2010-06-29

Automatic Extraction From and Reasoning About Genealogical Records: A Prototype

Charla Jean Woodbury
Brigham Young University - Provo

AUTOMATIC EXTRACTION FROM AND REASONING ABOUT GENEALOGICAL RECORDS: A PROTOTYPE

Charla Woodbury

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

David Embley, Chair
Deryle Lonsdale
Daniel Zappala

Department of Computer Science

Brigham Young University

August 2010

ABSTRACT

AUTOMATIC EXTRACTION FROM AND REASONING ABOUT GENEALOGICAL RECORDS: A PROTOTYPE

Charla Woodbury

Department of Computer Science

Master of Science

Family history research on the web is increasing in popularity, and many competing genealogical websites host large amounts of data-rich, unstructured, primary genealogical records. It is labor-intensive, however, even after making these records machine-readable, for humans to make these records easily searchable. What we need are computer tools that can automatically produce indices and databases from these genealogical records and can automatically identify individuals and events, determine relationships, and put families together. We propose here a possible solution—specialized ontologies, built specifically for extracting information from primary genealogical records, with expert logic and rules to infer genealogical facts and assemble relationship links between persons with respect to the genealogical events in their lives.

The deliverables of this solution are extraction ontologies that can extract from parish or town records, annotated versions of original documents, data files of individuals and events, and rules to infer family relationships from stored data. The solution also provides for the ability to query over the rules and data files and to obtain query-result justification linking back to primary genealogical records. An evaluation of the prototype solution shows that the extraction has excellent recall and precision results and that inferred facts are correct.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. David W. Embley for his unending help and guidance; Dr. Steven W. Liddle for his technical help and advice; and my family and friends for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

The many family history websites on the Internet compete to provide large systems of primary records that are easily and quickly searched by non-expert users. For example, most of the major family history websites have indexed United States census data from 1850 to 1930. Users look up names and places by searching indices that ultimately link to digital images of original census pages.

So far, family history sites such as Ancestry.com [www.ancestry.com], Family Tree Maker [www.familytreemaker.com], and Heritage Quest [www.heritagequest.com] have used large traditional relational databases. Manual data entry is generally used to populate those databases and to link those databases to digital images of original primary documents. The human effort to enter and index information from handwritten census pages is staggering, especially considering that many use double entry of the data to remove input errors. The Church of Jesus Christ of Latter-day Saints, for example, has organized large numbers of people to manually index such projects as passenger lists and census information.

What the industry needs is a smarter and faster way of producing searchable primary genealogical records and a better way to identify individuals and family relationships. Dallin Quass, the keynote speaker at the 2003 Family History Technology Workshop [Qua03], stated that we need "faster image indexing." He also said, "People currently index images manually" by using "two independent indexers and adjudication" which involves tremendous human effort. He indicated that simplistic indexing of records and images is not enough. We need to link records: "Given a person in a pedigree and a large set of genealogical records, do any of the

records match?" This is a very large goal covering varied disciplines to automate indexing and linkage, but a new approach in extraction and inference offers a potential solution to this question.

The development of the Semantic Web [http://www.w3.org/2001/sw] has produced toolsets that aid a computer in its ability to "understand" the meaning of a word in a particular subject domain. Scientists like Maedche et al. [MNS02] and Embley [Emb04] have suggested using lexical knowledge, extraction rules, and modeling to add semantic understanding to computer programs. Tools like ontologies with regular expressions and lexicons can be used to organize and give meaning to large amounts of unstructured, primary genealogical records in or out of the Semantic Web. And best of all, this toolset can be used today before the full rollout of the Semantic Web.

The functionality of the Semantic Web toolset, however, needs to be expanded to add specialized domain expertise for genealogical records. Once this expertise is defined and corresponding ontologies are built, then machine-readable genealogical records will be automatically indexable and fully searchable. If successful, this means that every bit of genealogical information in the primary records can be used to qualify an individual For example, records often include an individual's occupation, place of residence, or witnesses present, which are helpful in differentiating individuals with the same name, but are rarely available in a simple name index. Furthermore, expert logic could be used to make the machine do more of the work, both for extraction and indexing, as well as for partially assembling families.

Imagine researchers being able to pull partial or even whole families pre-assembled out of a parish or town register.

The prototype described in this paper provides for automating the information extraction process over unstructured machine-readable genealogical records and for querying the extracted information by:

- designing appropriate primary record extraction ontologies for family history records that produce formatted RDF data in OWL files,[1]
- adding SWRL[2] rules and logic to the files that, when applied, properly label and link primary genealogical data,
- constructing SPARQL[3] queries from free-form user queries that show selected extracted data with the rules and logic applied, and
- testing and evaluating the prototype for accuracy.

[1] RDF and OWL are standard Semantic Web languages for data and metadata respectively.
[2] SWRL is a Semantic Web rule language.
[3] SPARQL is a Semantic Web query language.

2. Data Preparation

To show the capabilities of the prototype system, we chose genealogical data sources from different geographical locations in order to include geographical diversity and allow us to determine the impact of that diversity. The first set of data comes from Beverly vital records in the state of Massachusetts. The second set is from British church records of South Petherton in Somersetshire, England. The third set comes from Danish parish records of Magleby in the county of Praesto, Denmark.

Several languages are included in these selections. The Beverly, Massachusetts vital records are in English. The South Petherton church records are in Latin and English. The Magleby parish records are in Latin and Danish.

Different methods were used to put the original genealogical data into machine-readable format: (1) The Beverly, Massachusetts vital records had been published and were digitized using an optical character reader into PDF format and then converted into HTML documents using Pixillion software. (2) The South Petherton data had been transcribed in HTML format on the www.genuki.uk web site. (3) The Danish records for Magleby parish were transcribed from the microfilm by hand.

The final format used for extraction is HTML. The Ontology Extraction System (OntoES) tool requires a web format. The British web pages were downloaded in that format. The others were quickly reformatted to the final format using Microsoft Word.

Figure 1 is a simple example of an abridged set of the data downloaded from the GENUKI web site. It shows a list of selected marriages from South Petherton, England. Figure 2 is the same group of marriages, but this time the data has been

copied from the original Latin church record as it appeared in the original church register.

South Petherton Marriages

same day 1576 Nicholas Patch and Christian Denman
26 Jan 1605 Richard Patch and Joan Lavor
25-Sep 1613 John Elliott and Joan Woodbery
7-Aug 1615 Thomas Prime and Maria Parry
29-Jan 1616 William Woodbery and Elizabeth Patch
2-May 1620 William Hillerd and Fortu: Patch
17-Sep 1622 Nicholas Patch and Elizabeth Owsley
22-Jan 1627 Richard Patch and Mary White
15-Jan 1630 Andrew Elliott and Joan Patch
12-Feb 1639 Andrew Elliott and Joan Pitts

Figure 1. South Petherton marriages from GENUKI web page.

South Petherton Marriages

1576/1577 eodem die Nicholaus Patch Christinam Denman
26 Jan 1605 Richard Patch et Joanna Lavor
1613 Septembris 26 Johannes Elliott et Joanna Woodbery matrimonis cominguntur
1615 Augusti 7 Thoms Prime et Maria Patch matrimonio cominguntur
1616/1617 Januarij 29 Wilhelmus Woodbery et Elizabetha Patch matrimonio cominguntur
1620 Maij 2 : Wilhelmus Hillerd et Fortu: Patch
1622 Septembris 17 Nicholas Patch et Elizabetha Owsley matrimonio cominguntur
1627/1628 Januarij 22 : Richardus Patch et Maria White matrimonio cominguntur
1630/1631 Januarij 15 Andreas Elliott et Joanna Patch matrimonio cominguntur
1639/1640 Februarij 12 Andreas Elliott et Joanna Pittes matrimonio cominguntur

Figure 2. South Petherton marriages transcribed directly from the parish register.

There are several noticeable differences between the two records from South Petherton, England. Figure 1 is in English with names given according to present-day spelling standards. Figure 2 is in Latin as it appeared is the original parish register with the original spelling of the names as they appeared in the parish record. Double dates were added to Figure 2 to indicate when January, February, and March entries were listed at the end of the year, which today would have been listed at the beginning of the new year.

The process described in this paper returns information according to the data given to the OntoES system. It can be no better than the data provided.

3. Extraction Ontologies

3.1 Description and Definition of Terms for Building Ontologies

An ontology consists of descriptions of the data entities and how they are related. OntoES [ECJ+99], developed by the Data Extraction Group at Brigham Young University, allows ontology designers to create ontologies by using modeling techniques. The basic component of an ontology is an object set. The objects or values in an object set are described by a data frame that encapsulates knowledge about the appearance, behavior, and context of a collection of data elements [Emb80].

Relationships among objects are captured in relationship sets. Constraints, such as cardinality constraints, serve to constrain object and relationship sets. Lexicons list all possible values for a particular entity. Figure 3 gives an example. It contains part of a month-name lexicon. Multiple spellings and abbreviations must be anticipated in a lexicon and thus prepared before using the ontology.

MONTH LEXICON

10ber	decembr
7ber	decembre
8ber	decembri
9ber	feb
apr	febr
april	februari
aprilis	february
aug	jan
august	januarij
augusti	january
augustus	jul
avr	juli
avril	julius
avrilis	july
dec	jun
december	june

Figure 3. Sample partial lexicon listing words denoting months.

Once successfully built, these lexicons can be re-used in new projects in the same domain. We added lexicons to the ontology for given name, patronymic name (Danish only), surname, feast date, place name, occupation, and family relationship. Problems such as abbreviations, misspelled words, and multiple languages are handled in the lexicons.

3.2 Ontology Models

3.2.1 Ontology Conceptual Model

Figure 4 shows the conceptual-model component of the ontology. Object sets are in boxes linked by relationship sets. Object-set boxes are dashed if they are lexical (have instances that are string representations of values such as a person's name) and are solid if they are non-lexical (have instances that are object identifiers identifying real-world objects such as persons). As Figure 4 shows, marriage records associate

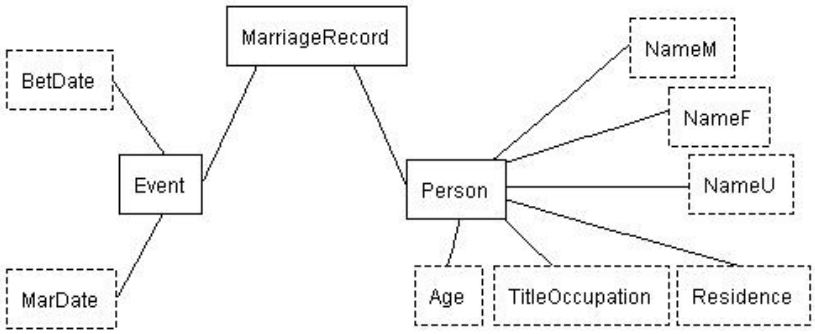


Figure 4. Marriage ontology.

with persons and events. Data of interest for a person includes names (and whether the name is a male name, a female name, or a name not associated with a gender),

age, occupations or titles, and residence. Data of interest for a marriage event includes betrothal dates and marriage dates.

3.2.2 Ontology Instance Recognizers

Instances for each of the lexical objects may be defined using regular expressions. Figure 5 shows the data-frame editor open for the marriage date, MarDate. For instances of each object set, users can declare value phrases that describe the lexical form of the instances and can declare keyword phrases that indicate the possible presence of an occurrence in free-form text. In Figure 5, the data-frame editor is open for Value Phrases. For example, one of the Value Expressions for MarDate is:

```
(0\d|1\d|2\d|30|31|\d)-{Month}\.?\s*(\d\d\d\d)
```

This expression accepts dates like “25-Sep 1613”, the third date in Figure 1. In general it accepts dates without day values or with day values up to 31 followed immediately by a hyphen and then a month from the month lexicon (partially shown in Figure 3) and then by potentially a period and finally by a four-digit year.

Users can add a “hint” (here, “mardate_simplehyphen”) to name the regular expression recognizer. As Figure 5 shows, the user has provided at least thirteen recognizers for MarDate values. Each tells how to handle a different date pattern, so that together they handle all date formats that are expected.

As Figure 5 shows, additional help for recognizing values can come from exception expressions, left-context expressions, and right-context expressions. Exception expressions describe instance data to be ignored. Context expressions

describe text expected immediately before or after the value phrase of an extraction target. A left-context expression describes a string expected to be at the left, but not included in the value to be extracted. A right-context expression describes a string expected to the right, but not included in the value to be extracted.

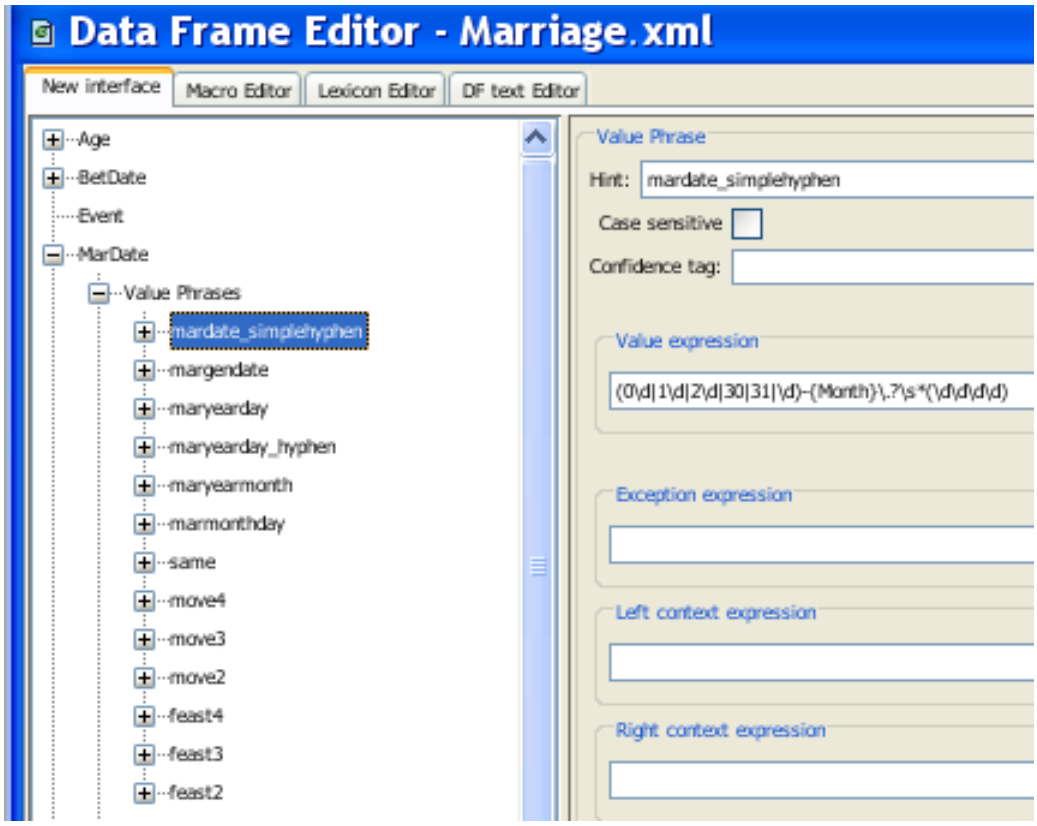


Figure 5. Dataframe example for the MarDate object set under Marriage ontology.

Keyword phrases describe context phrases that may be near a value that would help disambiguate the value from other values present in a record. For MarDate, the keyword recognizer is:

```
(\b(md\.?|marry|marriage|married|maried|wed||wedding)\b)
```

This helps classify a date as a marriage date if one of the keywords “md”, “marry”, “marriage”, “married”, “married”, “wed”, or “wedding” is found close by, and it thus helps to correctly disambiguate betrothal dates and marriage dates, which often occur in the same record.

3.2.3 Canonicalization of Datess

The form and meaning of dates varies so much that they must be regularized before computations can be applied. OntoES provides a way to link to methods written in Java. We created a Java method to regularize dates.

Although OWL and RDF have built-in data types and functions for handling dates, they are not rich enough to handle the complexity needed for dates in the historical genealogical domain. Thus, we instead decided to store dates as integers in Julian-date form *YYYYddd,* where *YYYY* is the four-digit year and *ddd* is the day of the year. In this form it is relatively easy to perform computations and handle historical irregularities.

Figure 6 shows how to declare functions in OntoES to convert recognized strings to internal values of some type and how to convert internal values to some standard string for value display. OntoES refers to the process as “canonicalization” because it standardizes instance values, converting the many ways of representing a value in writing to a single value of a built-in type and providing a single, uniform way of displaying the value when the system displays it. The canonicalization interface in OntoES lets us identify the Java method for changing the date to *YYYYddd* for MarDate, and the output formatting method for displaying dates chosen in this case to be *DD MMM YYYY*. So a date listed on the web page as “1620 2-May” will be stored

as “1620093” and displayed as “2 MAY 1620” which is a more-readable standard format. The value “1620093” is used internally for functions such as before or after some other date and the number of years between two dates, for example, to compute an estimated birth year.

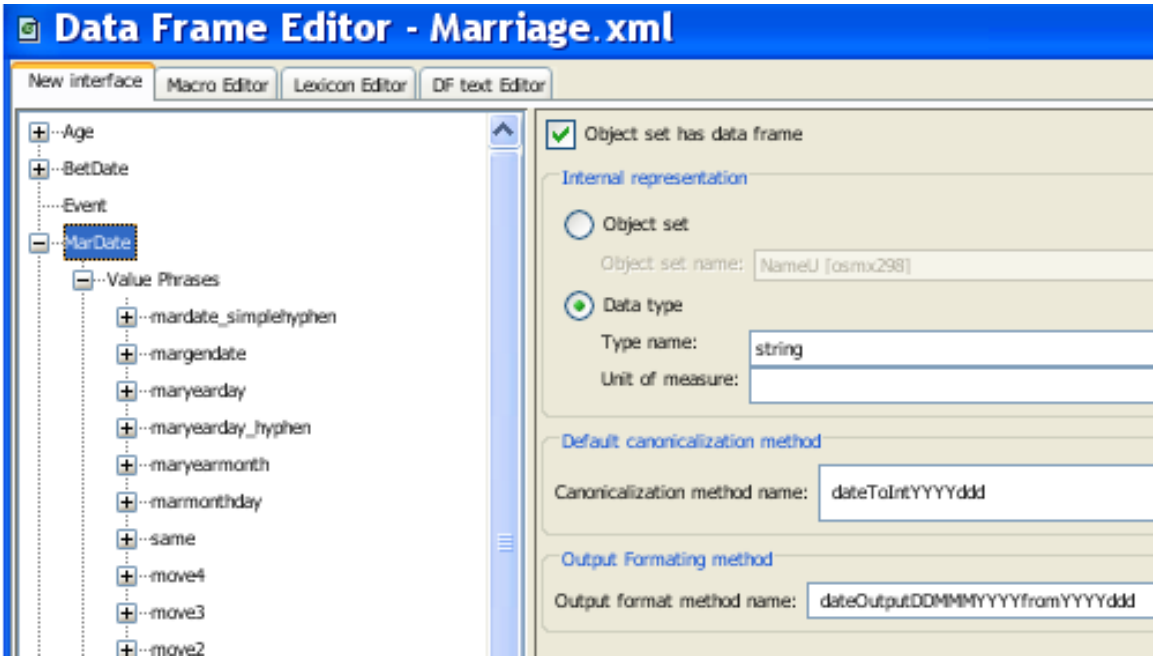


Figure 6. Data Frame Editor at object level.

3.2.4 Managing Feast Dates

Feast dates in genealogical records are particularly interesting and challenging to convert to standard dates. Nevertheless, the mechanisms to handle them in OntoES remain the same and feast dates can be handled in a way similar to regular date designations. There are two types of feast dates used in genealogical records: (1) fixed feast dates and (2) moveable feast dates.

Fixed feast dates occur on the same month and day every year. Examples of fixed feast dates are New Year’s Day (January 1st), Christmas (December 25th), All Saints

Day (November 1st). There is no major problem in extracting fixed feast dates and using a canonicalization function to determine the correct Julian date to be stored. Using a look up table along with knowledge about leap-year calculations, it is straight-forward to match the verbiage of a fixed date to its precise month and day. An English example is this christening:

```
1581 last day of Sep [blank] daughter of Robert Symes
```

Here the "last day of Sep" is handled as a fixed feast date, and our method converts it from 30 Sep 1581 to "1581273".

Moveable feast dates, on the other hand, are not in the same day every year. Easter and Thanksgiving are examples of moveable dates. Many of the parish dates are given as moveable dates or expressed in relationship to moveable dates. There are two types of moveable dates: those based on (1) a certain day of the week and (2) those based on Easter. Here are examples of two moveable feast dates:

```
1646 Dnica Septuagesima
1736 Dom: Quasimodog:
```

"Septuagesima" is Latin for a Sunday nine weeks before Easter. "Quasimodog:", an abbreviation for "Quasimodogeniti," is Latin for a Sunday one week after Easter. It is possible to compute these dates based on the year.

For those feast dates based on a certain day of the week, like Thanksgiving on the fourth Thursday in November or Mother's Day on the second Sunday in May, there are seven possible dates as the calendar rotates. They can be determined by a number

assigned to every year. So the algorithm is to look up the number for the year and then use that number assigned to each year to look up the correct month and day. For those feast dates based on Easter, there are 36 possible sequences of dates with another 36 sequences added for the variation of leap year. Again each year is given a number that indicates which of the 72 sequences to check. Using that number, it is possible to determine the year, identify a leap year, and retrieve the sequence number which then matches the feast date to the precise month and day. We used the "Calendar of Feast Days" compiled by Henry E. Christiansen as given in [Smi69] to build these 72 sequences.

3.3 Running OntoES

Figure 7 shows OntoES ready to process the Marriage ontology on the left with the data from the South Petherton marriage web page on the right. After the ontology

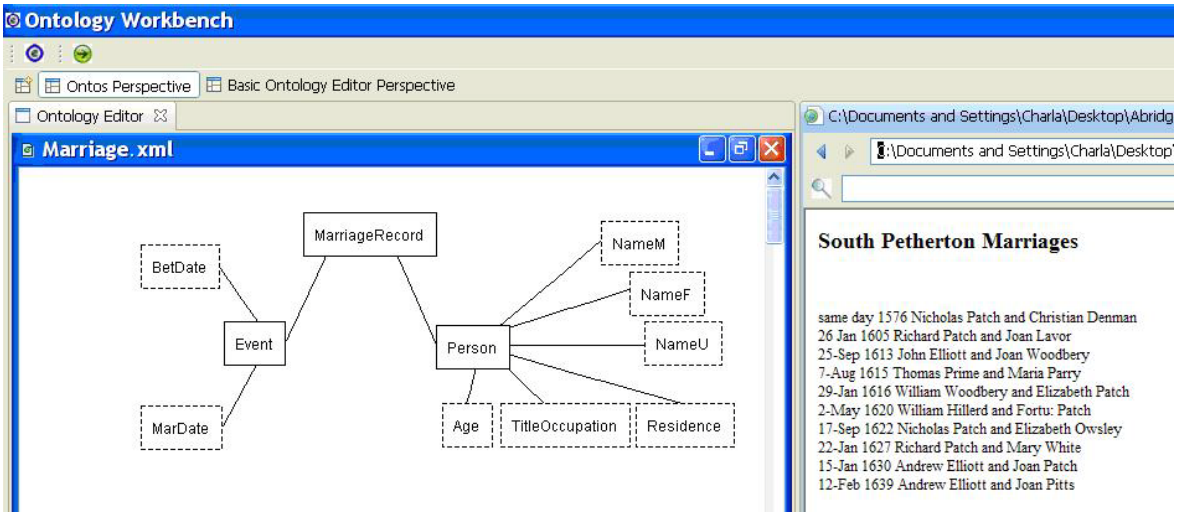


Figure 7. OntoES Workbench ready to run the extraction.

and the web page are selected, the extraction is begun by pressing the extraction

button which is near the upper left corner of the page.

3.4 Examples of Extraction Results

Figure 8 shows the results of processing the input in Figure 1 with the extraction ontology developed for marriage records in Figures 4–6. All the extracted information is correct. Two of the female names, however, are identified as NameU because the first name could be either gender. (Interestingly, the assignment of gender for these will be handled in the rules discussed below, where we will be able to reason that the gender must be female since the male in the marriage is known.) Because there were no betrothal keywords, all dates were determined to be marriage dates.

BetDate	MarDate	NameM	NameF	NameU
	same day 1576	Nicholas Patch		Christian Denman
	26 JAN 1605	Richard Patch	Joan Lavor	
	26 SEP 1613	John Elliott	Joan Woodbery	
	7 AUG 1615	Thomas Prime	Maria Parry	
	29 JAN 1616	William Woodbery	Elizabeth Patch	
	2 MAY 1620	William Hillerd		Fortu: Patch
	17 SEP 1622	Nicholas Patch	Elizabeth Owlsey	
	22 JAN 1627	Richard Patch	Mary White	
	16 JAN 1630	Andrew Elliott	Joan Patch	
	12 FEB 1639	Andrew Elliott	Joan Pitts	

Figure 8. OntoES-extracted data from the source in Figure 1.

Applying a Latin version of the marriage extraction ontology to the data in Figure 2 yields the data in Figure 9. The Latin language makes little difference in the

extraction although the ‘Christinam’ given name has a Latin female ending so is identified as female. The double date is handled by extracting the second year.

BetDate	MarDate	NameM	NameF	NameU
	eodem die 1577	Nicholaus Patch	Christinam Denman	
	26 JAN 1605	Richard Patch	Joanna Lavor	
	26 SEP 1613	Johannes Elliott	Joanna Woodbery	
	7 AUG 1615	Thoms Prime	Maria Patch	
	29 JAN 1616	Wilhelmus Woodbery	Elizabetha Patch	
	2 MAY 1620	Wilhelmus Hillerd		Fortu: Patch
	17 SEP 1622	Nicholas Patch	Elizabetha Owlsey	
	22 JAN 1627	Richardus Patch	Maria White	
	16 JAN 1630	Andreas Elliott	Joanna Patch	
	12 FEB 1639	Andreas Elliott	Joanna Pitts	

Figure 9. OntoES-extracted data from the source in Figure 2.

3.5 OWL and RDF

We store the results OntoES extracts using the standard languages OWL (Ontology Web Language) and RDF (Resource Description Framework). Doing so makes our results directly usable on the Semantic Web. Additionally, it opens the door to being able to declare reasoning rules in Semantic Web rule languages and to being able to query both the base data and the inferred data with Semantic Web query languages. OntoES not only captures and canonicalizes data from unformatted records, but also automatically produces an OWL specification representing the ontological structure of the data and an RDF specification representing the data with respect to the OWL-specified schema.

These OWL and RDF specifications appear together in a single file. In this single file, name-space prefixes identify whether an item is OWL metadata (“owl” prefixed) or RDF data (“rdf” prefixed) or RDFS (RDF Schema declarations, “rdfs” prefixed).

Further, when we add rules, which we explain below, these specifications also appear in the same file as SWRL items, prefixed with “swrl”.

The OWL/RDF/SWRL file starts with a header that defines the contents and the resources used. Thereafter, the class and property declarations appear followed by the instance data. For our extraction ontology in Figure 4, there is a class declaration for each object set, a data-type declaration for each lexical object set, and an object-property declaration for each relationship set.

As examples, the following class declarations are for three of the object sets in Figure 4. The NameU object set is lexical and therefore also has the type declaration “&xsd;string”. The “&xsd;” entity prefix here refers to an XML-Schema namespace where the type “string” is defined. Note that the name for the container for instance values for NameU is NameUValue. OntoES generates names for value containers by creating an OWL data type property mapping from the lexical object set to a string. The data type property name is formulated by appending “Value” to the end of the lexical object set name (NameU).

```
<owl:Class rdf:ID="MarriageRecord"/>
<owl:Class rdf:ID="Person"/>
<owl:Class rdf:ID="NameU"/>

<owl:DatatypeProperty rdf:ID="NameUValue">
  <rdfs:domain rdf:resource="#NameU"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

ObjectProperty declarations tie related object sets together. Thus, for example the relationship set in Figure 4 between `Person` and `NameU` has the following

declarations, which define the `Person-NameU` object property together with its inverse, `NameU-Person.`

```
<owl:ObjectProperty rdf:ID="Person-NameU">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range  rdf:resource="#NameU"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="NameU-Person"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Lexical data instances have both an object identifier and a lexical value. The following declaration makes “Christian Denman” the string value for the instance and “NameU_0” the object identifier for the name “Christian Denman”. OntoES generates unique identifiers for object and value instances by appending numbers to object-set names. Within each object set the number for each instance is unique.

```
<NameU rdf:ID="NameU_0">
  <NameUValue rdf:datatype="&xsd;string">Christian Denman
  </NameUValue>
</NameU>
```

The following declarations tie instances together across relationship sets. Here, the first declaration ties Christian Denman, whose name is identified by NameU_0 to Person_10. Thus, Person_10 identifies Christian Denman, NameU_0 identifies Christian Denman’s name, and “Christian Denman” is the lexicalization of Christian Denman’s name. Subsequent declarations here connect Christian (Person_10) to MarriageRecord_7. Person_4, who is Nicholas Patch, also connects to MarriageRecord_7 and is the other person in the marriage record.

```
<Person rdf:ID="Person_10">
  <Person-NameU rdf:resource="#NameU_0" />
</Person>

<MarriageRecord rdf:ID="MarriageRecord_7">
  <MarriageRecord-Person rdf:resource="#Person_4" />
  <MarriageRecord-Person rdf:resource="#Person_10" />
</rdf:MarriageRecord>

<NameM rdf:ID="NameM_4">
  <NameMValue>Nicholas Patch</NameMValue>
</NameM>
<Person rdf:ID="Person_4">
  <Person-NameM rdf:resource="#NameM_4" />
</Person>
```

Note that OntoES only records facts it extracts. There is nothing here about husband or wife and nothing about the unknown gender for Christian, which must be female. Inference rules, which we discuss next, provide this additional information.

4. OWL Rules

Currently one of the best tools for producing and editing OWL rules is Protégé [http://protege.stanford.edu] using Pellet [http://clarkparsia.com/pellet/]. As Semantic Web reasoning tools improve, we should be able to take advantage of them as well by maintaining our rules in the standard SWRL format.

4.1 Rule Declarations

The format of a SWRL rule is "body implies head." SWRL rules are based on Datalog, which in turn is based on Prolog, both longstanding logic languages. So that rules are both decidable and tractable, we limit heads to be single atoms and bodies to be conjunctions of atoms. All variables are universally quantified and variables that appear in the head must also appear in the body.

As a simple example, suppose we wish to identify the people whose names are extracted as NameU instances from the Petherton marriage records as husbands. A person *x* has the role of husband in these marriage records if *x* associates with a name *y* classified as a male name in the Person-Name relationship set. We write this rule in SWRL as follows:

```
Person-NameM(?x,?y) -> Husband(?x)
```

In SWRL syntax the body of the rule is to the left of implication arrow and the head is to the right. A question mark precedes variables. In the Person-NameM relationship set, variable *?x* matches with person identifiers like Person_10 or Person_4 seen in previous examples. There will only be a value for the variable *?y*, however, if *?y* is the identifier for a name in the NameM object set. As a result, Husband(*?x*) is only

true for substitutions for the variable *?x* that associate with a male name. In the underlying XML, this rule appears as follows. Note that the rule head is defined first, followed by the rule body. Also, we define an OWL class for Husband, which is introduced in the head of the rule.

```
<owl:Class rdf:ID="Husband"/>

<swrl:Imp rdf:ID="Def-Husband">
    <swrl:head>
      <swrl:AtomList>
        <rdf:rest rdf:resource="&rdf;nil"/>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate rdf:resource="#Husband"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:head>
    <swrl:body>
      <swrl:AtomList>
        <rdf:rest rdf:resource="&rdf;nil"/>
        <rdf:first>
          <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="
                  #Person-NameM"/>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:argument2 rdf:resource="#y"/>
          </swrl:IndividualPropertyAtom>
        </rdf:first>
      </swrl:AtomList>
    </swrl:body>
  </swrl:Imp>
```

The following is a similar rule for the wife role.

```
Person-NameF(?x,?y) -> Wife(?x)
```

These husband and wife rules are correct but do not cover all the possibilities. When the gender for a name is unknown, we can reason that the name is either male or female based on knowing the gender of the spouse name. The following rules declare a person, whose gender is unknown by the person's given name, to be a husband if the spouse name is female and vice versa.

```
Person-NameU(?x,?y) ∧ Person-NameF(?w,?v)
    ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?w)
        -> Husband(?x)

Person-NameU(?x,?y) ∧ Person-NameM(?w,?v)
    ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?w)
        -> Wife(?x)
```

Now, given husband and wife roles, we can reason that *x* is the husband of *y* if *x* is a husband, *y* is a wife, and they are connected by the same marriage record, as follows.

```
Husband(?x) ∧ Wife(?y)
    ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?y)
        -> HusbandOf(?x,?y)
```

This rule depends on other rules. In general, we can chain rules together to any depth. We can also make them recursive, so that they depend on themselves—an ideal way to compute AncestorOf.

4.2 Rule Application

To query the extracted and inferred data, we write queries in SPARQL and SPARQL-DL, which are Semantic Web standards. SPARQL lets us query base facts, while SPARQL-DL lets us query inferred facts as well.

We can, for example, query for marriage records between January of 1615 and December of 1625 with the following query. Prefixes (":" and "xsd:" in our example) shorten the body of the query. The prefix URI is substituted wherever the corresponding prefix appears. In this query we ask for four variables: Date, NameM, NameF, and NameU. The WHERE clause finds MarriageRecord-Event RDF triples[4] that are linked to Event-MarDate triples, which in turn link to MarDateValue triples; the filter clauses eliminate triples whose linked MarDateValue is prior to January 1615 or subsequent to December 1625. The optional clauses additionally look for linked NameM, NameF, and NameU values. Since each phrase is optional, NameM, NameF, and NameU may be null in the query result.

```
PREFIX : <http://www.deg.byu.edu/ontology/Marriage#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?Date ?NameM ?NameF ?NameU
WHERE
{
    ?Mr :MarriageRecord-Event ?Ev .
    ?Ev :Event-MarDate ?Md .
```

[4] RDF stores all data as triples of the form ⟨subject, predicate, object⟩.

```
        ?Md :MarDateValue ?Date .
          FILTER(xsd:integer(?Date) >= 1615001) .
          FILTER(xsd:integer(?Date) <= 1625365) .
        OPTIONAL { ?Mr   :MarriageRecord-Person ?Husb .
                   ?Husb :Person-NameM ?Nm .
                   ?Nm   :NameMValue ?NameM } .
        OPTIONAL { ?Mr   :MarriageRecord-Person ?Wife .
                   ?Wife :Person-NameF ?Nf .
                   ?Nf   :NameFValue ?NameF } .
        OPTIONAL { ?Mr   :MarriageRecord-Person ?Unk .
                   ?Unk  :Person-NameU ?Nu .
                   ?Nu   :NameUValue ?NameU }
    }
    ORDER BY ?Date
```

Applying this query to the data extracted from the marriage records in Figure 1 yields the following results:

```
Query Results (4 answers):
Date       | NameM              | NameF              | NameU
=================================================================
"1615219" | "Thomas Prime"    | "Maria Parry"     | <<null>>
"1616029" | "William Woodbery"| "Elizabeth Patch" | <<null>>
"1620123" | "William Hillerd" | <<null>>           | "Fortu: Patch"
"1622260" | "Nicholas Patch"  | "Elizabeth Owsley"| <<null>>
```

Suppose we wish to find the husband of Christian Denman. OntoES does not extract this information from the data in Figure 1. The information can, however, be inferred from the extracted data. The rules in the previous section along with the base information are almost enough. Without looking, we do not know whether Christian is in the set of female names or male names or names with unknown gender. Since in general this information is not known *a priori*, and also since we simply want to deal

with names independent of their classification, we can add the following nine simple rules.

```
NameM(?x) -> Name(?x)
NameF(?x) -> Name(?x)
NameU(?x) -> Name(?x)
NameMValue(?x,?y) -> NameValue(?x,?y)
NameFValue(?x,?y) -> NameValue(?x,?y)
NameUValue(?x,?y) -> NameValue(?x,?y)
Person-NameM(?x,?y) -> Person-Name(?x,?y)
Person-NameF(?x,?y) -> Person-Name(?x,?y)
Person-NameU(?x,?y) -> Person-Name(?x,?y)
```

With these rules, along with the rules in the previous section, we can pose the query "Who is the husband of Christian Denman?" Written in SPARQL-DL the query appears as follows. Note the use of the rules HusbandOf, NameValue, and Person-Name in the WHERE clause.

```
PREFIX : <http://www.deg.byu.edu/ontology/Marriage#>

SELECT ?Husband
WHERE
{
  ?X :NameValue "Christian Denman" .
  ?Y :Person-Name ?X .
  ?W :HusbandOf ?Y .
  ?W :Person-Name ?V .
  ?V :NameValue ?Husband
}
```

This query produces the following results over the extracted data in our running example:

```
Query Results (1 answers):
Husband
================
"Nicholas Patch"
```

For the data in Figure 1, this query result is correct, but we also note that in significantly larger files, it is likely that more than one Christian Denman may appear. In this case, of course, the query yields a list of all the husbands married to a Christian Denman. More precise queries that would provide marriage-date ranges and parish locations would be needed in this case. But this is exactly what having data in Semantic Web standard formats allows—the possibility to have all the data at one's disposal and the possibility to use it all to assist in reasoning over the base data.

5. Experimental Results and Implementation Status

5.1 Extraction Results

We created extraction ontologies for marriage records, birth records, and death records, and we applied them to data from three countries, England, the United States, and Denmark. The files we processed contained 967 marriage records, 4505 birth/christening records, and 4801 death/burial records. Each record contained multiple entities (e.g., names, dates, occupations, ages). All together, the records contained 28,659 entities.

Recall is a measure of what documents that are relevant are retrieved. Of these entities, OntoES correctly extracted 27,831, for a recall ratio of 96.9%. Precision is a measure of the percentage of retrieved documents were relevant. OntoES incorrectly extracted 291, yielding a precision ratio of 27,831/(27,831+291) = 99.0%.

	MARRIAGES	ENTITIES	RECALL	PERCENT	ERRORS	PRECISION
English	188	594	588	99.0%	8	98.7%
American	608	1824	1630	89.4%	34	98.0%
Danish	171	543	538	99.1%	10	98.2%
	BIRTHS					
English	3153	9489	9394	99.0%	61	99.4%
American	675	2055	1809	88.0%	33	98.2%
Danish	677	2061	2042	99.1%	15	99.3%
	DEATHS					
English	3458	8675	8589	99.0%	83	99.0%
American	510	1305	1148	88.0%	28	97.6%
Danish	833	2113	2093	99.1%	19	99.1%

Table 1. Extraction result detail.

Precision results were high for all data sets individually as well as collectively. If OntoES extracted an entity, it was usually extracted correctly. Recall results were mixed. For both English and Danish records, recall was high, averaging above 95%, but for the American records, recall averaged only about 88%.

For these American records, which were taken from Beverly, Massachusetts town records, there were special recall problems. The town records were constructed from vital, church, and cemetery records so that it is not unusual to have a great deal of duplicate information in parentheses or brackets like the following birth record:

```
WOODBURY, Charles Henry [Charles William, P. R. 4.], s. Henry
[housewright. dup.j and Henrietta (Galloup), Dec. 4, 1845.
```

In this case "Charles William" was missed as an alternate name for the same person as "Charles Henry" and was identified as another child, possibly a twin. A search for a surname for "Henry" incorrectly found "[housewright". The mother's name "Henrietta (Galloup)" was extracted, but with parentheses. Although accurate, the parentheses should be removed, but we have not written a method to postprocess names to remove anomalies.

As an interesting aside, we report a few insights on our development of the extraction ontologies themselves. For English and Danish data, the accuracy for our initial attempts at extraction of the first few dozen records averaged 78%. When the lexicons and terms were expanded, the accuracy rose to 99%, as reported. We conclude that for many data sets, experts can successfully build and improve extraction ontologies to attain near perfect accuracy. This is not the case for all

genealogical records, however. Our initial attempts to extract from the American town records averaged about 82%. Subsequent recall, however, never exceeded 88%.

5.2 Results of Rule Execution

In the rules section (Section 4), we described and illustrated several rules for the marriage extraction ontology (Figure 4). Altogether for the marriage ontology, we declared 21 rules of which 9 bring gendered names into a single name. (These 21 rules are in Appendix A.) Executing these 21 rules over the extracted information for the 10 marriage records in Figure 1 generated 120 inferred facts (60 inferred from the 12 non-name-simplification rules and 60 inferred from the 9 name simplification rules).

Examples of inferred facts based on the data in Figure 1 include:

```
HusbandOf(Person_4,Person_10)
Husband(Person_4)
Wife(Person_10)
Person_Name(Person_4,NameM_4)
Person_Name(Person_10,NameU_0)
NameValue(NameM_4,”Nicholas Patch”)
NameValue(NameU_0,”Christian Denman”)
```

These inferred facts are derived from the following extracted facts, which here are written both as abstract facts and in the generated OWL syntax:

```
NameUValue(NameU_0,“Christian Denman”)

   <NameU rdf:ID="NameU_0">
        <NameUValue>Christian Denman</NameUValue>
   </NameU>

Person-NameU(Person_10,NameU_0)

   (<Person rdf:ID="Person_10">
        <Person-NameU rdf:resource="#NameU_0" />
   </Person>
```

```
MarriageRecord-Person(MarriageRecord_7, Person_4)
MarriateRecord-Person(MarriageRecord_7, Person_10)


   <MarriageRecord rdf:ID="MarriageRecord_7">
        <MarriageRecord-Person rdf:resource="#Person_4" />
        <MarriageRecord-Person rdf:resource="#Person_10" />
   </rdf:MarriageRecord>

NameMValue(nameM_4,”Nicholas Patch”)

   <NameM rdf:ID="NameM_4">
        <NameMValue>Nicholas Patch</NameMValue>
   </NameM>

Person-NameM(Person_4,NameM_4)

   <Person rdf:ID="Person_4">
        <Person-NameM rdf:resource="#NameM_4" />
   </Person>
```

Whether inferred facts are correct depends on whether the rules are correctly specified and whether the base extracted facts are, themselves, correct. Given that the rules are correct, then since the extraction ontology correctly extracted all base facts from the 10 marriage records in Figure 1, all 120 inferred facts are correct.

In addition to the rule set for marriages, we defined rules for an Event ontology covering births and deaths. We declared 30 rules for the event ontology of which 18 were for name simplification. (These 30 rules are in Appendix B along with our ontology for these events.) We applied these 30 rules to the extracted information of 10 records (see Appendix B) of which 3 were christenings and 7 were burials. Altogether, applying the 30 rules generated 65 inferred facts (14 inferred from the 12 non-name-simplification rules and 51 inferred from the 18 simplification rules).

Table 1 shows that we extracted information for 10,273 records—967 marriage records, plus 4505 birth/christening records, plus 4801 death/burial records. We estimate that for the 967 marriages, the prototype system would infer 34,812 facts (967 records, times an estimated average of 6 inferred objects per record, times 3

simplification rules per name, times 2 for the average number of names in each record) and that for the 9306 birth, christening, death, and burial records, the prototype system would infer 335,016 facts (9306 records, times an estimated average of 6 inferred facts per record, times 3 simplification rules per name, times 2 for the average number of names per record). For all records in our source data, we therefore estimate that there would be a total of 458,292 inferred facts.

Results also show an increase in the size of files. Table 2 shows how OWL files grow with the addition of rules. The triples are counted; the lines in the OWL file are counted; and the size of the OWL file in kilobytes are measured and compared.

	MARRIAGE	**21 rules**		**EVENT**	**30 rules**	
	Triples	**OWL (# lines)**	**OWL File (kilobytes)**	**Triples**	**OWL (# lines)**	**OWL File (kilobytes)**
OWL File	814	498	14	2232	1405	15
W/Rules	1009	785	31	2983	1873	75
Difference	195	287	17	751	468	60
Increase	23.96%	57.63%	121.43%	33.65%	33.31%	400.00%

Table 2. Size results of adding rules to OWL files

5.3 Implementation Status and Future Work

We have implemented an ontology editor that lets us create ontology structures (like the one in Figure 4), add instance recognizers for each object set in an ontology structure (e.g., Figure 5), and specify canonicalization algorithms (e.g., Figure 6).

Using the ontology editor, we have created the extraction ontologies for this project as well as a few dozen others for different projects. We have also implemented translators that convert much the results of applying extraction ontologies to Semantic Web languages, OWL and RDF. We use Protégé [http://protégé.stanford.edu] which, in turn, uses Pellet [http://clarkparsia.com/pellet/] as our OWL reasoner. We load our OWL/RDF data files into Protégé and use its SWRL interface to create our rules. We then query the rules and base data with SPARQL-DL [SP07], a reasoner for OWL-DL (a decidable subset of OWL), which is based on SPARQL, a query language for RDF data.

With respect to highly relevant work completed and underway in the ontology workbench, we mention four features of interest: (1) annotation, (2) free-form queries, (3) results linked to original sources, and (4) explanatory reasoning chains. When we extract information, we keep annotation information—source documents and location information for each item extracted. Source documents include images, where the location information is a bounding box for items extracted originally via OCR or manually. Then, when we query extracted information, we intercept and rewrite the query so that it also picks up the annotation information so that when it displays results they are all clickable items. When a user clicks on a result, the system retrieves original documents, preprocesses them using the annotation information so that the query-result information is highlighted and then displays the page, scrolled to the part of the document where the information is highlighted. As an example, suppose a user types in the query:

```
Who is the husband of Christian Denman?
```

The highlighting on the query marks the items the system "understands" and is able to map to a populated extraction ontology. The system generates and executes a SPARQL-DL query, which returns the results:

Nicholas Patch

When a user now clicks on the result, Nicholas Patch, the system displays the highlighted source document in Figure 10 and the reasoning chain in Figure 11. Currently, our Ontology Workbench implementation provides for annotations, including annotations for images. It supports free-form queries but only over base facts with SPARQL (not SPARQL-DL). And it yields results that are clickable and returns documents with results highlighted. These implemented components have not, however, been integrated into the workbench in such a way that they are accessible by the genealogical prototype we are presenting here. Completing this integration is a near-term goal. No tool within the workbench displays reasoning

South Petherton Marriages

same day 1576 Nicholas Patch and Christian Denman
26 Jan 1605 Richard Patch and Joan Lavor
25-Sep 1613 John Elliott and Joan Woodbery
7-Aug 1615 Thomas Prime and Maria Parry
29-Jan 1616 William Woodbery and Elizabeth Patch
2-May 1620 William Hillerd and Fortu: Patch
17-Sep 1622 Nicholas Patch and Elizabeth Owsley
22-Jan 1627 Richard Patch and Mary White
15-Jan 1630 Andrew Elliott and Joan Patch
12-Feb 1639 Andrew Elliott and Joan Pitts

Figure 10. Highlighted document for sample query.

chains like the one in Figure 11. Since we use tools developed by others for rules and reasoning, and since these tools do not have mechanisms for storing, retrieving, and displaying reasoning chains as we wish, we are not on the verge of being able to support this functionality. As a future work item, we wish to bring the entire rule-specification interface and all the processing of the rules inside the workbench. At that time, we will have full control of rules and reasoning and will be able to support explanations such as the one in Figure 11.

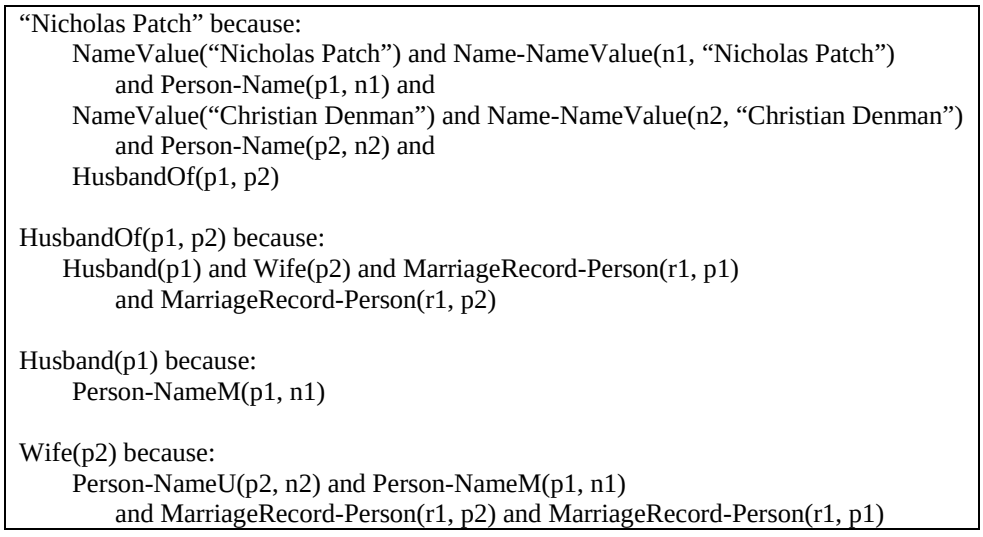


Figure 11. Reasoning chain in display form and with instance values inserted for variables.

Future work for the genealogical aspects of this work includes extending rules to identify the same person in different records which would support full family-linking based on facts extracted from collections of genealogical records. For example, by identifying fathers in different christening records as the same person, the father of one child would become the father of several children. This would also allow

families of several generations to form with the development of additional family rules

To accommodate additional genealogical records, extraction and canonicalization rules need to be expanded to include all possible date formations including those where the year is given only when changed. Culture-specific feast dates would be added as well as other culture-specific rules such as different naming traditions.

To accommodate the expected volume of genealogical records, there would also need to be work done on handling the increase in the size of the RDF database. This may be handled over time with the growth of technology and the increase of the size of storage and memory. Ultimately, techniques devised for modern search-engine technology would be needed to manage the data.

6. Conclusions

6.1 Thesis Contributions

Work accomplished for this thesis includes:

- creation of extraction ontologies,
- development of recognizers and lexicons for the various types of entities to be extracted,
- implementation of canonicalization routines,
- tuning extraction ontologies to achieve near perfect recall and precision,
- specifying rules for inferring family relationships,
- making various components of the complex system work together, and
- breaking new ground in developing a semantic web application for genealogy.

The best contribution of this work is the creation of a complete pipeline for processing unstructured, machine-readable vital records into a relationship-linked database. The data moves through an ontology with rules that identify persons at events for a specific time and place, standardizes the dates, labels the roles and relationships of the participants, and logically infers more facts. This process has never been previously assembled to a logical and useful solution.

Each of the tools is a Semantic Web tool which promise, as they mature in development, to fit well together and to further enhance the processing abilities of each other. Hopefully a platform will be developed where each tool can be available and where specific lexicons and rules peculiar to a specific location can be easily switched in and out.

Developing recognizers for feast dates was particularly challenging because of the numerous formats and abbreviations for each feast date and the sheer large number of feast dates which include saints' days. The formats and abbreviations were handled by recognizing the first few letters of the common dates. For example, Trinity Sunday, Trinitatis, Trin. or Tr. are recognized as 'Tr'.

Developing rules was technically challenging because the support systems are themselves early prototypes. SWRL rules could be edited in Protégé, but only in the earlier versions. As of this writing, the latest Protégé version 4.1 still does not have the SWRL tab available. At first, we copied the rules out of Protégé into the OWL file produced by OntoES. Then we discovered that it was much simpler to load the OWL file into Protégé 3.4 and allow the software to enter the rules. This method does require some deletion of code added by Protégé, but it works better than the alternatives.

OntoES also lacked the linking records for linking sections of the rules. At present we solved this problem by adding these lines of OWL code by hand, but the generation of these lines of code will soon be automated.

In the end, the work for the thesis shows that these ideas can be deployed on the semantic web. All generated or developed code is OWL, RDF, and SWRL, which are semantic web standards. The prototype developed can be an example of a semantic web application. Since the semantic web itself, is only in its initial stages of development, having good sample applications is critical to its success.

6.2 Future Outlook

The applications of the ideas embodied in this prototype to the field of Family History are that a full industrial-strength work-up of this prototype would:

- **Speed up data indexing** — The machine could do more of the work, changing the human task from manual indexing to editing the indexes already completed.
- **Make producing a full index easier** — Is it as easy to produce an index of all genealogical entities as it is to index only a few. Residence, for example, is a helpful indicator to determine which father is which when the father's names are the same, but this information is rarely extracted. Automated indexing makes it easier to include all information, and additional ground information makes it easier to create rules to postulate facts of interest.
- **Ground the index in original documents** — The prototype retains trace-back information that enables a link to the original primary record with a simple click. Explanations of any reasoning chains used to derive inferred information are also possible.
- **Provide for inferred facts** — The addition of rules, most likely provided by experts as a one-time effort for the various document types, can lead to a myriad of new facts and plausible facts to be checked and confirmed by interested family history researchers.
- **Simplify as well as augment record search** — It is possible to deploy all the information, both ground facts and inferred facts, as a Semantic Web application. This can make search for primary genealogical records quick and

easy and far more comprehensive than current techniques. The addition of free-form queries makes the system usable by untrained users.

- **Help link records and form family groups and ancestral lines** — Inferred family group and ancestral lines are possible. However, a missing ingredient, which we have not addressed, is to be able to match the same person from one record to another. Names and other information vary in many ways and the data itself is uncertain. Deploying the data as a Semantic Web application as we have described here should make this easier both because more information is readily available for reasoning and because of the automated trace-back through explained reasoning chains to original records.

References

[ECJ+99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith, Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering,* 31(3):227-251, November 1999.

[Emb80] D.W. Embley, Programming with data frames for everyday data items. *Proceedings of the 1980 National Computer Conference*, pages 301-305, Anaheim, California, May 1980.

[Emb04] D.W. Embley, Towards semantic understanding: An approach based on information extraction ontologies. In *Proceedings of the Fifteenth Australasian Database Conference*, pages 3-12, Dunedin, New Zealand, January 2004.

[MNS02] A. Maedche, G. Neumann, and S. Staab, Bootstrapping an ontology-based information extraction system. *Intelligent Exploration of the Web*, pages 345-359, Physica-Verlag GmbH, Heidelberg, Germany, *2002.*

[Qua03] D. Quass, Perspective on research problems in family history from the LDS Family and Church History Department, *Family History Technology Workshop,* Provo, Utah, March 2003. (http://www.fht.byu.edu/prev_workshops/workshop03/).

[Smi69] F. Smith, F. and A. Thomsen, *Genealogical Guidebook & Atlas of Denmark*. Bookcraft, Salt Lake City, Utah, 1969.

[SP07] E. Sirin and B. Parsia, SPARQL-DL: SPARQL query for OWL-DL, Proceedings of the 3rd OWL Experiences and Directions Workshop, Innsbruck, Austria, June, 2007.

[TE09] C. Tao and D.W. Embley, Automatic hidden-web table interpretation, conceptualization, and semantic annotation, *Data & Knowledge Engineering*, 68(7):683-703, July 2009.

[TEL09] C. Tao, D.W. Embley, and S.W. Liddle, FOCIH: Form-based ontology creation and information harvesting, *Proceedings of the 28th International Conference on Conceptual Modeling*, pages 346-359, Gramado, Brazil, November, 2009.

[TEL+05] Y.A. Tijerino, D.W. Embley, D.W. Lonsdale, Y. Ding, and G. Nagy, Toward ontology generation from tables, *World Wide Web: Internet and Web Information Systems*, 8(3):261-285, September, 2005.

[Vic06] M. Vickers, Ontology-based free-form query processing for the Semantic Web. Master's thesis, Computer Science Department, Brigham Young University, Provo, Utah, June 2006.

Appendix A. Marriage

A.1 Sample Data

South Petherton Marriages

same day 1576 Nicholas Patch and Christian Denman
26 Jan 1605 Richard Patch and Joan Lavor
25-Sep 1613 John Elliott and Joan Woodbery
7-Aug 1615 Thomas Prime and Maria Parry
29-Jan 1616 William Woodbery and Elizabeth Patch
2-May 1620 William Hillerd and Fortu: Patch
17-Sep 1622 Nicholas Patch and Elizabeth Owsley
22-Jan 1627 Richard Patch and Mary White
15-Jan 1630 Andrew Elliott and Joan Patch
12-Feb 1639 Andrew Elliott and Joan Pitts

Figure A.1. South Petherton marriages from GENUKI web page.

South Petherton Marriages

1576/1577 eodem die Nicholaus Patch Christinam Denman
26 Jan 1605 Richard Patch et Joanna Lavor
1613 Septembris 26 Johannes Elliott et Joanna Woodbery matrimonis cominguntur
1615 Augusti 7 Thoms Prime et Maria Patch matrimonio cominguntur
1616/1617 Januarij 29 Wilhelmus Woodbery et Elizabetha Patch matrimonio cominguntur
1620 Maij 2 : Wilhelmus Hillerd et Fortu: Patch
1622 Septembris 17 Nicholas Patch et Elizabetha Owsley matrimonio cominguntur
1627/1628 Januarij 22 : Richardus Patch et Maria White matrimonio cominguntur
1630/1631 Januarij 15 Andreas Elliott et Joanna Patch matrimonio cominguntur
1639/1640 Februarij 12 Andreas Elliott et Joanna Pittes matrimonio cominguntur

Figure A.2. South Petherton marriages transcribed directly from the parish register.

Vital Records of Beverly, Essex, Massachusetts

WOODBERRY, Abbigaile, and Richard Ober, Dec. 26, 1671.
WOODBERRY, Abigail, of Salem, and William Ellingwood, at Salem, Feb. 14, 1718.
WOODBERRY, Abigail, and Joseph Pickott, jr., Dec. 13, 1768.*
WOODBERRY, Abigail, and John Lennon, Dec. 23, 1819.*
WOODBERRY, Abigail, a. 39 y. 6 m. 14 d., d. Freeborn and Joanna, and John Woodberry, widr., a. 37 y. 4 m., cordwaine
WOODBERRY, Alvah, and Nancy B. Dennis, Oct. 21, 1832.*
WOODBERRY, Andrew, 2d, and Leafee Poland of Hamilton, Nov. 19, 1811.*
WOODBERRY, Andrew, and Hannah Trask, July 29, 1819.*
WOODBERRY, Andrew, and Mary Jane Hanners, Dec. 4, 1834.*
WOODBERRY, Ann, and James Edmester of Maiden, Apr. 3, 1828.*
•Intention also recorded.

Figure A.3. Beverly marriages from scanned images.

MAGLEBY, PRAESTO, DENMARK MARRIAGES 1691 1700

Page 248
1694
1691 Dend 28 September blef Indladt i det Hellige Egteskab K: L: PEDER NIELSEN BERG KIRSTINE JESPERSDATTER af Butzemk
1694 Trolovet for min Onkomste Dom Cantate PEDER CLEMMINGSEN og KIRSTEN SOFFRENSDATTER
1694 Viet siden min Ankomiste Dom: 17 p: Trin: PEDER CLEMMINGSEN og KIRSTEN SOFRENSDATTER
1694 Trolovet for min ankomist d 8 Jun: PEDER MOGENSEN og KIRSTEN PEDERSDATTER af Butene Viet Sondag Dend 19 Jun:

Page 249
1694 Troloved Dom 19 post Trin: NIELS HANSEN RENDER af Mandemk og JOHANNA BUNDISD: Bunde Nielsen i Bidsinge, Niels Frimand
1694 Viet Dom: 25 post Trin: NIELS HANSEN af Mandemk og JOAHNNA BUNDISD:
1694 Troloved d 31 Octobr. KNUD MICHELSEN af Soemk og KIRSTEN PEDERSDATTER
1694 Viet Dom: 2 Adventus KNUD MICHELSEN af Somk og KIRSTEN PEDERSDATTER

Figure A.4. Danish marriages transcribed directly from the microfilm.

A.2 Extraction Ontology

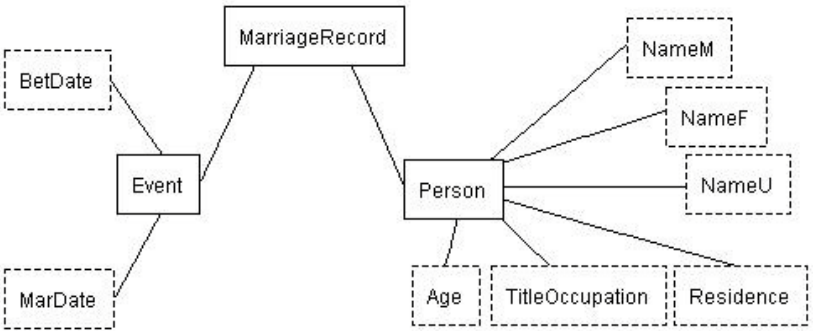


Figure A.5. Marriage ontology.

A.3 Rules

1. In a marriage record, a person *x* is a husband if *x* has a male name.

```
Person-NameM(?x,?y) -> Husband(?x)
```

2. In a marriage record, a person *x* is a wife if *x* has a female name.

```
Person-NameF(?x,?y) -> Wife(?x)
```

3. In a marriage record, if a person *x* has a name with unknown gender and the other person has a female name, then *x* is a husband.

```
Person-NameU(?x,?y) ∧ Person-NameF(?w,?v)
```

```
    ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?w)
        -> Husband(?x)
```

4. In a marriage record, if a person *x* has a name with unknown gender and the other person has a male name, then *x* is a wife.

```
Person-NameU(?x,?y) ∧ Person-NameM(?w,?v)
    ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?w)
        -> Wife(?x)
```

5. In a marriage record, if person *x* is a husband and person *y* is a wife, then *x* is the husband of *y*.

```
Husband(?x) ∧ Wife(?y) ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?y)
        -> HusbandOf(?x,?y)
```

6. In a marriage record, if person *x* is a wife and person *y* is a husband, then *x* is the wife of *y*.

```
Wife(?x) ∧ Husband(?y) ∧ MarriageRecord-Person(?z,?x)
    ∧ MarriageRecord-Person(?z,?y)
        -> WifeOf(?x,?y)
```

7. In a marriage record, if *y* is the marriage date and *x* is a person, then *y* is the date of marriage of *x*.

```
MarDate-Event(?y,?s) ∧ Event-MarriageRecord(?s,?t)
    ∧ MarriageRecord-Person(?t,?x)
        -> DateOfMarriage(?x,?y)
```

8. In a marriage record, if *y* is the betrothal date and *x* is a person, then *y* is the date of betrothal of *x*.

```
BetDate-Event(?y,?s) ∧ Event-MarriageRecord(?s,?t)
    ∧ MarriageRecord-Person(?t,?x)
        -> DateOfBetrothal(?x,?y)
```

9. In a marriage record, if person *x* is a husband, then *y* is the place of residence of *x*.

```
Husband(?x) ∧ Person-Residence(?x,?y)
        -> HusbandResidence(?x,?y)
```

10. In a marriage record, if person *x* is a wife, then *y* is the place of residence of *x*.

```
Wife(?x) ∧ Person-Residence(?x,?y)
        -> WifeResidence(?x,?y)
```

11. In a marriage record, if person *p* has an age *a* and a Julian marriage date m, then *y = (m div 1000) - a* is the birth year of *p*.

```
Person-Age(?p,?w) ^ AgeValue(?w,?a) ^
    Person-MarriageRecord(?p,?r) ^
    MarriageRecord-Event(?r,?e) ^
    Event-MarDate(?e,?m) ^ MarDateValue(?m,?v) ^
    swrlb:integerDivide(?z,?v,1000) ^
    swrlb:subtract(?y,?z,?a)
        -> BirthYearValue(?p,?y)
```

12. In a marriage record, if person *p* has an age *a* and a Julian betrothal date *m*, then *y = (m div 1000) - a* is the birth year of *p*.

```
Person-Age(?p,?w) ^ AgeValue(?w,?a) ^
    Person-MarriageRecord(?p,?r) ^
    MarriageRecord-Event(?r,?e) ^
    Event-BetDate(?e,?m) ^ BetDateValue(?m,?v) ^
    swrlb:integerDivide(?z,?v,1000) ^
    swrlb:subtract(?y,?z,?a)
        -> BirthYearValue(?p,?y)
```

A.4 Simplification Rules

Nine rules that simplify references to names:

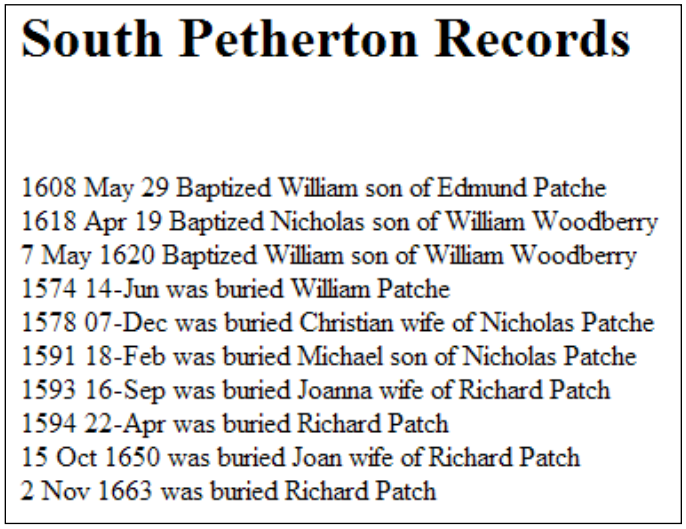
```
NameM(?x) -> Name(?x)
NameF(?x) -> Name(?x)
NameU(?x) -> Name(?x)

NameMValue(?x,?y) -> NameValue(?x,?y)
NameFValue(?x,?y) -> NameValue(?x,?y)
NameUValue(?x,?y) -> NameValue(?x,?y)
```

```
Person-NameM(?x,?y) -> Person-Name(?x,?y)
Person-NameF(?x,?y) -> Person-Name(?x,?y)
Person-NameU(?x,?y) -> Person-Name(?x,?y)
```

Appendix B. Event

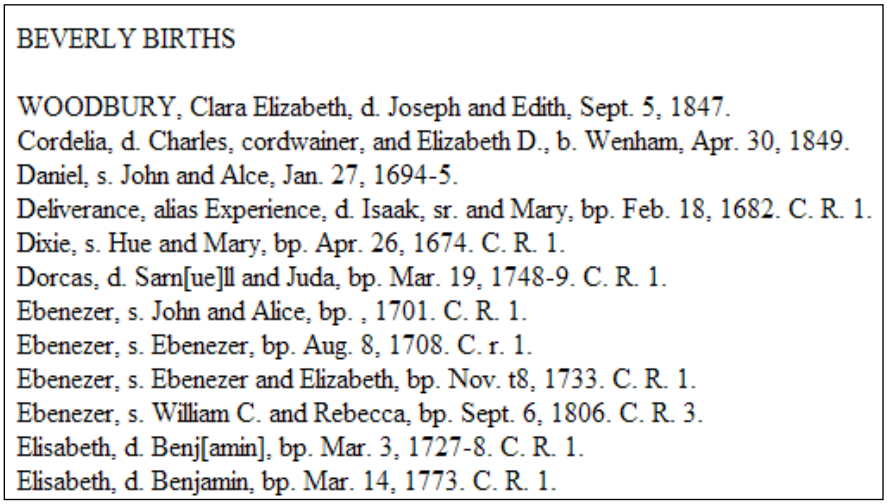
B.1 Sample Data



South Petherton Records

1608 May 29 Baptized William son of Edmund Patche
1618 Apr 19 Baptized Nicholas son of William Woodberry
7 May 1620 Baptized William son of William Woodberry
1574 14-Jun was buried William Patche
1578 07-Dec was buried Christian wife of Nicholas Patche
1591 18-Feb was buried Michael son of Nicholas Patche
1593 16-Sep was buried Joanna wife of Richard Patch
1594 22-Apr was buried Richard Patch
15 Oct 1650 was buried Joan wife of Richard Patch
2 Nov 1663 was buried Richard Patch

Figure B.1. South Petherton christenings and burials from GENUKI.



BEVERLY BIRTHS

WOODBURY, Clara Elizabeth, d. Joseph and Edith, Sept. 5, 1847.
Cordelia, d. Charles, cordwainer, and Elizabeth D., b. Wenham, Apr. 30, 1849.
Daniel, s. John and Alce, Jan. 27, 1694-5.
Deliverance, alias Experience, d. Isaak, sr. and Mary, bp. Feb. 18, 1682. C. R. 1.
Dixie, s. Hue and Mary, bp. Apr. 26, 1674. C. R. 1.
Dorcas, d. Sarn[ue]ll and Juda, bp. Mar. 19, 1748-9. C. R. 1.
Ebenezer, s. John and Alice, bp. , 1701. C. R. 1.
Ebenezer, s. Ebenezer, bp. Aug. 8, 1708. C. r. 1.
Ebenezer, s. Ebenezer and Elizabeth, bp. Nov. t8, 1733. C. R. 1.
Ebenezer, s. William C. and Rebecca, bp. Sept. 6, 1806. C. R. 3.
Elisabeth, d. Benj[amin], bp. Mar. 3, 1727-8. C. R. 1.
Elisabeth, d. Benjamin, bp. Mar. 14, 1773. C. R. 1.

Figure B.2. Beverly, Massachusetts births.

BEVERLY DEATHS

WOODBERRY, Malachi, decay, Nov. 20, 1839, a. 87 y. C. R. 1.
Maria A. (Porter), w. Francis, Jan. 16, 1842.
Mark, s. John and Elisabeth, farmer [old age. C. R. 5.], Feb. 25, 1847, a. 8r y. 5 m. 25 d.
Martha, w. Thomas, d. John and Martha Cleaves, Oct. 2, 1729), a. 26 y.
Martha, w. Benjamin, bur. Sept. 16, 1801, a. 65 1-2 y. C. R. 1.
Martha, consumption, bur. Dec. 2, 1808, a. 65 y. C. R. 1.

Figure B.3. Beverly, Massachusetts deaths.

.

[FHL Film#050,646] MAGLEBY, PRAESTO, DENMARK, DEATHS 1645-1671

Page 76
1645
Den 9 Julij bleff begraffvett Laurids Smeds liden datter i Somk: KIRSTINE
den 27 Julij Peder Hiortis daatter i Budzemarck: METTE
Den 21 Augusti ett u-ecte barn ved naffn HENNING
Den 23 Augusti Peder Brochis pige barn i Somk: ANNE
Den 3 Septembris Jorgen Jensens PIGE BARN Somk, som dode fras udj fodzslem
Dnica 23 post Trinit HANS VEFFE i Soemk
Den 28 Decembris Laurids Brochis hustrue i Soemk: KARENE

Figure B4. Danish deaths.

B.2 Extraction Ontology

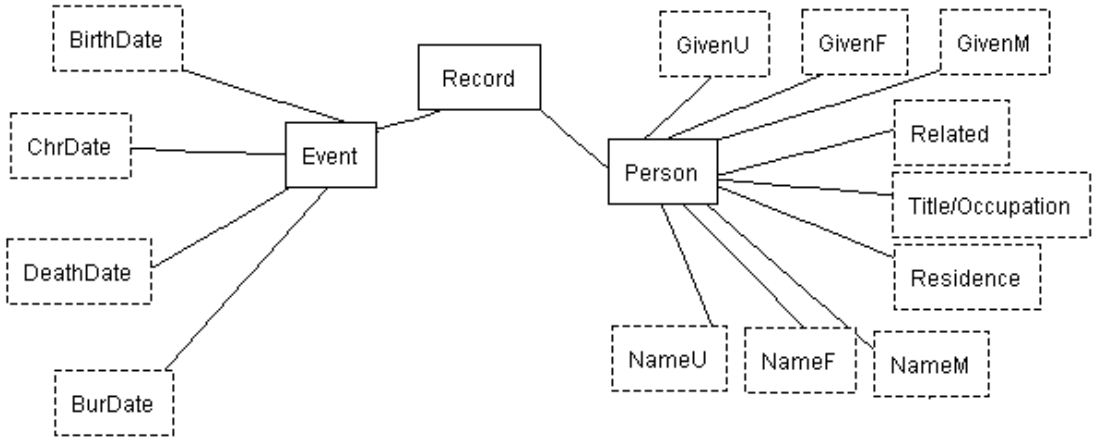


Figure B.5 Event Ontology.

B.3 Rules

1. In a birth record, a person *x* is a son if *x* has only a male given name.

```
BirthDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-GivenM(?u,?x)
        -> Son(?u,?x)
```

2. In a birth record, a person *x* is a daughter if *x* has only a female given name.

```
BirthDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-GivenF(?u,?x)
        -> Daughter(?u,?x)
```

3. In a christening record, a person *x* is a son if *x* has only a male given name.

```
ChrDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-GivenM(?u,?x)
        -> Son(?u,?x)
```

4. In a christening record, a person *x* is a daughter if *x* has only a female given name.

```
ChrDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-GivenF(?u,?x)
        -> Daughter(?u,?x)
```

5. In a christening record, a person *x* is a father if *x* has a male full name.

```
ChrDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-NameM(?u,?x)
        -> Father(?x)
```

6. In a christening record, a person *x* is a mother if *x* has a female full name.

```
ChrDate-Event(?r,?s) ∧ Event-Record(?s,?t)
    ∧ Record-Person(?t,?u) ∧ Person-NameF(?u,?x)
        -> Mother(?x)
```

7. In a record, if person *x* is a father and person *y* is a son, then *x* is the father of *y*.

```
Son(?y,?v) ∧ Person-Record(?v,?z)
    ∧ Person-Record(?u,?z) ∧ Father(?u,?x)
        -> FatherOf(?x,?y)
```

8. In a record, if person *x* is a father and person *y* is a daughter, then *x* is the father of *y*.

```
Daughter(?y,?v) ∧ Person-Record(?v,?z)
    ∧ Person-Record(?u,?z) ∧ Father(?u,?x)
        -> FatherOf(?x,?y)
```

9. In a record, if person *x* is a mother and person *y* is a son, then *x* is the mother of *y*.

```
Son(?y) ∧ Person-Record(?y,?z)
    ∧ Person-Record(?x,?z) ∧ Mother(?x)
        -> MotherOf(?x,?y)
```

10. In a record, if person *x* is a mother and person *y* is a daughter, then *x* is the mother of *y*.

```
Daughter(?y) ∧ Person-Record(?y,?z)
    ∧ Person-Record(?x,?z) ∧ Mother(?x)
        -> MotherOf(?x,?y)
```

11. In a record, if person *x* is a son and person *y* is a father, then *x* is the son of *y*.

```
Father(?y) ∧ Person-Record(?y,?z)
    ∧ Person-Record(?x,?z) ∧ Son(?x)
        -> SonOf(?x,?y)
```

12. In a record, if person *x* is a daughter and person *y* is a father, then *x* is the daughter of *y*.

```
Father(?y) ∧ Person-Record(?y,?z)
    ∧ Person-Record(?x,?z) ∧ Daughter(?x)
        -> DaughterOf(?x,?y)
```

B.4 Simplification Rules

Eighteen rules that simplify references to names:

```
NameM(?x) -> Name(?x)
NameF(?x) -> Name(?x)
NameU(?x) -> Name(?x)

NameMValue(?x,?y) -> NameValue(?x,?y)
NameFValue(?x,?y) -> NameValue(?x,?y)
NameUValue(?x,?y) -> NameValue(?x,?y)

Person-NameM(?x,?y) -> Person-Name(?x,?y)
Person-NameF(?x,?y) -> Person-Name(?x,?y)
Person-NameU(?x,?y) -> Person-Name(?x,?y)

GivenM(?x) -> Given(?x)
GivenF(?x) -> Given(?x)
GivenU(?x) -> Given(?x)

GivenMValue(?x,?y) -> GivenValue(?x,?y)
GivenFValue(?x,?y) -> GivenValue(?x,?y)
GivenUValue(?x,?y) -> GivenValue(?x,?y)

Person-GivenM(?x,?y) -> Person-Given(?x,?y)
Person-GivenF(?x,?y) -> Person-Given(?x,?y)
Person-GivenU(?x,?y) -> Person-Given(?x,?y)
```