Formalizing and Querying Heterogeneous Documents with Tables

Krishnaprasad Thirunarayan and Trivikram Immaneni

Department of Computer Science and Engineering
Wright State University, Dayton, OH 45435, USA.
t.k.prasad@wright.edu, immaneni.2@wright.edu
http://www.cs.wright.edu/~tkprasad

Abstract. This paper explores the application and extension of XML technology for formalizing and querying heterogeneous documents containing text and tables. Specifically, it analyzes the pros and the cons of embedding the formalization into the original document to obtain an XML Master document that improves traceability while enabling making explicit interpretation of tables separately, in a modular fashion. This work is relevant for formalizing, representing, and manipulating (domain-specific) content in legacy semi-structured documents, and for authoring new documents with tables that should be simultaneously human readable and machine processable.

1 Introduction and Motivation

XML has been used to annotate documents with metadata in the realm of document processing and content extraction, to be read and maintained by humans. XML has also been widely used as a standard text-based format for information exchange/serialization in the context of Web services, for manipulation by machines. These two prevailing applications of XML have been termed *document-centric* and *data-centric* respectively. This paper explores an approach to unifying these two views by using XML elements to materialize abstract syntax, and together with XML attributes, to represent the semantics via annotation, obtaining an XML Master document that is both human sensible and machine processable. This work can be beneficial for formalizing, representing, and manipulating (domain-specific) content in legacy semi-structured documents, and for authoring new documents that are simultaneously human and machine consumable.

In order to better situate the current work, consider the relationship between information extraction and document authoring for the Semantic Web, in terms of client-server paradigm. Information Extraction deals with automatic filtering of legacy documents and filling-in a pre-specified, domain-specific template by a client/end user. In contrast, Semantic Web requires authoring of documents conforming to a fixed ontology by a server/document creator [1, 3, 11]. This paper pursues a pragmatic, semi-automatic approach to annotation that straddles

these two extremes. The ultimate goal is to develop the document and its formalization hand in hand, and keep them side by side, to improve traceability (that is, maintain implicit link between original document fragments and its formalization).

The documents of interest are heterogeneous and semi-structured, containing text and tables. For example, consider the following table type that appears frequently in materials and process specs, which gives tensile strenth and yield strength as a function of the thickness of a specimen:

Thickness (in)	Tensile Strength (ksi)	Yield Strength @0.2% offset (ksi)
0.5 and under	165	155
0.50 - 1.00	160	150
1.00 - 1.50	155	145
...	...	...

Table 1. Tensile Strength and Yield Strength Table in Spec

Handling tables requires recognizing table layout and understanding table content, for subsequent manipulation. The table topology alone cannot be used to understand tables automatically because the semantics of a table is normally gleaned by a user from the heading labels and captions that relate various columns and rows. A potential semi-automatic approach to dealing with tables is to develop a catalog of table types and its processing via XML and XSLT stylesheets, and then make explicit the interpretation of each table instance occuring in a document manually, using specific XML-based annotations. The composite XML document not only provides a human sensible view, but also supports machine manipulation. This approach holds promise in so far as the number of table instances far outnumbers the number of table types because the manual effort required to annotate a table instance is relatively less compared to formalizing table processing that respects table's semantics.

In Section 2, we discuss several interesting issues and techniques related to table handling. Specifically, we situate, analyze, and review our past work on formalizing and querying tables in Water [13]. In Section 3, we present details on formalizing tables in XML and using XSLT stylesheets for table manipulation, and discuss obstacles, advantages and disadvantages. In Section 4, we conclude with a summary of remaining problems to be solved and suggestions for future work.

2 Related Work

There are a number of practical, orthogonal research issues pertinent to handling of tables found in documents as discussed below:

Extraction of Tabular Data: Tables in documents available in plain text, MS Word, or PDF form may be hand-formatted or created using table primitives. The techniques we have developed in the past can be used to convert MS Word document into plain text that delimits and preserves table layout, so that it is human readable [12]. The work on table extraction of Pinto *et al* [6] deals with the isolation of complex tables from the rest of the text, and identifying the title, row headings, cell boundaries, etc. Similarly, earlier work on Wrapper Induction and its robust generalization to accommodate visual cues implicit in the geometry of the tables can assist in table extraction from HTML documents [4] [2]. In the realm of content extraction from materials and process specs, we also need to deal with complex column headings. Similarly, there are systems that address issues such as the recognition of table components in a text document (e.g., TINTIN [8], [5],[15]) or the representation of structure and flexible presentation of tables (e.g., Tabula-Magica [9]). For the current state of the art, it seems reasonable to expect the extractor to manually identify and tag the table components, and focus on how to interpret the domain-specific table content.

Representation of Tabular Data for Semi-automatic Translation : There are several different issues to be considered regarding the semantics of tabular data:

- Consider an XML-inspired approach to providing semantics to tables in plain text that promotes traceability, where a table contains both the headings and the data. The precise relationships among the various values in a row/column are tacit in the heading labels, and obvious to extractors. However, this semantics needs to be made explicit to do any machine processing. But storing only a semantics rich translation in a new formalism is not always conducive to human comprehension or flexible. So the representation language should have the provision to more or less preserve the grid layout of a table to promote readability and enable changes to the original table to be easily incorporated in text, while describing the interpretation of each row/column in a way that is flexible and applicable to all rows/columns for further machine manipulation. We have looked into two different avenues, each with its own pros and cons [13]: In Water [7], annotation definition can encapsulate interpretation and be treated as a method, while the annotated data can be viewed as a method call. This novel view of annotation enables the interpretation of data to be described in an additive fashion, shared among multiple annotated table instances of the same kind. Unfortunately, this Water program is not a well-formed XML document, thereby losing the ability to reuse techniques and tools developed for manipulating XML documents. Furthermore, Water is not conducive to convenient embedding of the formalization into the original document, because Water requires the original text to be delimited and incorporated as comments. On the other hand, a well-formed XML annotated table intersperses tags with table data which is not always desirable considering the effort required to create it and the form that results.

- Another approach worthy of exploration is to define a language of table expressions with compositional semantics that enables one to build and manipulate tables with headings algebraically along the lines of Wolfram[14].
- At this juncture, a viable approach to dealing with tabular information is to develop a catalog of predefined tables and map the tabular data into a set of pre-defined tables, possibly qualified. For instance, a complex table can be built as a union of qualified simple tables. Overall, manual mapping of complex tables into simpler regular structures have the following benefits:
  - It provides semantics to data, thereby removing any lurking ambiguities.
  - It provides natural expression of data for traceability and ease of use.
  - It enables automatic manipulation, that is, querying and translation.

Manipulation of Tabular Data : Once the problem of table representation is solved, we need to develop the corresponding language and techniques for querying, combining and detecting conflicts among related tables.

3 XML/XSLT-based Approach to Tables

Recall that heterogeneous, semi-structured text documents are not conducive to machine processing. So it makes sense to develop techniques to abstract, formalize, and represent their content in a more structured manner. In order to ascertain the soundness of the formalization/translation, it is important to link the original document fragments with their formalization. The additional data structures needed to capture this association can be simplified if the formalization can in fact be *embedded* in the original document. Furthermore, the composite document has potential to be readily understood and updated by a human user due to its resemblance to the original document.

The document-centric and data-centric views of XML seems to provide a means to the desired end:

- XML can encode text and tabular data, to make explicit abstract syntax and the semantics via annotations, and
- XSLT stylesheets can be used to describe various interpretations respecting the semantics for formal manipulation, in a modular fashion.

Relationships described in plain text can be formalized using XML elements and XML attributes. However, it is much harder to deal with tables as discussed below.

Water, an XML-inspired programming language, provides a rich substrate for formalizing and querying heterogeneous documents [13]. The annotated data can be *interpreted* as a method call, and the XML-element as a method. However, the correspondence between formal parameters and actual arguments is positional, yielding an ill-formed XML document in the presence of tables. Specifically, this

approach does not permit flexible embedding of annotations into text, or use of XML techniques and tools (such as XSLT).

We will now attempt to annotate a document containing the text of a table, to capture its semantics via suitably chosen XML tags and XSLT stylesheets that manipulate the table according to its semantics. Any deviation from XML well-formedness criteria will be remedied by *reinterpreting* the resulting document in terms of an "equivalent" XML document. Specifically, we will reinterpret positional association of actual arguments to formal parameters using fixed name-based associations that can be captured in XML using attribute-value pairs and manipulated using XSLT stylesheets. Once this association is clarified, the annotated data can in fact be interpreted differently by programming-in different interpretations for the XML-element using different XSLT stylesheets. For instance, one can recover just the text sans the annotations, verify integrity constraints, transform data or even facilitate data querying (such as by mapping the annotated document into Prolog-like syntax). Concretely, the requirement that `temperature` must be a number (static type), or should be in the range from 300°F to 500°F (dynamic constraint) can be made explicit by defining `temperature` constraints via XSLT stylesheets.

We illustrate the XML-based representational issues and XSLT-based transformational details using the following illustrative example of a tensile data table taken from materials and process specs.

Thickness (in)	Tensile Strength (ksi)	Yield Strength @0.2% offset (ksi)
0.5 and under	165	155
0.50 - 1.00	160	150
1.00 - 1.50	155	145
...	...	...

Table 2. Input Tensile Data Table

This table can be extracted as text from an MSWord document and subsequently manually annotated to bring out its structure as shown below:

```
<table type="Tensile">
<parameter  name="Yield Offset" value="0.2%"/>
<tableSchema  "Thickness(min)"  "Thickness(max)" "Tensile Strength" "Yield Strength"/>
<tableUnits      "inch"            "inch"           "ksi"             "ksi"   />
  <tableData      "0"              "0.50"           "165"             "155"    />
  <tableData     "0.50"            "1.00"           "160"             "150"    />
  <tableData     "1.00"            "1.50"           "155"             "145"    />
...
</table>
```

In particular, the double quotes and annotations clearly delimit atomic values, relate data and clarify the column headings, the units of measure, and the table parameters which are crucial for proper interpretation of the tensile table data. Unfortunately, the annotated table is not a well-formed XML fragment. To ensure well-formedness, we need to come up with a regular scheme for automatically deriving an equivalent XML document, for example, by introducing

sequencing attributes one, two, three, . . . , etc to capture positional associations via named-associations.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<table type="Tensile">
<parameter  name="Yield Offset" value="0.2%"/>
<tableSchema  one="Thickness(min)" two="Thickness(max)" three="Tensile Strength" four="Yield Strength"/>
  <tableUnits    one="in"               two="in"           three="ksi"              four="ksi"  />
  <tableData     one="0"                two="0.50"         three="165"              four="155"  />
  <tableData     one="0.50"             two="1.00"         three="160"              four="150"  />
  <tableData     one="1.00"             two="1.50"         three="155"              four="145"  />
...
</table>
```

This *in-place* formalization of a table instance, when augmented with reusable, table type specific XSLT stylesheets yields a representation that exhibits a prescribed semantics and is both human accessible and machine manipulable. In particular, XSLT stylesheets can be designed to carry out the following operations on the XML document that contains both the original text table and the annotated table:

- Query: to perform table look-ups.
- Transform: to change units of measure such as from standard SI units (International System of units) to FPS units (Foot, Pound and Second system of units) and vice versa.
- Format: to display the table in HTML form.
- Extract: to recover the original table in text form.
- Verify: to check static semantic constraints on table data values.

We now present two example XSLT stylesheets (that can be skipped by a casual reader without any loss of continuity) and the generated results for illustrative purposes, enabling better appreciation of the programming difficulties and the pay-offs:

- Given a thickness, determine the tensile strength.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="text"/>

<xsl:template match="text()"/>

<xsl:template name="calculateTensile">
<xsl:param name="thick" select="0.65"/>
<xsl:if test="tableSchema[@one='Thickness(min)' and
                          @two='Thickness(max)' and @three='Tensile Strength']">
    <xsl:if test="tableUnits[@one='inch' and @two='inch']">
<xsl:for-each select="tableData">
     <xsl:if test="@one &lt; $thick and $thick &lt;= @two">
  <xsl:value-of select="@three"/>
      </xsl:if>
</xsl:for-each>
    </xsl:if>
</xsl:if>
</xsl:template>

<xsl:template match="table[@type='Tensile']">
<xsl:call-template  name="calculateTensile">
<xsl:with-param name="thick" select="0.25"/>
```

```
</xsl:call-template>
</xsl:template>

</xsl:stylesheet>
```

The stylesheet ignores text data, determines the appropriate tensile table formalization in XML, and then searches through this table to determine the applicable tensile strength value. For the example tensile table in XML, for the thickness value of 0.25 inch, the looked up tensile strength value is 165 ksi (kilo-pounds per square inch).

– Given the table in FPS units, create an HTML table that displays the data in both FPS units and SI units.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
   <xsl:template match="/document">
    <html>
      <body>
<h2><xsl:value-of select="table/@type"/></h2>
<xsl:variable name="pval"><xsl:value-of select="table/parameter/attribute::value"/></xsl:variable>
<xsl:variable name="pname"><xsl:value-of select="table/parameter/attribute::name"/></xsl:variable>
<table border="1">
<tr bgcolor="#9acd32">
   <xsl:for-each select="table/tableSchema/attribute::*">
      <xsl:choose>
<xsl:when test="name(.) = 'four'">
<th>
<xsl:value-of select="concat(.,' @',$pval,' ',$pname)"/>
</th>
</xsl:when>
<xsl:otherwise>
<th>
<xsl:value-of select="."/>
</th>
</xsl:otherwise>
       </xsl:choose>
   </xsl:for-each>
</tr>
<tr bgcolor="#9acd32">
   <xsl:for-each select="table/tableUnits/attribute::*">
<xsl:if test=".='inch'">
<th><xsl:value-of select="concat(.,' (','mm',')')"/></th>
</xsl:if>
<xsl:if test=".='ksi'">
<th><xsl:value-of select="concat(.,' (','Mpa',')')"/></th>
</xsl:if>
   </xsl:for-each>
</tr>
<xsl:for-each select="table/tableData">
   <tr>
       <xsl:for-each select="attribute::*">
<xsl:variable name="a">
<xsl:value-of select="."/>
</xsl:variable>
<xsl:choose>
<xsl:when test="name(.)='one' or name(.)='two'">
<th>
  <xsl:value-of select="concat(.,' (',format-number($a*25.4,'#.#'),')')"/>
</th>
</xsl:when>
<xsl:otherwise>
<th>
  <xsl:value-of select="concat(.,' (',format-number($a*6.895,'#.#'),')')"/>
```

```
</th>
</xsl:otherwise>
</xsl:choose>
       </xsl:for-each>
   </tr>
</xsl:for-each>
</table>
       </body>
     </html>
    </xsl:template>
</xsl:stylesheet>
```

For the example tensile table in XML that has thickness values in inch and strength values in ksi, we generate HTML table for presentation that shows thickness values in both inch and mm (millimeter), and strength values in both ksi and MPa (MegaPascal or Newton per square millimeter) as shown below.

Thickness (min)	Thickness (max)	Tensile Strength	Yield Strength @0.2% Yield Offset
inch(mm)	inch(mm)	ksi(MPa)	ksi(MPa)
0(0)	0.5(12.7)	165(1137.6)	155(1068.7)
0.50(12.7)	1.00(25.4)	160(1103.2)	150(1034.2)
1.00(25.4)	1.50(38.1)	155(1068.7)	145(999.7)

Table 3. Transformed Tensile Data Table for Presentation

To ensure practicality of this approach, we need a tool (1) with MS FrontPage like interface where the Master document is the annotated form, the user explicitly interacts with/edits only a view of the annotated document, for readability reasons, and (2) with support for *export as XML* option which can turn the annotated document into a well-formed XML document. For instance, in the current context, this means adding attributes such as one, two, three, ..., etc using a transformation shown below that can only be carried out outside of XML/XSLT:

```
    <elem     "P1"      "P2"          "P3"   ...>
==>
    <elem   one="P1"   two="P2"   three="P3" ...>
```

Ideally, we do not need to create a separate annotated table, distinct from what can be obtained by annotating the original document, which further provides the context for interpretation of data and is amenable to track revisions to the embedded tables.

4 Conclusions and Future Work

This paper explored techniques to imbue table instances with machine-processable annotations to obtain an XML Master document, to promote traceability. The

positional association of table data with table headings is captured via semi-automatically created named associations to obtain an XML-compliant document that enables reuse of XML techniques and tools (such as XSLT) for flexible interpretation and manipulation of text documents containing tables.

One can also view this approach as mapping a larger collection of annotated documents into "equivalent" XML documents, or reinterpreting an annotated fragment such as `<elem "P1" "P2" "P3" .../>` as a concise description of sequentially assigned attributes in an XML fragment such as `<elem one="P1" two="P2" three="P3" .../>`.

Encoding tables in Prolog for querying provides flexible alternative to using Water or XML/XSLT. However, for document representation and manipulation, XML/XSLT-based approach is more powerful. Furthermore, treatment of more general tables such as those containing multiple columns with common headings, or containing non-uniform rows combining multiple tables, or containing tables multiple units of measure, etc is non-trivial and complicated even for this approach.

The essence of Semantic Web is to make explicit semantics of data in a machine processable form. What we have accomplished here is a pragmatic first step in the context of tables that enables programming in various interpretations respecting the semantics of an annotated table.

References

1. Antoniou, G., and van Harmelen, F.: *A Semantic Web Primer*, The MIT Press, 2004.
2. Cohen, W. W., Hurst, M., and Jensen, L. S.: A Flexible Learning System for Wrapping Tables and Lists in HTML Documents, In: *The Proceedings of the Eleventh International World Wide Web Conference (WWW 2002)*, pp. 232-241, 2002.
3. Fensel, D., Hendler, J., Lieberman, H., and Wahlster, W. (Eds.): *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, The MIT Press, 2003.
4. Kushmerick, N.: Wrapper induction: Efficiency and expressiveness, In: *Artificial Intelligence*, 118(1-2):15-68, 2000. (Special issue on Intelligent Internet Systems.)
5. Pande, A. K.: *Table Understanding for Information Retrieval*, Master's Thesis, Virginia Polytechnic Institute and State University, 55 pages, 2002.
6. Pinto, D., McCallum, A., Wei, X. and Croft, W.B.: Table Extraction Using Conditional Random Fields, In: *SIGIR '03 Conference*, pp. 235-242, 2003. (`http://www.ciir.edu`)
7. Plusch M.: Water : Simplified Web Services and XML Programming, Wiley Publishing, 2003. (`http://www.waterlanguage.org/`, Retrieved 5/10/2005.)
8. Pyreddy, P. and Croft W. B.: TINTIN: A System for Retrieval in Text Tables, *2nd ACM International Conference on Digital Libraries*, pp. 193-200, 1997.
9. Silberhorn H.: TabulaMagica: An Integrated Approach to Manage Complex Tables, In: *2001 ACM Symposium on Document engineering*, pp. 68-75, 2001
10. `http://www.cs.umd.edu/projects/plus/SHOE/index.html`, Retrieved 5/10/2005.
11. W3C Semantic Web URL: `http://www.semanticweb.org/`, Retrieved 5/10/2005.

12. Thirunarayan K., Berkovich A., and Sokol D.: An Information Extraction Approach to Reorganizing and Summarizing Specifications, In: *Information and Software Technology Journal*, Vol. 47, Issue 4, pp. 215-232, 2005.
13. Thirunarayan, K., On Embedding Machine-Processable Semantics into Documents, In: *IEEE Knowledge and Data Engineering Journal*, Vol. 17, No. 7, July 2005.
14. Wolfram, K: *Compositional Syntax and Semantics of Tables*, SQRL Report No. 15, Dept. of Computing and Software, McMaster University, 62 pages, 2003.
15. Zanibbi R., Blostein D., and Cordy J. R.: *A Survey of Table Recognition: Models, Observations, Transformations, and Inferences*. International Journal of Document Analysis and Recognition, Vol. 7, No. 1, pp. 1-16, Sept. 2004.