Pattern Markup Language: A Pattern-Based Tool for Quickly Automating Genealogy Data Extraction

Jonathan Baker, Hilton Campbell,
Jordan Crabtree, and David W. Embley

Many websites contain genealogical data that is meaningful only to humans. How can we easily extract this data and make it available so that users can intelligently query genealogical pages in a multitude of different formats on thousands of different web sites? Most web pages have structures that are easily described by common patterns, in which case data extraction may be automated. We present a language—Pattern Markup Language, or PatML—and a set of tools that allow a user to specify a set of regular expressions that are subsequently used to extract and store meaningful data for machine use. We have successfully used PatML to automate the extraction of genealogical data from several web sites in different formats. Extracted data is available to query within the framework of Genesis [www.dftproject.org]—an open source tool for querying and combining online genealogical information.

1 Introduction

One of the biggest obstacles in genealogy research today is that information, scattered across a multitude of unrelated web pages, cannot be uniformly queried and combined. Data on the same family, or even the same individual, may be spread across and duplicated on hundreds of different websites and dozens of incompatible formats. To give a rough indication of the breadth of information available, Cyndi's List, the premiere directory of genealogical websites, lists over a quarter of a million verified and categorized links [www.cyndislist.com]. Likewise, WeRelate.org, a specialized search engine for Internet genealogy, boasts an index of over six million web pages from more than 1.3 million unique sources [www.werelate.org].

At least one program, PAF Insight [www.ohanasoftware.com], tries to solve this by making it easy to search for, compare, and combine information about known individuals. But this approach is limited because it can only be used on one database (the International Genealogical Index) and only for individuals already in the user's personal database. Another solution in development is Genesis, a program with a GUI similar to a browser, but with code running in the background that identifies genealogy pages, transforms them into a common format, and then searches for and virtually merges identical individuals [Cam06].

Unfortunately, the problem of transforming pages from a variety of different formats to one common format currently requires a crowd of hand-coded transformation programs. To alleviate this difficulty we present the Pattern Markup Language (PatML). PatML simplifies the process by enabling users to quickly write programs at a higher level to extract data from any regular page layout.

PatML lets users specify the regular pattern of information on a page. Given this description, PatML automatically generates a transformation program that will extract the information from all pages conforming to the pattern. To facilitate pattern specification, PatML provides a GUI that shows a schema of the target information values and lets users specify regular expressions for each. A regular expression is a concise description of finite state automata. For example, the regular expression *a*ba* describes a string of characters commencing with any number of *a*'s followed by *ba*. The GUI lets users color-code regular expressions and simultaneously shows what it recognizes both as a rendered HTML page and as source text. This enables users to see which information will be extracted.

2 Pattern Markup Language (PatML)

In order to use PatML, the user must create or otherwise acquire a simplified XML schema to specify what information can be found in pages of a particular type. Figure 1 shows a completed schema for genealogical web pages. On a page with genealogy, we often find information about people with several attributes including name, gender, birth, and death. Each of these attributes may have additional attributes such as date or location.

Once a schema is obtained, the user creates a PatML file that conforms to the schema. This specifies a hierarchy of regular expressions that identify values of interest in a data source. There is one XML schema for a given type of information (people, cars, countries), but a different PatML file with instructions for extracting data from each format.

We developed two interfaces to simplify this process: one to generate the schema and another to create PatML files.

- The *Schema Generator* is a graphical user interface that helps the user to define what kinds of information are of interest for a specific domain.

- The *PatML Generator* allows the programmer to select which pieces of information from the schema can actually be found in a given page type, as well as regular expressions that identify where that information can be found.

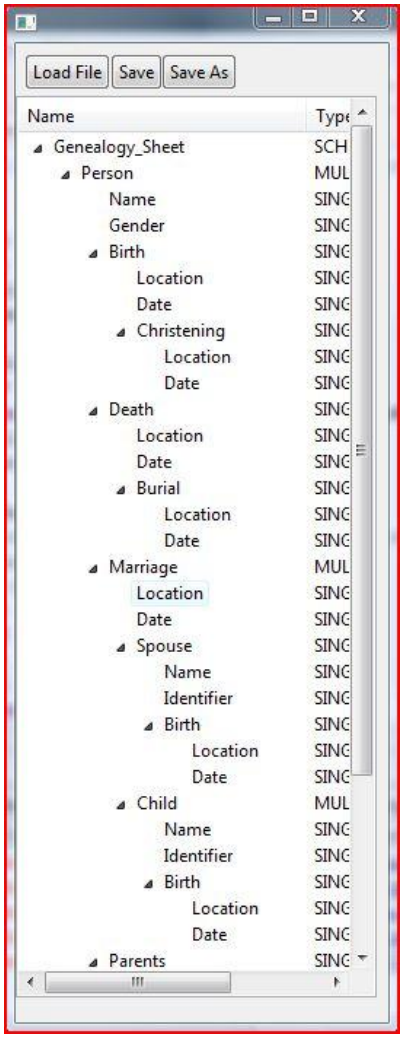

Figure 1: Screenshot of the Schema Generator.

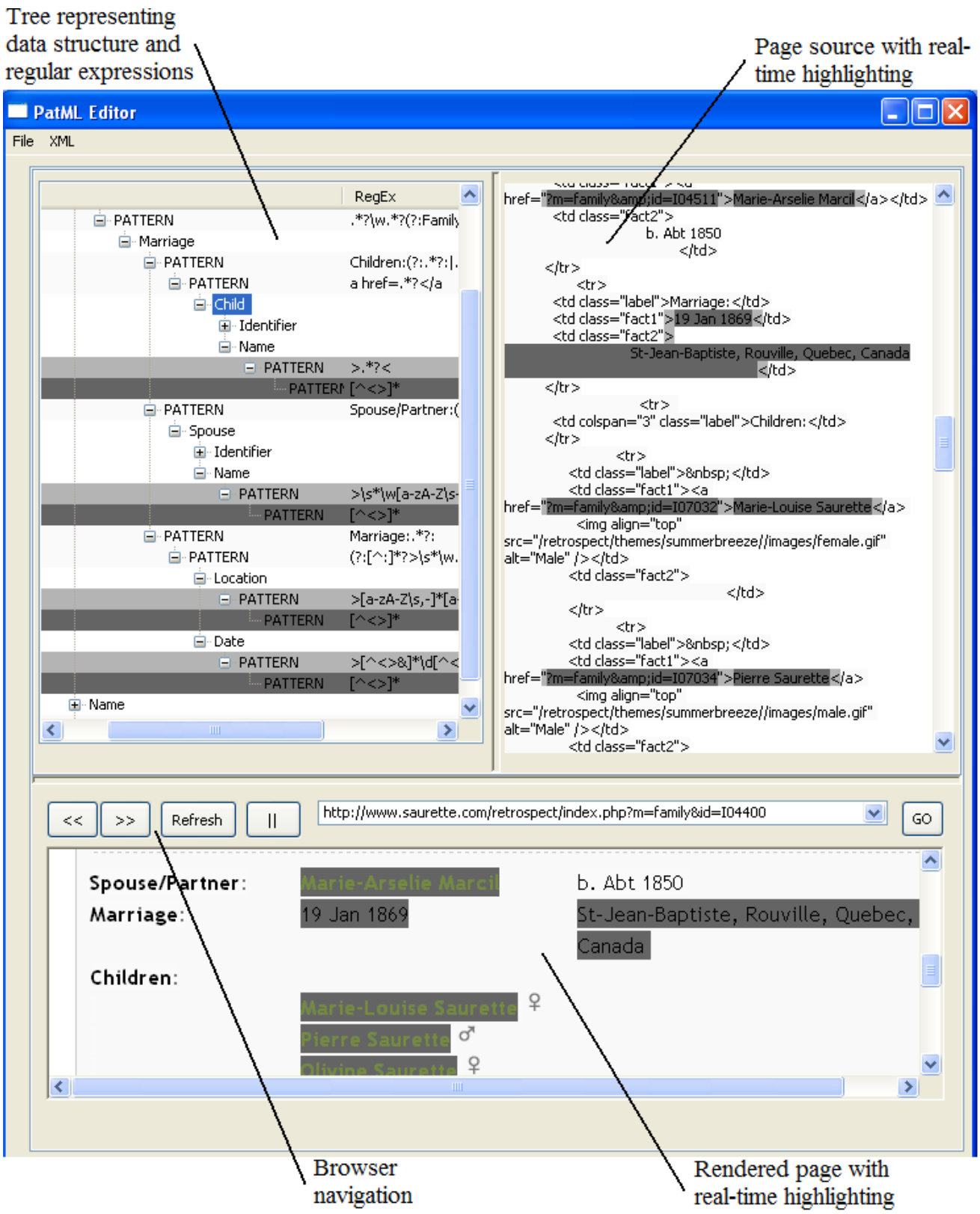

Figure 2: Screenshot of the PatML Generator.

2.1 Schema Creation

To help generate a schema of information to locate, we developed a GUI that allows a user to quickly establish the hierarchical relationships of the data being sought.[1] The user adds and renames nodes representing the types of data. They may also be deleted or dragged to a different position in the tree.

The nodes seen as children in the tree will be attributes of that structure: a Birth node, for example, would have nodes under it such as a Date and Location. The tree representing the entire structure is saved as a simplified XML schema file that may be read by the PatML generator.

2.2 Pattern Specification

Once a schema is established, a set of regular expressions is required for each group of similarly arranged pages to enable data extraction. Figure 2 shows the GUI that the *PatML Generator* uses. The user loads a genealogy web page for a site that needs to be extracted in the browser at the bottom. The source of this page appears in the panel at the top right.

After the user loads the XML schema file created with the *Schema Generator*, a boilerplate tree is created in the panel at the top left. The user then adds to this tree each element that will be found on pages of this site. The user adds regular expressions to specify where each element can be found in the source of the page. The software aids the user by highlighting matches to the currently selected regular expression as it is typed in both the source text and on the rendered page. As regular expressions are adjusted, other pages of the same type should be loaded in the browser to confirm that they are correctly handled.

When all expressions are satisfactory, the current page may be viewed as an XML. This will occasionally reveal additional errors in the regular expressions that must be fixed. The XML generated here is what Genesis will use to display data when a page of this type is loaded. The tree representing the data may be saved at any point as a PatML file representing the elements and regular expressions specified so far.

2.3 Data Extraction

When the PatML tree is complete, it is used to parse the source for the indicated data. First, the root node receives the entire document. The root and each other node follow similar procedures:

1) The node receives text from its parent node.

2) The node passes text to its children.

    a) If it is a pattern node, it searches for its pattern in the received text and then passes the matching part(s) to each of its children.

        i) If its pattern matches nothing, it means this piece of information is not found on this

[1] Note that this tool may be used for hierarchical data other than genealogy. For example, to search for car repair information, one might generate a schema specifying that an owner has a name, area of residence, and several vehicles each having a make, model, year, and past problems.

particular page. The node returns an empty text in step 3.

b) If it is a node representing a type of data (such as a name), it passes the entire text it received to each of its children.

3) If the node has no children, it will pass its match back *up* the tree to its parent.

a) When a node receives the match from its children, it returns these matches with XML tags surrounding the information to its own parent.

i) The data as returned to the root may then be previewed. When PatML is incorporated in a larger project, this output may be redirected to another format.

3 Experimental Evaluation

As part of our work with Genesis, we began by attempting to hand-code site-specific parsers for *The Next Generation of Genealogy Sitebuilding* (TNG) and *Retrospect GDS* sites. The work of hand-coding these parsers was labor-intensive and somewhat difficult to debug. A single site could require approximately 14-24 hours to find a general solution. Using PatML a regular expression tree for these sites was written in approximately 2-3 hours. Highlighted copies of source and rendered HTML aided effectively in writing correct regular expressions.

Using our solution, Genesis is now able to access data from sites running either TNG or *Retrospect GDS* software. The *User Site Index* for TNG features 146 sites and claims that "thousands more exist" [Lyt08]. The featured sites alone contain approximately half a million individual records. We also used PatML to extract data from FamilySearch.org, which contains over one billion names [Fam08].

4 Conclusions and Future Work

The tool set arising from our research is a useful abstraction for producing machine-readable output from previously unreadable sites.

PatML is limited to websites with regular structures, so our tool cannot be considered a general solution for the entire web. However, our approach appears to be effective for an extensive volume of web pages and can be utilized by applications requiring semantic understanding of a specific class of web pages. From our own experience, using the tool is much more efficient for extracting data from regular sites than writing site-specific parsers. The tool may also be used for data structures other than genealogy to simplify data extraction for other data models.

In the future we would like to generate PatML descriptions for additional websites to expand the scope of genealogy searches. We would also like to pursue further automation of the process so that pattern descriptions can be generated from examples instead of requiring the user to hand-craft regular expressions.

5 Bibliography

[Cam06] H. Campbell. Enabling the Distributed Family Tree. Master's Thesis Proposal, Brigham Young University, Provo, Utah, November 2006.

[Lyt08] D. Lythgoe. The Next Generation of Genealogy Sitebuilding, January 2008. (lythgoes.net/genealogy/software.php).

[Fam08] FamilySearch. Wikipedia, January 2008. (en.wikipedia.org/wiki/FamilySearch).