y2))))

'((x ruler (> x birthdate $c1))
   (:comp ("born after " $c1))
   (:comp ("who was born after " $c1)))

'((x ruler (< x birthdate $c1))
   (:comp ("born before " $c1))
   (:comp ("who was born before " $c1)))

'((x ruler (> x deathdate $c1))
   (:comp ("died after " $c1))
   (:comp ("who died after " $c1)))

'((x ruler (< x deathdate $c1))
   (:comp ("died before " $c1))
   (:comp ("who died before " $c1)))

'((x ruler (> x dethrowned $c1) (< x enthrowned $c2)
 (<> x dethrowned x birthdate))
   (:comp ("who ruled between " $c1 " and " $c2))
   (:comp ("that ruled between " $c1 " and " $c2))
   (:comp ("who reigned between " $c1 " and " $c2))
   (:comp ("that reigned between " $c1 " and " $c2)))

'((x ruler (+ (y) (event y)
 (<= x enthrowned y startdate)
 (>= x dethrowned y enddate)
 (*y)))
   (:comp ("during " (:ref *y)))
   (:comp ("who ruled during " (:ref *y)))
   (:comp ("who reigned during " (:ref *y))))

'((x ruler (+ (y) (event y)
 (<= x enthrowned y startdate)
 (>= x dethrowned y enddate)
 (= y name $c1)))
   (:comp ("during " $c1))
   (:comp ("who ruled when " $c1 " occured"))
   (:comp ("who reigned when " $c1 " occured")))

'((x ruler (> x dethrowned $c1) (< x enthrowned $c1)
 (<> x birthdate x deathdate))
   (:comp ("who ruled on " $c1)(:nlu))
   (:comp ("that ruled on " $c1)(:nlu))
   (:comp ("that reigned on " $c1)(:nlu))
   (:comp ("that reigned in " $c1))
   (:comp ("who ruled in " $c1)(:nlu))
   (:comp ("that ruled in " $c1)(:nlu))
   (:head ($c1) () (:proper-name)))

'((x ruler (= x power $c1)
 (< x enthrowned $c2)
 (> x dethrowned $c2)
 (<> x name x power))
   (:comp ("who ruled " $c1 " on " $c2)(:nlu))
   (:comp ("who controlled " $c1 " on " $c2)(:nlu))
   (:comp ("of " $c1 " on " $c2)(:nlu))
   (:comp ("who ruled " $c1 " in " $c2)(:nlu))
   (:comp ("who controlled " $c1 " in " $c2)(:nlu))
   (:comp ("of " $c1 " in " $c2)(:nlu))
   (:comp ("who reigned " $c1 " in " $c2))
   (:head ($c1) () (:proper-name)))

'(((x birthdate deathdate) person)
   (:head ("birthdate") (:nlu))
   (:head ("birthdates") (:nlu))
   (:head ("birthyear") (:nlu))
   (:head ("birth date") (:nlu))
   (:head ("birth dates") (:nlu))
   (:head ("birth year") (:nlu))
   (:head ("date of birth") (:nlu))
   (:head ("deathdate") (:nlu))
   (:head ("deathyear") (:nlu))
   (:head ("death date") (:nlu))
   (:head ("death year") (:nlu))
   (:head ("date of death") (:nlu))
   (:head ("lifespan") (:nlu))
   (:head ("lifespans") (:nlu))
   (:head ("when") (:nlu))
   (:head ("life span") (:sing))
   (:head ("life spans") (:pl)))

'(((x birthdate deathdate) person
 (= x name $c1)
 (= x birthdate $c2)
 (= x deathdate $c3))
   (:head ($c1) () (:proper-name))
   (:comp ("person " $c1))
   (:comp ("person called " $c1))
   (:comp ("person named " $c1))
   (:answer ($c1 " (" $c2 " - " $c3 ")")))

'(((x birthdate deathdate) person (= x name $c1))
   (:comp ("of " $c1))
   (:comp ("for " $c1))
   (:comp ("did " $c1 " die") (:nlu))
   (:comp ("was " $c1 " born") (:nlu)))

'(((x birthdate deathdate) person (*x))
   (:comp ("of " (:ref *x)))
   (:comp ("for " (:ref *x))(:nlu)))

'(((x birthdate deathdate) person
 (+ (y1) (ruler y1)
 (= x name y1 name)
 (= x birthdate y1 birthdate)
 (= x deathdate y1 deathdate) (*y1)))
   (:comp ("of " (:ref *y1)))
   (:comp ("for " (:ref *y1))(:nlu)))

'((x power)
   (:head ("powers") (:nlu))
   (:head ("power") (:nlu))
   (:head ("states") (:nlu))
   (:head ("state") (:nlu))
   (:head ("nations") (:pl))
   (:head ("nation") (:sing))
   (:head ("national states") (:pl))
   (:head ("national state") (:sing)))

'((x power (= x name $c1))
   (:comp ("named " $c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))

'((x power
 (+ (y1) (ruler y1) (= x name y1 power) (*y1)))
   (:comp ("that had " (:ref *y1))))

'((x country)
   (:head ("countries") (:pl))
   (:head ("country") (:sing))
   (:head ("lands") (:nlu))
   (:head ("land") (:nlu))
   (:head ("countrys") (:nlu)))
```

```
'((x country (= x name $c1))
   (:comp ($c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))

'((x country
 (+ (y1 y2) (containsRegion y1) (region y2)
 (= y1 country x name)
 (= y1 region y2 name)
 (= y2 name $c1)
 (<= y1 vst $c2)
 (>= y1 vet $c2)
 (*y2)))
   (:comp ("that controlled " $c1 " in " $c2))
   (:comp ("that had control over " $c1 " in " $c2))
   (:comp ("that contained " $c1 " in " $c2)(:nlu)))

'((x country
 (+ (y1 y2) (containsRegion y1) (region y2)
 (= y1 country x name)
 (= y1 region y2 name)
 (= y2 name $c1)
 (*y2)))
   (:comp ("that have had control over " $c1))
   (:comp ("that controlled " $c1)(:nlu))
   (:comp ("that contained " $c1)(:nlu)))

'((x country
 (+ (y1 y2) (containsRegion y1) (region y2)
 (= y1 country x name)
 (= y1 region y2 name)
 (= y1 VET "9999-12-31")
 (*y2)))
   (:comp ("containing" (:ref *y2))(:nlu))
   (:comp ("controlling" (:ref *y2))(:nlu))
   (:comp ("that contains" (:ref *y2))(:nlu))
   (:comp ("that controls" (:ref *y2))))

'((x region)
   (:head ("area") (:nlu))
   (:head ("areas") (:nlu))
   (:head ("region") (:sing))
   (:head ("regions") (:pl)))

'((x region (= x name $c1))
   (:comp ($c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))
```

```
'((x region
(+ (y1 y2) (containsLocation y1) (location y2)
(= y1 region x name)
(= y1 location y2 name)
(*y2)))
   (:comp ("of " (:ref *y2))(:nlu))
   (:comp ("containing " (:ref *y2)))
   (:comp ("that contains " (:ref *y2)))
   (:comp ("that controls " (:ref *y2))(:nlu)))

'((x region
 (+ (y1 y2) (containsRegion y1) (country y2)
 (= y1 country y2 name)
 (= y1 region x name)
 (= y2 name $c1)
 (<= y1 vst $c2)
 (>= y1 vet $c2)
 (*y2)))
   (:comp ("of " $c1 " in " $c2)(:nlu))
   (:comp ("in " $c1 " in " $c2)(:nlu))
   (:comp ("that was in " $c1 " in " $c2)(:nlu))
   (:comp ("that was part of " $c1 " in " $c2))
   (:comp ("that " $c1 " controlled in " $c2)(:nlu))
   (:comp ("that " $c1 " contained in " $c2)(:nlu)))

'((x region
 (+ (y1 y2) (containsRegion y1) (country y2)
 (= y1 country y2 name)
 (= y1 region x name)
 (= y2 name $c1)
 (<> y1 vet y1 vst)
 (*y2)))
   (:comp ("that " $c1 " have had control over")(:nlu))
   (:comp ("that " $c1 " controlled")(:nlu))
   (:comp ("that " $c1 " contained")(:nlu))
   (:comp ("that has been in " $c1)(:nlu))
   (:comp ("that has been part of " $c1)))

'((x region
 (+ (y1 y2) (containsRegion y1) (country y2)
 (= y1 country y2 name)
 (= y1 region x name)
 (= y1 VET "9999-12-31")
 (*y2)))
   (:comp ("of" (:ref *y2))(:nlu))
   (:comp ("in" (:ref *y2))(:nlu))
   (:comp ("that is in" (:ref *y2))(:nlu))
   (:comp ("that is part of" (:ref *y2))))
```

```
'((x location)
   (:head ("battlefield") (:nlu))
   (:head ("battlefields") (:nlu))
   (:head ("settlement") (:nlu))
   (:head ("settlements") (:nlu))
   (:head ("location") (:sing))
   (:head ("locations") (:pl)))

'((x location (= x name $c1))
   (:comp ($c1))
   (:comp ("named " $c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))

'((x location
 (+ (y1 y2) (happenedIn y1) (event y2)
 (= y1 event y2 name)
 (= y1 location x name)
 (*y2)))
   (:comp ("of" (:ref *y2)))
   (:comp ("of the event" (:ref *y2))(:nlu))
   (:comp ("for" (:ref *y2))(:nlu)))

'((x location
 (+ (y0 y2) (containsLocation y1) (region y2)
 (= y1 region y2 name)
 (= y1 location x name)
 (*y2)))
   (:comp ("in" (:ref *y2)))
   (:comp ("in the region" (:ref *y2))(:nlu))
   (:comp ("within" (:ref *y2))(:nlu)))

'((x event)
   (:head ("event") (:sing))
   (:head ("events") (:pl)))

'((x event (= x name $c1))
   (:comp ("named " $c1))
   (:comp ("called " $c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1)))

'((x event
 (+ (y1 y2) (involvedIn y1) (ruler y2)
 (= y1 person y2 name)
 (= y1 birthdate y2 birthdate)
 (= y1 deathdate y2 deathdate)
 (= y1 eventName x name)
 (*y2)))
```

```
   (:comp ("which involved " (:ref *y2))))

'((x event
 (+ (y1 y2) (involvedIn y1) (ruler y2)
 (= y1 person y2 name)
 (= y1 birthdate y2 birthdate)
 (= y1 deathdate y2 deathdate)
 (= y1 eventName x name)
 (= y2 name $c1)))
   (:comp ($c1 " was involved in"))
   (:comp ("in which " $c1 " was involved")))

'((x event
 (+ (y1 y2) (happenedIn y1) (location y2)
 (= y1 event x name)
 (= y1 location y2 name)
 (*y2)))
   (:comp ("that occured in " (:ref *y2)))
   (:comp ("that happened in " (:ref *y2))))

'((x event
 (+ (y1 y2) (happenedIn y1) (region y2)
 (= y1 event x name)
 (= y1 region y2 name)
 (*y2)))
   (:comp ("that occured in " (:ref *y2)))
   (:comp ("that happened in " (:ref *y2))))

'((x event (+ (y) (ruler y)
 (>= x startdate y enthrowned)
 (<= x enddate y dethrowned)
 (*y)))
   (:comp ("that happened during the rule of " (:ref *y)))
   (:comp ("that happened during the reign of " (:ref *y)))
   (:comp ("that occured during the rule of " (:ref *y)))
   (:comp ("that occured during the reign of " (:ref *y))))

'((x event (+ (y) (ruler y)
 (>= x startdate y enthrowned)
 (<= x enddate y dethrowned)
 (= y name $c1)))
   (:comp ("that happened during " $c1 " reign")(:nlu))
   (:comp ("that happened during " $c1 " rule")(:nlu))
   (:comp ("that occured during " $c1 " reign")(:nlu))
   (:comp ("that occured during " $c1 " rule")(:nlu))
   (:comp ("that occured during " $c1 "s rule")))

'((x event (+ (y) (person y)
 (<= y birthdate x startdate)
 (>= y deathdate x enddate)
 (*y)))
   (:comp ("that occured during the lifetime of " (:ref *y))))

'((x event (+ (y) (person y)
 (<= y birthdate x startdate)
 (>= y deathdate x enddate)
 (= y name $c1)))
   (:comp ("that occured when " $c1 " was alive"))
   (:comp ("that happened when " $c1 " was alive")))

'(((x startdate enddate) event)
   (:head ("startdate") (:nlu))
   (:head ("start date") (:nlu))
   (:head ("startdates") (:nlu))
   (:head ("start dates") (:nlu))
   (:head ("enddate") (:nlu))
   (:head ("end date") (:nlu))
   (:head ("enddates") (:nlu))
   (:head ("end dates") (:nlu))
   (:head ("duration") (:nlu))
   (:head ("durations") (:nlu))
   (:head ("when ") (:nlu))
   (:head ("interval") (:nlu))
   (:head ("intervals") (:nlu))
   (:head ("date") (:nlu))
   (:head ("dates") (:nlu))
   (:head ("date of event") (:nlu))
   (:head ("dates of events") (:nlu))
   (:head ("date of the event") (:sing))
   (:head ("dates of the events") (:pl)))

'(((x startdate enddate) event
 (= x name $c1)
 (= x startdate $c2)
 (= x enddate $c3))
   (:comp ("event" $c1))
   (:comp ("event called" $c1))
   (:comp ("event named" $c1))
   (:head ($c1) () (:proper-name))
   (:answer ($c1 " (" $c2 " - " $c3 ")")))

'(((x startdate enddate) event
 (= x name $c1))
   (:comp (" " $c1))
   (:comp (" for " $c1)(:nlu))
   (:comp (" of " $c1)(:nlu))
   (:comp (" did " $c1 " occur")(:nlu))
   (:comp (" did " $c1 " happen")(:nlu)))
```

```
'(((x startdate enddate) event (*x))
   (:comp (" for " (:ref *x)))
   (:comp (" of " (:ref *x))))

'((x capital)
   (:head ("capital"))
   (:head ("capitals"))
   (:answer ("The database does not contain information on
             which cities that are capials of nations.")))

'((x invention)
 (:head ("invention"))
 (:head ("inventions"))
 (:head ("discovery"))
 (:head ("discoveries"))
 (:answer ("The database does not contain information on
           inventions or discoveries.")))

'((x predecessor)
   (:head ("father"))
   (:head ("mother"))
   (:head ("son"))
   (:head ("sons"))
   (:head ("daughter"))
   (:head ("daughters"))
   (:head ("wife"))
   (:head ("wifes"))
   (:head ("husband"))
   (:head ("husbands"))
   (:head ("cousins"))
   (:head ("cousin"))
   (:head ("family"))
   (:head ("predecessor"))
   (:head ("successor"))
   (:head ("predecessors"))
   (:head ("successors"))
   (:answer ("The database does not contain family relationships or
             ordered relationships between rulers.")))

'((x have (+ (y1) (person y1) (= y1 name $c1)))
   (:comp ($c1 " have "))
   (:comp ($c1 " own "))
   (:comp ($c1 " use "))
   (:answer ("This database does not contain any informations on items")))

'((x have)
   (:head ("Did"))
   (:head ("Does")))
 '((x happenedInTheYear)
  (:head ("happened in the year"))
  (:head ("what happened in the year"))
  (:head ("what happened on the date"))
  (:head ("happened on the date"))
  (:answer ("The database can not say what happened on a specific date."))))

;; ******************  Synonyms *******************

(set-syns
 '((location name)
    ("Västerås" ("Vasteras"))
    ("Linköping" ("Linkoping")))

 '((event name)
    ("Thirty Years War" ("Trettioåriga kriget" "Thirty Years' War"))
    ("Stockholm Bloodbath" ("Stockholms bloodbath"))
    ("Linköping Bloodbath" ("Linköpings bloodbath"))
    ("Battle of Ratan and Sävar" ("Battle of Ratan" "Battle of Sävar"
                                  "Battle of Sävar and Ratan")))

 '((region name)
    ("Swedish Pomerania" ("Vor-Pommern" "Hinter-Pommern" "Vorpommern"
                         "HinterPommern" "Swedish Pommern"))
    ("Swedish Livonia" ("Livonia" "Swedish Livland" "Estonia"))
    ("Vyborg" ("Viborg")))

 '((power name)
    ("Russian Empire" ("Russia" "the Russian Empire"))
    ("Russian Federation" ("Russia" "the Russian Federation")))

 '((ruler name)
    ("Carl XVI Gustaf" ("Carl Gustaf Folke Hubertus"
                       "Carl Gustaf Folke Hubertus Bernadotte"))
    ("Gustaf VI Adolf" ("Oscar Fredrik Wilhelm Olaf Gustaf Adolf Bernadotte"
                       "Oscar Fredrik Wilhelm Olaf Gustaf Adolf"))
    ("Gustav V" ("Oscar Gustaf Adolf" "V Gustaf"))
    ("Oscar II" ("Oskar II" "Oscar Fredric" "Oscar Fredric Bernadotte"
                "Oscar Fredrik" "Oscar Fredrik Bernadotte"))
    ("Karl XV" ("Karl Ludvig Eugen" "Karl Ludvig Eugén"
               "Carl Ludvig Eugen" "Carl Ludvig Eugén"
               "Carl XV" "Carl IV"
               "Karl IV" "Charles XV"))
    ("Oscar I" ("Oskar I" "Josef Frans Oskar"
               "Joseph François Oscar Bernadotte"
               "Oscar Bernadotte"))
    ("Karl XIV Johan" ("Jean Baptiste Bernadotte"
                      "Jean Baptiste Jules Bernadotte"
```

```
                     "Jean Deu Pouey"
                     "Carl XIV Johan"
                     "Charles XIV" "Karl III" "Carl III"))
   ("Karl XIII" ("Karl II" "Carl II" "Carl XIII" "Charles XIII"))
   ("Jakob Johan Anckarström" ("Anckarström"))
   ("Fredrik I" ("Frederick I" "Friedrich I von Hessen-Kassel"
                "Fredrik von Hessen"))
   ("Ulrika Eleonora" ("Ulrika Eleonora d.y."))
   ("Peter I" ("Peter the Great" "Peter den Store" "Pyotr Alexeevich"))
   ("Karl X Gustav" ("Charles X"))
   ("Karl XI" ("Charles XI"))
   ("Karl XI" ("Charles XII" "Carolus Rex" "Alexander of the North"
              "Demirbas sarl" "Charles the Habitue"))
   ("Kristina" ("Christina" "Maria Christina Alexandra" "Count Dohna"))
   ("Gustav II Adolf" ("Gustav II Adolph" "Gustavus Adolphus"
                      "Gustav the Great" "Gustav den Store"))
   ("Karl IX" ("Charles IX" "Hertig Karl" "Duke Charles"))
   ("Gustav Vasa" ("Gustav I Eriksson" "Gustaf Vasa" "Gustav I"
                  "Gustav I Vasa"))
   ("Erik XIV" ("Erik Eriksson Vasa" "Erik XIV Vasa" "Eric XIV"
               "Eric XIV Vasa"))
   ("Sigismund" ("Sigismund III" "Zygmunt III" "Zygmunt III Waza"
                "Sigismund III Vasa" "Sigismund I"))
   ("Kristian II" ("Kristian tyrann" "Christian the tyrant"
                  "Christian II"))
   ("Vasili III" ("Vasilii III" "Vasily III" "Vasili III Ivanovich"
                 "Vasilii III Ivanovich" "Basil III"))
   ("Vladimir Putin" ("Vladimir Vladimirovich Putin"))
   ("Alexander I" ("Aleksander Pavlovich Romanov" "Alexander the blessed")))

'((person name)
   ("Carl XVI Gustaf" ("Carl Gustaf Folke Hubertus"
                      "Carl Gustaf Folke Hubertus Bernadotte"))
   ("Gustaf VI Adolf" ("Oscar Fredrik Wilhelm Olaf Gustaf Adolf Bernadotte"
                      "Oscar Fredrik Wilhelm Olaf Gustaf Adolf"))
   ("Gustav V" ("Oscar Gustaf Adolf" "V Gustaf"))
   ("Oscar II" ("Oskar II" "Oscar Fredric" "Oscar Fredric Bernadotte"
               "Oscar Fredrik" "Oscar Fredrik Bernadotte"))
   ("Karl XV" ("Karl Ludvig Eugen" "Karl Ludvig Eugén"
              "Carl Ludvig Eugen" "Carl Ludvig Eugén"
              "Carl XV" "Carl IV"
              "Karl IV" "Charles XV"))
   ("Oscar I" ("Oskar I" "Josef Frans Oskar"
              "Joseph François Oscar Bernadotte"
              "Oscar Bernadotte"))
   ("Jean Baptiste Bernadotte" ("Jean Baptiste Jules Bernadotte"
                               "Jean Deu Pouey"))
   ("Karl XIV Johan" ("Carl XIV Johan" "Charles XIV" "Karl III" "Carl III"))
   ("Karl XIII" ("Karl II" "Carl II" "Carl XIII" "Charles XIII"))
```

```
    ("Jakob Johan Anckarström" ("Anckarström"))
    ("Fredrik I" ("Frederick I" "Friedrich I von Hessen-Kassel"
                  "Fredrik von Hessen"))
    ("Ulrika Eleonora" ("Ulrika Eleonora d.y."))
    ("Peter I" ("Peter the Great" "Peter den Store" "Pyotr Alexeevich"))
    ("Karl X Gustav" ("Charles X"))
    ("Karl XI" ("Charles XI"))
    ("Karl XI" ("Charles XII" "Carolus Rex" "Alexander of the North"
                "Demirbas sarl" "Charles the Habitue"))
    ("Kristina" ("Christina" "Maria Christina Alexandra" "Count Dohna"))
    ("Gustav II Adolf" ("Gustav II Adolph" "Gustavus Adolphus"
                        "Gustav the Great" "Gustav den Store"))
    ("Karl IX" ("Charles IX" "Hertig Karl" "Duke Charles"))
    ("Gustav Vasa" ("Gustav I Eriksson" "Gustaf Vasa" "Gustav I"
                    "Gustav I Vasa"))
    ("Erik XIV" ("Erik Eriksson Vasa" "Erik XIV Vasa" "Eric XIV"
                 "Eric XIV Vasa"))
    ("Sigismund" ("Sigismund III" "Zygmunt III" "Zygmunt III Waza"
                  "Sigismund III Vasa" "Sigismund I"))
    ("Kristian II" ("Kristian tyrann" "Christian the tyrant"
                    "Christian II"))
    ("Vasili III" ("Vasilii III" "Vasily III" "Vasili III Ivanovich"
                   "Vasilii III Ivanovich" "Basil III"))
    ("Vladimir Putin" ("Vladimir Vladimirovich Putin"))
    ("Alexander I" ("Aleksander Pavlovich Romanov" "Alexander the blessed"))))
```

Appendix E

Tasks from the Comparative Study

E.1 Simpler tasks

1. Using interface A produce a list of all Swedish regents.
2. Using interface A find out who ruled Sweden on January 12, 1700.
3. Using interface A produce a list of all rulers that reigned before January 12, 1700.
4. Using interface A find out on which dates the event referred to as the battle of Sävar began and ended
5. Using interface A find out the occupation and nationality of the person named Alfred Nobel
6. Using interface A produce a list of all the major events that Gustav Vasa was involved in
7. Using interface B produce a list of all major events in Swedish history.
8. Using interface B produce a list of all persons involved in the assassination of Gustav III
9. Using interface B produce the life span of the person named Jean Baptiste Bernadotte
10. Using interface B produce a list of all persons born before November 6, 1632
11. Using interface B produce a list of all major event that occurred in the city of Stockholm
12. Using interface B produce a list of all powers that at some time have had control over the region called Swedish Pomerania

E.2 Elaborate tasks

1. Using interface A produce a list that only contains persons which fulfills all of the following criteria.
   - They were/are the regent of Sweden.
   - They were at some time involved in a major event that occurred in the city of Stockholm
2. Using interface A produce a list of all persons involved in an major event at the time of their death
3. Using interface A find out the location of the major event that occurred on March 29, 1638
4. Using interface A produce a list that only contains persons which fulfills all of the following criteria.
   - They were/are the regent of Sweden.
   - They were alive at the same time as the thirty years' war occurred
5. Using interface A produce a list with four persons that were alive at the same time.
6. Using interface A produce a list of all rulers that has been king over more than on country
7. Using interface A produce a list with regions that fulfill the following conditions
   - The region has at some time been under Swedish control
   - The region has at some time been under Danish control
   - The region was controlled by Sweden before it was controlled by Denmark.
8. Using interface B produce a list with regions that fulfill the following conditions
   - The region has at some time been under Swedish control
   - The region has at some time been under Danish control
   - The region was controlled by Denmark before it was controlled by Sweden.
9. Using interface B produce a list of *all* persons that lived at the same time as Gustav Vasa
10. Using interface B produce a list that only contains persons which fulfills all of the following criteria.
    - They were/are the regent of Sweden.
    - They were alive at the same time as the thirty years' war occurred
11. Using interface B produce a list of locations which fulfills the following criteria
    - A ruler of Sweden was involved in an event that occurred on this location
    - The location was at the time of the event not under Swedish control

12. Using interface B produce a list of all powers that have had control over the region called Swedish Livonia at least twice
13. Using interface B produce a list of all events that involved at least two rulers of Sweden