# Ontological Deep Data Cleaning

Scott N. Woodfield[1], Spencer Seeger[1], Samuel Litster[1],
Stephen W. Liddle[1], Brenden Grace[1], and David W. Embley[1,2]

[1] Brigham Young University, Provo, Utah 84602, USA
[2] FamilySearch International, Lehi, Utah 84043, USA

**Abstract.** Analytical applications such as forensics, investigative journalism, and genealogy require deep data cleaning in which application-dependent semantic errors and inconsistencies are detected and resolved. To facilitate deep data cleaning, the application is modeled ontologically, and real-world crisp and fuzzy constraints are specified. Conceptual-model-based declarative specification enables rapid development and modification of the usually large number of constraints. Several field tests serve as evidence of the prototype's ability to detect errors and either resolve them or provide guidance for user-involved resolution. The results of a user study show the value of declarative specification for rapidly developing and modifying deep data cleaning applications.

**Keywords**: data quality, data cleaning, declarative constraint specification, conceptual-model-based deep data cleaning.

## 1 Introduction

*Data cleaning* improves data quality by detecting and removing errors and inconsistencies [10]. *Deep data cleaning* includes data cleaning but adds general constraints that serve to detect application-dependent semantic errors and inconsistencies. These general constraints can be either crisp or fuzzy and are often expressed probabilistically. Ontologies, which are shared, commonly agreed upon conceptualizations of a domain of interest [6], are a natural framework for these crisp and fuzzy semantic constraints. Within an ontology their purpose is to accurately characterize objects and data instances and their interrelationships. As such, they become the basis for *ontological deep data cleaning*.

In this paper we illustrate the principles and possibilities of ontological deep data cleaning in the realm of genealogy—an application not only for family history enthusiasts, but also for studying inherited diseases and for establishing public policy such as for intergenerational poverty. Among other possible applications, the particular data cleaning application we study is automatic information extraction from scanned and OCR'd historical documents, which exacerbates the data cleaning problem. Not only does the system have to wrangle with human errors in deciphering, reasoning about, and entering information taken from historical documents, but also the often wildly spurious errors computers make in automatic data extraction.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.) Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840. } Twins.
3. Donald McKenzie, b. 1840, d. 1843. }
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. 1860.
4. Emma Goble, b. 1862.

**Fig. 1.** Text Snippet from *The Ely Ancestry*[12], Page 421.

To illustrate ontologically deep data cleaning within our application, consider the sample page in Figure 1 taken from *The Ely Ancestry* [12]. When processed by an ensemble of information extraction engines [2], most of the genealogical information on the page was correctly extracted. The ensemble, however, did incorrectly associate Mary Ely and Gerard Lathrop (the first two numbered children in Figure 1) with four parents: Mary Eliza Warner, Joel M. Gloyd, Abigail Huntington Lathrop, and Donald McKenzie. The constraint checker in our implemented data cleaning tool not only flagged Mary Ely and Gerard Lathrop as having more than two parents, but also flagged Mary Eliza Warner as likely being too young to be the mother of Mary and Gerard since she would have respectively been only about 10 and 12 years old at the time of their births. After detection, our data-cleaning system considers the possibility of retracting assertions to correct problems and in this case can automatically retract the assertions linking Mary Ely and Gerard Lathrop to incorrect parents.

Preparation of data for import into a target repository is only one part of our genealogical deep cleaning application. Upon import, the first problem encountered is to reconcile the new data with the old. As a specific example, when we imported the data in Figure 1, a search for duplicates revealed that the three

Mary Elys asserted to be mothers of Abigail, William, and Charles Lathrop together with a particular Mary Ely in the repository and another Mary Eli in the repository were all possible duplicates. Our data cleaning system checks potential duplicates by doing a temporary merge and asking if any ontological constraints are violated. None were, and the temporary merge became permanent. A check for merging Mary Ely, the daughter of Abigail in Figure 1, with these merged Mary Elys failed based on the ontological constraint that it is impossible to be one's own ancestor. Our data cleaning tool can also generally check assertions in the repository. As Figure 2 shows, our cleaning system detected that Abigail's third great-grandmother, Sarah Sterling, died before a child of hers was born. When a violation is detected, the tool explains what may be wrong as Figure 2 shows.

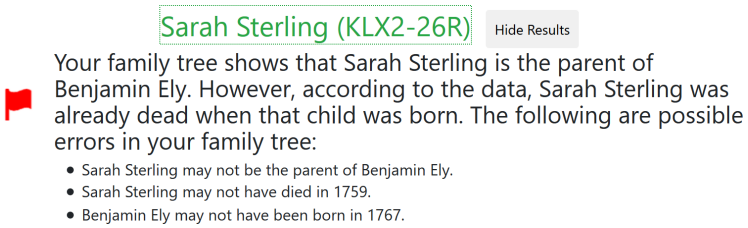



**Fig. 2.** Detected Genealogical Problem with Recommended Solutions.

Although our approach to data cleaning corresponds with the central ideas of detecting and removing errors [4, 10, 11], it differs from other data cleaning research in two fundamental ways: (1) it is deep—centered on ontological semantic constraints; and (2) for applications that of necessity must accommodate dirty data, it applies to cleaning the repository post-import as well as to cleaning pre-import and on-import. Aspects of our approach have similarities with other data cleaning research: ontological reasoning for data cleaning [1], duplicate detection [9], entity resolution [7], and declarative cleaning operators [4].

The contributions of this paper include the following:

1. A definition of deep data cleaning based on ontologies that include real-world crisp and fuzzy constraints.
2. Post-import data cleaning for repositories that necessarily include erroneous, incomplete, and inconsistent data.
3. A prototype implementation in the context of a large application, including:
   (a) instance data cleaning in preparation for deep data cleaning;
   (b) conceptual-model-based declaration of ontological constraints to ease the programming of deep data cleaning applications; and
   (c) resolution of issues either automatically or with user-guided assistance.

We provide details of these contributions in Section 2 in which we describe our large-scale application and repository, and in which we discuss our pre-, on-, and post-import deep data cleaning tools. In Section 3 we report on experimental

field tests and on a user study involving our proposed declarative specification. We make concluding remarks in Section 4.

## 2 Deep Data Cleaning

Our large-scale application is about family history. Its repository includes genealogical information along with photos, stories, and memories about ancestors. Discovering, analyzing, and organizing family histories is necessarily investigative, requiring researchers to gather and organize clues from census records, obituaries, parish records, and many other sources. Source information is not always accurate, is sometimes unreadable, and is often non-existent. Researchers inevitably draw some conclusions that are incomplete and inaccurate. Automatic extraction only exacerbates the problem. Of necessity, deep data cleaning is required both to clean data for import into the repository and to clean data already residing in the repository.

### 2.1 Application System

FamilySearch [3] hosts *Family Tree*, a wiki-like repository of genealogical information that allows world-wide collaboration on a single, shared, ancestry of humankind. *Family Tree* contains over 1.2 billion person records. Among other collections of interest to family historians, FamilySearch also hosts historical records containing 6.2 billion searchable names. Most of these historical records, typically hand-written, have been indexed for search by more than a million registered volunteers. Advances in automatic indexing (equivalent to automated information extraction) are being recognized as a possible way (perhaps the only way) to keep up with indexing the vast number of documents.

Among other work on automated information extraction, our Fe6 project [2], which includes the data cleaning tools we present here, is aimed at automatically extracting genealogical information from FamilySearch's growing collection of over 360,000 family history books that have been scanned and OCR'd. Figure 1 is one among the many millions of pages of these books. Targets for the Fe6 extracted information are (1) the historical records collection of indexed documents and (2) *Family Tree*, but for the tree only if the generated information is manually checked and entered.

Once entered into these repositories, the information becomes searchable. Figure 3 shows a search for the person record of Abigail Huntington Lathrop in *Family Tree*. A click on Abigail's name takes the user to a page containing all her family information along with all the sources that document the information. Figure 4 shows a snippet of one of these sources which was automatically generated as a result of extraction and cleaning by the Fe6 system. Figure 5 shows the result of running a similar query for Abigail in the historical records repository. The human-indexed information for the image is at the bottom of the figure.

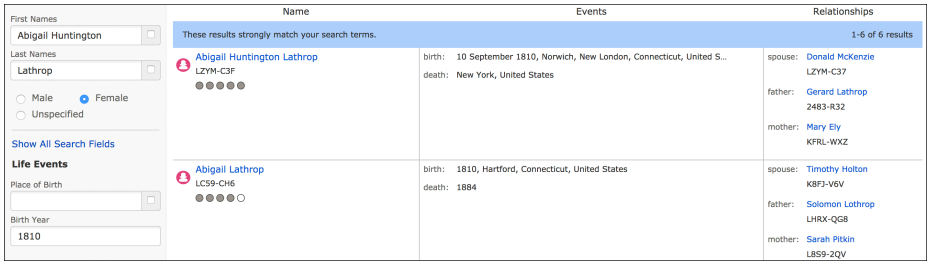
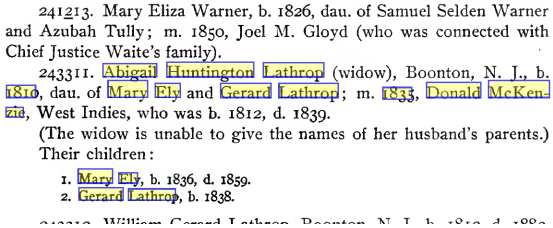



**Fig. 3.** Query for Abigail Huntington Lathrop in *Family Tree*.



241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1833, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.)

Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

**Fig. 4.** Highlighted Source for Abigail Huntington Lathrop in *Family Tree*.

## 2.2 Pre-Import Data Cleaning

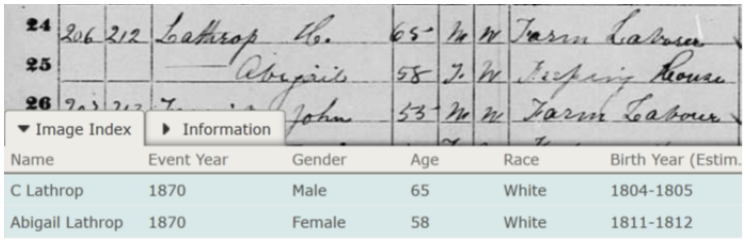
**Fe6** stands for **F**orms-based **e**nsemble of extraction tools with **6** pipeline phases. It is the pre-import pipeline that extracts information from the pages of a book and prepares it for import into FamilySearch repositories. The six pipeline phases include: (1) prepare book data for processing, (2) extract data with an ensemble of extraction engines, (3) merge, clean, and semantically check extracted data, (4) provide for human checking and correction within a forms motif, (5) enhance data by standardizing values and inferring additional information, and (6) generate documents for import. Both shallow and deep data cleaning occur in Phases 3 and 4 with some follow-on data cleaning in Phase 5. For completeness, we briefly mention shallow data cleaning, but explicate deep data cleaning and its connection to ontological conceptual models as the focus of this paper.

In Phase (2) Fe6 tools perform several types of shallow data cleaning:

- merge data extracted by the ensemble of extraction engines based on extracted text being located at the same source page coordinates;
- resolve end-of-line hyphens (e.g. "Donald McKen- zie" in Figure 1 becomes "Donald McKenzie");
- check name forms and fix when feasible (e.g. "William Gerard Lathrop, Boonton" in Figure 1, which is an improper name form extracted by one of the extraction engines, is transformed to "William Gerard Lathrop");
- check for OCR errors and fix when feasible (e.g. In Figure 1, Theodore's birth year, "i860" is transformed to "1860");
- parse dates, converting them to Julian dates for ease of computation.

| Name | Events | Relationships | View |
|---|---|---|---|
| Abigail Huntington Lathrop<br>Connecticut Births and Christenings, 1649-1906 | birth: 10 September 1810 NORWICH TWP, NEW LONDON, CONNETICUT | father: Gerard Lathrop<br>mother: Mary | |
| Abigail Lathrop<br>United States Census, 1870 | birth: from 1811 to 1812 New York<br>residence: 1870 New York, United States | other: C Lathrop | |

**Fig. 5.** Results of a Query for Abigail Huntington Lathrop in FamilySearch Records.



▼ Image Index ▶ Information

| Name | Event Year | Gender | Age | Race | Birth Year (Estim. |
|---|---|---|---|---|---|
| C Lathrop | 1870 | Male | 65 | White | 1804-1805 |
| Abigail Lathrop | 1870 | Female | 58 | White | 1811-1812 |

**Fig. 6.** Abigail Huntington Lathrop in Census Record with Indexed Information.

Given this cleaned data stored in the ontological model in Figure 7, our *Constraint Enforcer* tool performs deep data cleaning. Formally, each object set in the conceptual model is a one-place predicate (e.g. $Person(x)$ and $BirthDate(x)$ in Figure 7) and each $n$-ary relationship set is an $n$-place predicate (e.g. $Person(x_1)$ *was born on* $BirthDate(x_2)$ and $Person(x_1)$ *married* $Spouse(x_2)$ *on* $MarriageDate(x_3)$ *at* $MarriagePlace(x_4)$). Thus, we are able to specify deep data cleaning detection rules as Datalog-like inference rules. Added to the predicates that come from the target conceptual model, we have predicates whose instance values can be computed from the populated predicates in the conceptual model (e.g. $Age(x)$, computed from date differences). To obtain the probability distributions for fuzzy constraints, we define discrete functional predicates (e.g. a distribution over the age at which people die). We populate these distributions by sampling the vast store of data in FamilySearch's *Family Tree*.

As an example, the following rule gives the likelihood of a child being born to a couple a number of years after (or before) the couple's marriage.

$$
\begin{aligned}
&Child(x_1)\ is\ a\ child\ of\ Person(x_2)\\
&Child(x_1)\ was\ born\ on\ BirthDate(x_3)\\
&Person(x_2)\ married\ Spouse(x_4)\ on\ MarriageDate(x_5)\ at\ MarriagePlace(x_6)\\
&Years(x_7) = YearOf(x_3) - YearOf(x_5)\\
&child\ being\ born\ Years(x_7)\ after\ marriage\ date\ of\ parent\ Person(x_2)\ has\ Probability(x_8)\\
&\Rightarrow\\
&Child(x_1)\ being\ born\ Years(x_7)\ after\ marriage\ date\ of\ parent\ Person(x_2)\ has\ Probability(x_8).
\end{aligned}
$$

When Constraint Enforcer applies this rule to the information extracted from Figure 1, we see, as one example, that Mary Eliza Warner and Joel M. Gloyd are unlikely to be the parents of Mary Ely and Gerard Lathrop, who were respectively born 14 and 12 years *before* their presumed parents were married.

When a constraint violation is detected, we can always report it, and we can sometimes automatically resolve it. Reporting takes place in Phase 4, as Figure 8

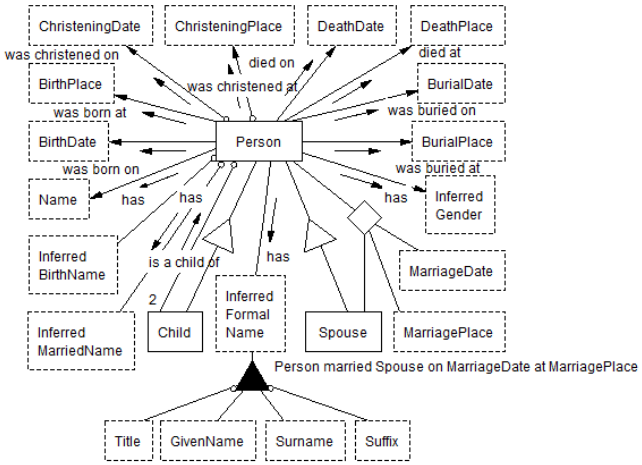



**Fig. 7.** Target Ontology Expressed as a Conceptual Model.

shows. A user can edit extraction results using COMET [2], our **C**lick **O**nly, or at least **M**ostly, **E**xtraction **T**ool. COMET lets users fill in fields of form records on the left by clicking on text in a page on the right. (In Phase 1, we align the OCR'd text with the image text so that it is superimposed over the OCR'd text.) The Fe6 extraction engines prepopulate records with data, and Constraint Enforcer adds warning icons for each record field for which it finds a constraint violation. When a user hovers over a record, COMET highlights its fields and the corresponding text in the page and also displays warning icons, if any. When a user clicks on a warning icon, COMET pops up an explanation window. The middle explanatory note of the three warning messages in Figure 8 corresponds to the rule above in which the non-lexical object set values are replaced by names of persons and lexical object set values are parenthetically added to their object set name. Thus, for example, in the third antecedent statement in the rule above, the non-lexical object sets *Person* and *Spouse* in Figure 7 are replaced respectively by "Mary Eliza Warner" and "Joel M. Gloyd", and the lexical object sets *MarriageDate* and *MarriagePlace* have parenthetically appended to them "1850" and the special null designator "unknown" respectively.

When an implication rule is violated, one or more of the antecedent predicate assertions must be incorrect—indeed, for our application, one of the antecedent assertions obtained from the Fe6 extraction engines. Except for the case of only one such assertion, the task of automatically determining which one(s) are in error is non-trivial. The top explanatory note in Figure 8, however, does not come from an implication rule, but rather from the participation constraint in Figure 7 stating that a child has exactly two parents. Because family history books group families together, it is reasonable to assume that children belong to the closest set of parents when extracted parents either all precede or follow the

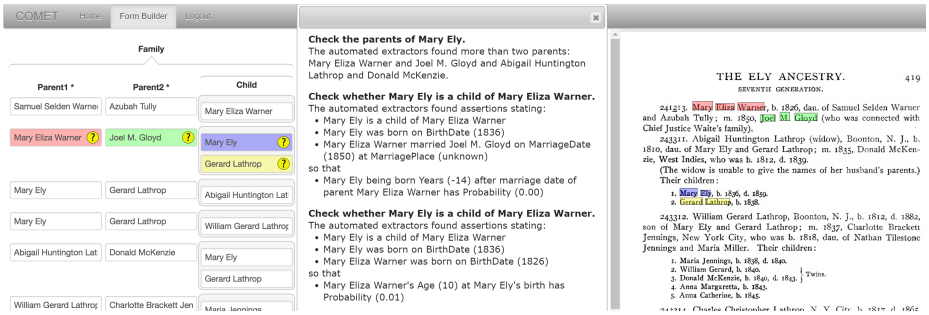



**Fig. 8.** Messages Resulting from a Click on the Mary Ely Yellow Warning Icon.

children. Thus, we are able to automatically reject the assertions linking Mary and Gerard to Mary Eliza Warner and Joel M. Gloyd.

Following Phase 4, Fe6 tools perform some additional shallow data cleaning:

- standardize dates (e.g. "Nov. 4, 1898" becomes "4 November 1898");
- standardize names (e.g. when names are extracted as last-name-first, the name components are reordered);
- infer additional information, making *Person-Spouse* relationships symmetric and filling in object and relationship instances in Figure 7 for *InferredGender*, *InferredBirthName* (e.g. "Maria Jennings Lathrop" in addition to her extracted name "Maria Jennings" in Figure 1), *InferredMarriedName* (e.g. "Abigail Huntington Lathrop McKenzie"), and *InferredFormalName* (e.g. "Mrs. Mary Augusta Andruss Lathrop") along with name component parts (e.g. *Title*: "Mrs.", *GivenName*: "Mary" and "Augusta", and *Surname*: "Andruss" and "Lathrop").

In Phase 6, Fe6 generates for import into FamilySearch's historical records repository a GedcomX [5] file for the page being processed that contains all the object and relationship instances stored in the conceptual model in Figure 7. It also adds to the GedcomX file the bounding box information obtained in Phase 1 for each extracted text element. In Phase 6, Fe6 also generates for import into *Family Tree* two files for each person: (1) a file containing the information in the person's instance graph stored in the conceptual model in Figure 7 and (2) a PDF file containing a marked-up image of the page, like the one in Figure 4, which highlights the person's extracted information.

### 2.3 Post-Import Data Cleaning

We program post-import deep data cleaning with a conceptual-model-equivalent language [8].[3] Figure 9 shows an example. The first three statements indicate

[3] We could have programmed pre-import deep data cleaning with the same language, but the language itself was developed as we worked on programming the Fe6 pipeline. Henceforth, pre-import deep data cleaning is being programmed with this language.

how the conceptual model in Figure 7 is specified. Each statement declares a relationship set in the model in which its related object sets are capitalized nouns. Colons specify is-a hierarchies; if an object set name resolves (possibly through transitivity) to *String*, it is *lexical* (e.g. *Birthdate*:*String*) and is otherwise *nonlexical*. Participation constraints for a statement's relationship set declaration are in square brackets (e.g. the "[2]" in Figure 9 specifies that *Child*, which is a specialization of *Person*, participates in the relationship with exactly *2 Person*s, who are a child's parents). General constraints for the model are specified in *ENSURE* statements. The *ENSURE* statement in Figure 9 is one of the ontological constraints defined for the conceptual model's representation of the ontology. Its name is "I am not my own ancestor" with which it can be referenced. Its body is a Datalog-like statement, defining the constraint. Note the use of relationship set names in general constraints.

```
Person[1] was born on Birthdate:String[1:*];
Child:Person[2] is a child of Person[0:*];
Person[0:*] married Spouse[1:*] on MarriageDate:String[1:*]
    at MarriagePlace:String[1:*];
...
ENSURE I am not my own ancestor BEGIN
    IF Child(c) is a child of Person(p) THEN
        Person(p) is an ancestor of Descendant:Person(c);
    IF Person(p) is an ancestor of Descendant(d) AND
       Child(c) is a child of Person(d) THEN
           Person(p) is an ancestor of Descendant(c);
    IF Person(p) is an ancestor of Descendant(p) THEN
       Person(p) is an ancestor of Descendant(p) has Probability(prob)
           WHERE prob = PROBABILITY OF PersonIsOwnAncestor(p);
END;
```

**Fig. 9.** Conceptual-Model-Equivalent Language

The language is fully declarative, which allows a user to easily change both the model and the constraints. Thus, it is easily configurable to accommodate additional concepts such as an occupation that might be of interest or to accommodate a host of GedcomX *FactType*s (e.g Adoption, BarMitzva, ...). More importantly for ontological deep data cleaning, it provides for ease of adding, deleting, and modifying constraints.

On import into *Family Tree*, the first check is for potential duplicates. Given potential duplicates $x$ and $y$, the check loads the instance graphs of $x$ and $y$ into the conceptual model defined in Figure 9. (Instance graphs are hypergraphs since relationships are not always binary.) It then conflates the *Person* object identifiers for $x$ and $y$ and runs the model's constraints against the resulting instance graph.

As an example, consider merging the instance graphs obtained from Figure 1 for Mary Ely (the first numbered child on the page) and Mary Ely (the mother of Abigail Huntington Lathrop). As *Person*-IDs, let Mary Ely (the child) be $P_1$, Mary Ely (Abigail's mother) be $P_2$, and Abigail be $P_3$. Conflating $P_1$ and $P_2$ (as $P_{1\cdot2}$), yields a new graph with many relationships. Figure 10 shows the graph reduced to just those edges connected to $P_{1\cdot2}$ plus the edges connecting person objects to names. Among the many relationships are ($P_3$:Person, $P_{1\cdot2}$:Person) since Abigail is a child of Mary Ely (her mother) and ($P_{1\cdot2}$:Person, $P_3$:Person) since Mary Ely (the child) is a child of Abigail. We now have a cycle which the *ENSURE* constraint in Figure 9 will detect and thus will state that the probability of such an occurrence is zero—i.e. "I am not my own ancestor" is false, a clear violation, and so the proposed merge should be rejected.

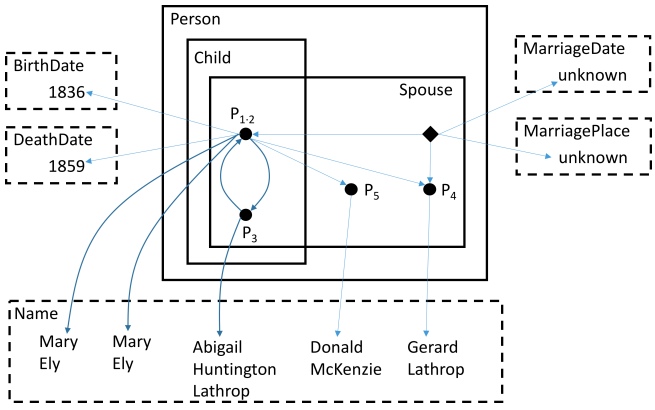



**Fig. 10.** Instance Graph with Mary Elys Conflated.

Beyond import, even when the input data is clean and even when duplicates are correctly detected and merged, there is no guarantee that the data in *Family Tree* is clean. (Over 4.3 million users have contributed to the creation of the tree, each in his or her own way based on available data or even just on memories of family lore.) To aid in cleaning *Family Tree*, we have implemented a tool, called *Tree Sweeper*, which runs over the ancestry of a given person in *Family Tree*. It imports into the conceptual model in Figure 9 the instance graphs of ancestors in the tree up to a specified generation. Then, given the ontological constraints of the model, it detects and reports encountered problems. As Figure 2 shows, a Tree Sweeper run over the ancestry of Abigail Huntington Lathrop in Figure 1 raised a red flag about Sara Sterling, Abigail's great-grandmother. Tree Sweeper can also detect fictitious persons in *Family Tree*, most likely created by improper merges of presumed duplicates. An analysis of the person's instance graph is often sufficient to detect bogus persons.

## 3 Experimental Evaluation

We have conducted several field tests of Constraint Enforcer and of Tree Sweeper, and we have conducted a user study to assess how well people with varying computer expertise can modify *ENSURE* rules.

### 3.1 Field Tests

*Constraint Enforcer Error-Detection.* Early in our work on deep data cleaning, we wanted to know how well and with what coverage Constraint Enforcer identifies errors made by the Fe6 extraction engines. We selected three pages of three different books, ran the extraction engines against them, and without using the Constraint Enforcer identified all semantic extraction errors. We then wrote and tested constraint rules to catch all these errors. Thereafter, we applied Constraint Enforcer to extraction results from four other randomly chosen pages from each of the three books. The extraction engines filled 1201 fields in 479 records. The Constraint Enforcer marked 239 of these filled-in fields as potentially erroneous. Hand-checking, we found that 12 of these fields were erroneously marked and that Constraint Enforcer failed to mark 25 fields filled in with erroneous data—thus yielding accuracy scores of 94.1% precision and 90.5% recall.[4] As a result of this field test, we realized that coding, modifying, and even discarding constraint rules would be the main impediment to increasing accuracy. This observation provided impetus for making constraint rules easy to specify, resulting in the development of the latest version of our model-equivalent programming language.

*Error Detection: Tree Sweeper vs. FamilySearch.* To evaluate the efficacy of Tree Sweeper, we compared its error detection capability with FamilySearch's, which runs as an error reporting background process in *Family Tree*. We randomly chose four persons in *Family Tree* and collected the records of all their ancestors up to the eighth generation, yielding a sample size of 423 persons. Table 1 shows the results of running Tree Sweeper over these 423 person records and the errors reported by FamilySearch within these records. Tree Sweeper found 17 more crisp errors (31% more) than did FamilySearch. In addition, Tree Sweeper found 17 probabilistically unlikely (fuzzy) errors. FamilySearch does not consider fuzzy errors. Thus, for example, a 12-year-old mother is considered impossible, but a 13-year-old mother is not an error. Tree Sweeper uses a distribution when evaluating a mother's age at the birth of her first child. It establishes an unlikely threshold and an error threshold, both of which are percentages. The Tree Sweeper error threshold for a mother's age at the birth of her first child was 0.01% and the unlikely threshold was 1%. This capability is enabled by the use of distributions in Tree Sweeper to measure the probability that a constraint has been violated rather than using a single crisp error threshold.

*Merge Problem Detection.* Wanting to determine the extent to which merge problems occur in *Family Tree*, we sampled 51 people randomly from the tree and found that 9.8% were formed by merging. Next, we sampled 140 people

[4] See [14] for additional details of this field test.

**Table 1.** Error Detection: Tree Sweeper vs. FamilySearch.

| | **Crisp** | **Fuzzy** |
|---|---|---|
| **Number of people in sample** | 423 | 423 |
| **Number of errors found** | 72 | 17 |
| **Errors per person** | 16.9% | 4.0% |
| | | |
| **Errors Found & Percent** | | |
| **Tree Sweeper** | 68, 94.4% | 17, 100% |
| **FamilySearch** | 55, 76.4% | 0, 0.0% |
| | | |
| **Errors found by Tree Sweeper but not by FamilySearch** | 17 | 17 |
| **Errors found by FamilySearch but not by Tree Sweeper** | 4 | 0 |
| **Errors found by both** | 51 | 0 |

with reported errors and found that 76% of them had been formed by merging. There is a significant difference in the merge percentages, implying that there is a correlation between people with errors and whether they are merged individuals. This observation, however, does not suggest causality. To investigate further we evaluated the 18 individuals who had three or more reported errors. Of these 18, 15 were merged individuals. Further evaluation of those 15 showed that 6 of the merges were erroneous. Thus, a significant number of errors are caused by improper merging, but there are certainly other factors to be considered.

### 3.2 User Study: Declarative vs. Imperative Specification

In our Error Detection field test in Section 3.1, we found that FamilySearch and Tree Sweeper each had one false positive. In Tree Sweeper's case, it reported that a person was probably too old to get married. The problem occurred because of our misinterpretation of the distribution we were using. We thought it represented the probability of a person marrying at a given age. Instead, it was the probability of a person's age when married for the first time.

The underlying conceptual model of Tree Sweeper, including general constraints, was designed to be easy to write, understand, and modify. To evaluate this claim, we performed an experiment involving the solution of our misunderstanding of Tree Sweeper's marrying age distribution. We conducted an experiment with 31 subjects. Three of the participants were experts in reading, writing, and modifying the text-based conceptual modeling language; the others had never seen the language.

In the experiment each subject was given a one-page tutorial on the language. Examples were included. Next subjects were shown how to modify the syntax of an unrelated constraint. The subjects were then given a second page describing the problem and the erroneous marrying-age constraint. After read-

ing the description, they started a timer and began modifying the constraint to solve the problem. When they thought they were finished, they presented their solution to the proctor who indicated whether their solution was correct or not. If incorrect, they were to modify their solution and resubmit. This submission process repeated until they had correctly modified the constraint. We used two metrics: time to correctly modify the constraint and the number of tries it took to be successful. Table 2 shows the results.

**Table 2.** Declarative vs. Imperative Coding.

| **Group (Nr. Subjects)** | **Avg. Time** | **Avg. Tries** | **Max Time** | **Max Tries** |
|---|---|---|---|---|
| Experts (3) | 1.0m | 1.0 | 1m | 1 |
| Computer Scientists (12) | 1.8m | 1.3 | 3m | 2 |
| Non-Computer Scientists (6) | 3.2m | 1.3 | 5m | 2 |
| Uncategorized Subjects (9) | 2.3m | 1.6 | 5m | 3 |
| Overall | 2.1m | 1.3 | 5m | 3 |

At an $\alpha$-level of 0.05, we determined that experts were faster than computer scientists, and computer scientists were faster than non-computer scientists. Even the worst subject took only five minutes and three tries. We have no timing data for this specific modification in FamilySearch. However, based on experiments run to determine modification time of imperatively written code [13], we conjecture that the modification time for this error would take much longer. Moreover, Tree Sweeper's deployment time is comparatively orders of magnitude faster. We need only copy the modified text file containing the model to the deployment site. This takes just a few minutes. In contrast, FamilySearch must re-compile, link, and deploy in a subsequent release.

## 4 Concluding Remarks

Ontological deep data cleaning relies on a plethora of declaratively specified crisp and fuzzy constraints. It is particularly useful for investigatory applications in which the data in the application's repository is never complete, consistent, and error free and in which data cleaning is needed both pre- and post-data import.

Our prototype implementation within a genealogical information-extraction application is declarative—based on a conceptual-model-equivalent programming language. Field tests bespeak its usefulness in discovering and providing guidance for fixing discovered errors. A user study indicates that a broad spectrum of users can quickly learn the language and effectively modify its constraint rules.

In future work, (1) we intend to enhance and deploy Tree Sweeper as an app to aid users in their quest to clean their ancestry in *Family Tree*. (2) After additional work in which we need to code the temporary conflation of merge-proposed individuals and test and enhance the accuracy of proposed ontological

constraints, we plan to offer it as a background sanity check for the merge operation in FamilySearch. (3) Since the Fe6 extraction engines are capable of reading documents [2], an interesting future possibility is to have them read source documents like those in Figures 4 and 6 and determine whether the evidence supports the conclusions posted in *Family Tree*.

## References

1. Dolby, J., Fan, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Ma, L., Murdock, W., Srinivas, K., Welty, C.: Scalable cleanup of information extraction data using ontologies. In: Proceedings of the 6th International Semantic Web Conference, pp. 100–113. Busan, Korea (2007)
2. Embley, D., Liddle, S., Lonsdale, D., Woodfield, S.: Ontological document reading: An experience report. Enterprise Modelling and Information Systems Architectures: International Journal of Conceptual Modeling pp. 133–181 (February 2018)
3. FamilySearch. http://familysearch.org
4. Galhardas, H.: Data cleaning and transformation using the AJAX framework. In: Lämmel, R., Saraiva, J., Visser, J. (eds.) Generative and Transformational Techniques in Software Engineering, pp. 327–343. Springer, Berlin, Germany (2006)
5. Gedcom X. http://www.gedcomx.org/
6. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2), 199–220 (1993)
7. Kang, H., Getoor, L., Shneiderman, B., Bilgic, M., Licamele, L.: Interactive entity resolution in relational data: A visual analytic tool and its evaluation. IEEE Transactions on Visualization and Computer Graphics **14**(5) (2008)
8. Liddle, S., Embley, D., Woodfield, S.: Unifying modeling and programming through an active, object-oriented, model-equivalent programming language. In: Proceedings of the Fourteenth International Conference on Object-Oriented and Entity-Relationship Modeling (OOER'95). pp. 55–64. Gold Coast, Queensland, Australia (December 1995)
9. Low, W., Lee, M., Ling, T.: A knowledge-based approach for duplicate elimination in data cleaning. Information Systems **26**(8), 585–606 (December 2001)
10. Rahm, E., Do, H.: Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin **23**(4), 3–13 (2000)
11. Raman, V., Hellerstein, J.: Potter's Wheel: An interactive data cleaning system. In: Proceedings of the 27th International Conference on Very Large Data Bases. pp. 381–390. VLDB'01, Rome, Italy (September 2001)
12. Vanderpoel, G.: The Ely Ancestry: Lineage of RICHARD ELY of Plymouth, England. The Calumet Press, New York, New York (1902)
13. Woodfield, S.: An experiment on unit increase in problem complexity. IEEE Transactions on Software Engineering **SE-5**(2), 76–79 (March 1979)
14. Woodfield, S., Lonsdale, D., Liddle, S., Kim, T., Embley, D., Almquist, C.: Pragmatic quality assessment for automatically extracted data. In: Proceedings of ER 2016. pp. 212–220. Gifu, Japan (November 2016)