AUTOMATIC EXTRACTION FROM AND REASONING ABOUT GENEALOGICAL RECORDS: A PROTOTYPE

Charla J. Woodbury,* David W. Embley,* and Stephen W. Liddle†
*Department of Computer Science
†Information Systems Department
Brigham Young University

ABSTRACT

There is great interest in family history research on the web and a great many competing genealogical websites that contain large amounts of data-rich, unstructured, primary genealogical records. The problem is that it is so labor-intensive even after making these records machine-readable, for humans to make the same records searchable. What we need are computer tools that can automatically produce indices and databases from the data-rich, unstructured genealogical records and can identify individuals and events, determine relationships, and put families together. We propose here a possible solution—a specialized ontology, built specifically for the information extraction of 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 a set of specialized extraction ontologies used to extract parish or town records, a marked-up version of the original document, a data file of individuals and events, and rules used to define family relationships and manipulate the data file. The solution also provides for the ability to query over the rules and data files. An evaluation of the prototype solution shows that the extraction has good recall and precision results and that inferred facts are correct.

1 Introduction

The many family history websites on the Internet are in high competition to provide large systems of primary records that are easily and quickly searched by naïve 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 the original census pages. 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.

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 the original primary documents. 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 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 fully successful, this means that every bit of genealogical information in the primary records can be used to qualify an individual and that all of this information is automatically indexed. 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. Expert logic could be used to make the machine do more of the work, both for extraction and indexing, as well as 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 the automation of the information extraction process on unstructured machine-readable genealogical records by:

- designing appropriate primary record extraction ontologies for family history records that produce formatted RDF data in OWL files (RDF and OWL are Semantic Web standards for data and metadata),
- adding rules and logic to the files that, when applied, properly label and begin to link primary genealogical data,
- constructing queries that show selected extracted data with the rules and logic applied, and
- testing and evaluating the results for accuracy.

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) Finally, 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 quickly 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 Ontology-Building

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. Once success-

fully 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.

```
MONTH LEXICON

1Ober       decembre
7ber        decembri
8ber        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
decembr
```

Figure 3. Sample partial lexicon listing words denoting months.

3.2 Ontology Models

3.2.1 Ontology Conceptual Model

The conceptual model component of the ontology model shows object sets 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 records associate with persons and events. The records being modeled in this ontology are marriage records. 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.

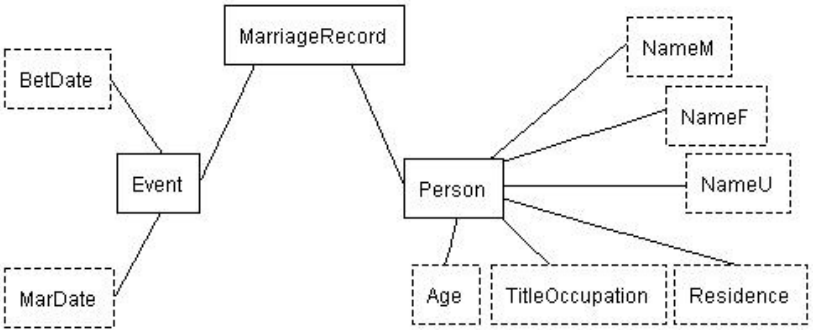

Figure 4. Marriage ontology.

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. Notice that there is a place for value phrases and another for keyword phrases, and that the data frame editor is currently open for value phrases. Value phrases describe data to be extracted. 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 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 six 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.

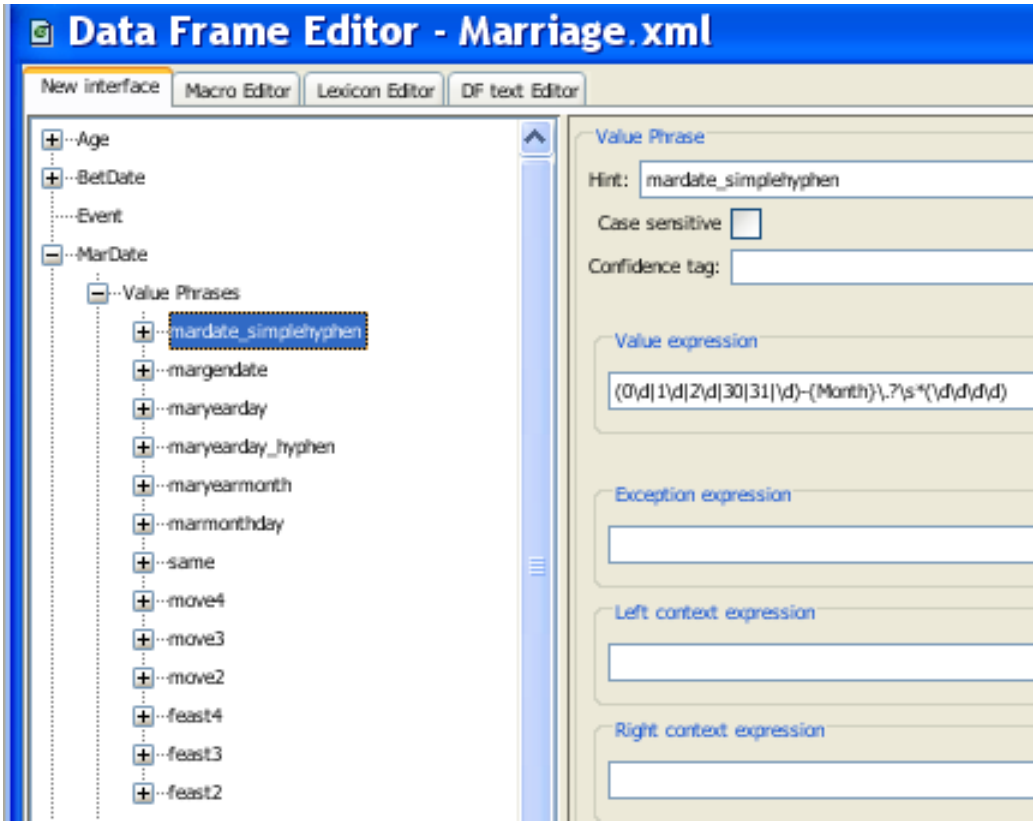

Figure 5. Dataframe example for the MarDate object set under Marriage ontology.

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 the computer to correctly disambiguate betrothal dates and marriage dates, which often occur in the same record.

3.2.3 Canonicalization of dates

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 Java methods to regularize dates, one for each "hint". OntoES then used the methods to determine values and manage 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 Julian dates in the 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 identifies 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" will be 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 age.

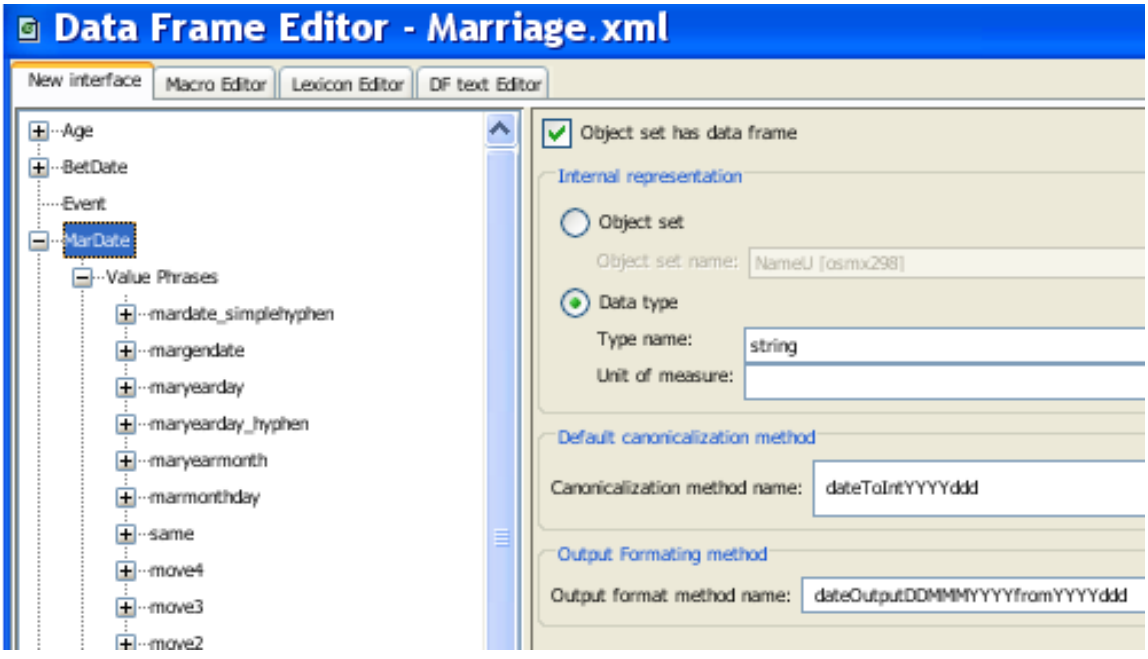

Figure 6. Data Frame Editor at object level.

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 Years 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 to "1581272".

Moveable feast dates, on the other hand, are not on 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:

```
1723 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. Using the year guide in Figure 7, the key for the year 1723 is '7'. Using that key and the feast day guide in Figure 8, "Septuagesima 1723" can be decoded as "24 Jan 1723". In a similar manner, all feast days can be translated into month and day.

Year	No.
1723	7
1724	26
1725	11
1726	31
1727	23
1728	7
1729	27
1730	19
1731	4
1732	23
1733	15
1734	35
1735	20

Figure 7. Sample year guide for decoding moveable feast days.

	SEPTUAGE-SIMA	SEXAGE-SIMA	QUINQUAGE-SIMA	INVOCAVIT	REMINI-SCERE	OCULI	LAETARE	JUDICA	PALMARUM	PASCHA
1	18 Jan	25 Jan	1 Feb	8 Feb	15 Feb	22 Feb	1 Mar	8 Mar	15 Mar	22 Mar
2	19 Jan	26 Jan	2 Feb	9 Feb	16 Feb	23 Feb	2 Mar	9 Mar	16 Mar	23 Mar
3	20 Jan	27 Jan	3 Feb	10 Feb	17 Feb	24 Feb	3 Mar	10 Mar	17 Mar	24 Mar
4	21 Jan	28 Jan	4 Feb	11 Feb	18 Feb	25 Feb	4 Mar	11 Mar	18 Mar	25 Mar
5	22 Jan	29 Jan	5 Feb	12 Feb	19 Feb	26 Feb	5 Mar	12 Mar	19 Mar	26 Mar
6	23 Jan	30 Jan	6 Feb	13 Feb	20 Feb	27 Feb	6 Mar	13 Mar	20 Mar	27 Mar
7	24 Jan	31 Jan	7 Feb	14 Feb	21 Feb	28 Feb	7 Mar	14 Mar	21 Mar	28 Mar
8	25 Jan	1 Feb	8 Feb	15 Feb	22 Feb	1 Mar	8 Mar	15 Mar	22 Mar	29 Mar

Figure 8. Sample feast day by year key.

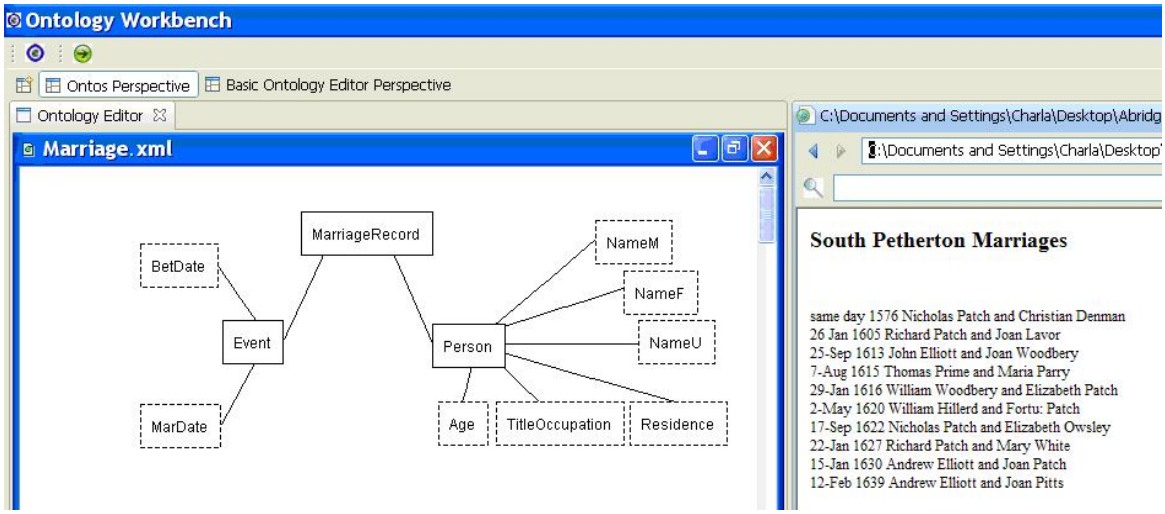

Figure 9. OntoES Workbench ready to run the extraction.

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. In general, 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. We used the "Calendar of Feast Days" compiled by Henry E. Christiansen as given in [Smi69] to build these 72 sequences.

Managing feast dates in this way means that feast dates can be extracted, converted to dates, and then stored in Julian format by linking to the proper canonicalization programs.

3.3 Running OntoES

Figure 9 shows the OntoES workbench 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 and the web page are selected, the extraction is begun by pressing the green extraction button which is at the far right at the top of the page.

3.4 Examples of the Ontological Results

Figure 10 shows the results of processing the input in Figure 1 with the extraction ontology developed for marriage records. 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 gen-

der 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.

Applying a Latin version of the marriage extraction ontology to the data in Figure 2 yields the data in Figure 11. 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
	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 10. OntoES-extracted data from the source in Figure 1.

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

3.4.4 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 automati-

cally 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 (Semantic Web Rule Language) items. SWRL items are prefied 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 name space 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 datatype property mapping from the lexical object set to a string. The datatype 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"/>
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</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 Christian. OntoES generates unique identifiers for object and value instances by appending a numbers to the names of the object sets. Within each object set the numbers for each item are 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, who is identified by NameU_0 to Person_10. Thus, Person_10 is Christian Denman. 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>

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

<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 the 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 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.

```
<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 to be a husband if the spouse name is female and vice versa.

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

Person-NameU(?x,?y) ∧ Person-NameM(?w,?v)
    ∧ Person-MarriageRecord(?x,?z)
    ∧ 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. 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.

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

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 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 .
    ?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>>
```

†The string "^^http://www.w3.org/2001/XMLSchema#string" appears here in the output.

Suppose we wish to find the husband of Christian Denman. This information is not directly extracted by the extraction ontology when applied to the data in Figure 1. It 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) -> NameValue(?x)
NameFValue(?x) -> NameValue(?x)
NameUValue(?x) -> NameValue(?x)

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 following query:

```
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"^^http://www.w3.org/2001/XMLSchema#string
```

For the data in Figure 1 this 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. Of these entities, OntoES correctly extracted 27,831, for a recall ratio of 97.1%. OntoES incorrectly extracted 291, yielding a precision ratio of 27,831/(27,831+291) = 99.0%.

Extraction accuracy was 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%.

	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.

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 [house-
wright. 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 Beverly Town records averaged about 82%. Subsequent recall, however, never exceeded 88%.

5.2 Implementation Status

The prototype system we have built is large (large for a proof-of-concept prototype) and comprehensive (includes not only tools for information extraction and reasoning with rules, as explained here, but also tools like TISP [TE09] for automatic table interpretation, FOCIH [TEL09] for form-based ontology creation and information harvesting, TANGO [TEL+05] for generating ontologies from table collections, and AskOntoES [Vic06] for processing end-user free-form queries). We call our larger prototype system the "Ontology Workbench" because it lets us experiment with tools for semi-automatically generating extraction ontologies and for applying extraction ontologies to annotate, organize, store, and query data in web documents. Through our ontology workbench we are exploring ways to superimpose a web of knowledge over the current web and thus enable a next-generation semantic web in which users can search the web with keywords as they do now but can also directly ask and and get answers to arbitrary free-form queries.

With respect to the work reported here, 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 12 and the reasoning chain in Figure 13.

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 12. Highlighted document for sample query.

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 return 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.

```
"Nicholas Patch" because:
    NameValue("Nicholas Patch") and Name-NameValue(n1, "Nicholas Patch")
        and Name(n1) is NameM(n1) and Person-NameM(p1, n1)
    NameValue("Christian Denman") and Name-NameValue(n2, "Christian Denman")
        and Name(n2) is NameU(n2) and Person-NameU(p2, n2)
Husband(p1) because:
    Person-NameM(p1, n1)
Wife(p2) because:
    Person-NameU(p2, n2) and Person-MarriageRecord(p2, r1)
        and MarriageRecord-Person(r1, p1) and Person-NameM(p1, n1)
HusbandOf(p1, p2) because:
    Husband(p1) and Wife(p1) and MarriageRecord-Person(r1, p1)
        and MarriageRecord-Person(r1, p1)
```

Figure 13. Reasoning chain based on the rules in Section 4, written in display form and with instance values inserted for variables.

No tool within the workbench displays reasoning 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 13.

6 Conclusions

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 computer could do more of the work, changing the human interface from that of indexer to that of editor of the indexing already completed by the computer.
- **Make production of a full index easier** — It is 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.
- **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 webpages. *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, *Genealogial 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*, Gramado, Brazil, November, 2009, pages 346-359.

[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.