Extreme Markup Languages 2004

Montréal, Québec
August 2-6, 2004

Interpretation Beyond Markup

David Dubin
University of Illinois

David Birnbaum
University of Pittsburgh

Abstract

The meaning conveyed by documents and their markup often goes well beyond what can be inferred from the markup alone. It often depends on context, so that to interpret document markup adequately we must situate markup interpretation within a broader interpretive problem. Markup is just one of several sources of evidence brought to bear in processing digital documents; it must be integrated with other information to be exploited fully. An example drawn from an ongoing project on the metrical analysis of Russian poetry helps illustrate these points: the explicit markup of the rhyme scheme can be understood only on the basis of a broader understanding of Russian meter and poetics.

A Conference of IDEAlliance
Extreme Markup 2004 Languages®

Interpretation Beyond Markup

Table of Contents

A Conference of IDEAlliance
Extreme Markup Languages® 2004

Interpretation Beyond Markup

David Dubin and David Birnbaum

§ Introduction

A key limitation in the effective use of document markup is that its *interpretation* often depends on information that is not fully present in the markup itself [raymond96] [cmsmcq00] [renear02:doceng] [cmsmcq02:extreme] [bayerl03] [dubin03:extreme] [dubin03:llc]. A variety of difficulties emerge from this limitation, including:

- that the same markup can convey different meanings in different contexts
- that markup can communicate the same meaning in different ways using very different syntax
- that the same markup can license inferences in multiple semantic domains simultaneously
- that markup may provide evidence of intended meaning that cannot be interpreted fully through inferences based on markup alone

The relationship between syntax and semantics in markup has often been the starting point for discussions of best practice issues, such as the design of schemas, their documentation and annotation, and how markup should or shouldn't be used in a document. The issues noted above have also motivated proposals for how markup and supporting technologies should evolve to meet the needs of applications (see, e.g., [cover00] for a review). However, it is sometimes difficult to disentangle best practice recommendations from strong theoretical claims that seem either to motivate them or to be entailed by them. For example, in a preface to a recent book on RELAX NG [vlist04:RELAXNG], James Clark writes:

> I would argue that the right abstraction is a very simple one. The abstraction is a labelled tree of elements. Each element has an ordered list of children where each child is a Unicode string or an element. An element is labelled with a two-part name consisting of a URI and local part. Each element also has an unordered collection of attributes where each attribute has a two-part name, distinct from the name of the other attributes in the collection, and a value, which is a Unicode string. That is the complete abstraction. The core ideas of XML are this abstraction, the syntax of XML and how the syntax and abstraction correspond. If you understand this, then you understand XML.
>
> In my view, the most important lesson to learn from SGML is not the syntax but the concept of generic markup. Generic markup means describing things in terms of their semantics rather their appearance. Generalizing, the lesson is to keep your data as independent as possible of assumptions about how you are going to process it. The way to do this for XML is to focus on this minimal labelled-tree abstraction. The more you build alternative abstractions on top of that and look at XML instead as a serialization of some other abstraction such as an object or a remote procedure call, the more you are building in assumptions about how components will process XML and the more any rationale for using XML disappears. (pages ix–x)

This quote provides insight into RELAX NG's compelling argument for the proper role of schemas and validating parsers in XML processing. But at the same time, Clark seems to be advancing a claim for the limits of what ought to be represented using XML: description should be in terms of semantics, but if the semantics of an application cannot be represented in Clark's MLTA with full fidelity, then the suitability of XML for that application is called into question. One objection to such a claim is that the MLTA is insufficiently expressive even for the semantics of ordinary document markup [renear02:doceng]. But for the present discussion, we note that Clark's response to problems relating to syntax and semantics is to identify what he feels is the *right abstraction*.

The Semantic Web initiatives represent another kind of response to these same problematic issues. Insofar as conventional markup is incapable of expressing all the distinctive relations among and properties of our domain entities, we can create markup languages (such as RDF and OWL) that are so capable. Once again the focus of the proposals is on models and formalisms: how to equip markup languages with the full power of knowledge representation languages.

As important and as useful as these initiatives are, we believe that in the focus on formalisms, too little attention has been given to the way that people use markup in actual practice. It is our thesis that

Figure 1

```
<LI><PARA>
<HD>Taxpayer identification number needed for each
qualifying person.</HD><para>You must include on line 2
of Form 2441, <ITL>Child and Dependent Care
Expenses,</ITL> or Schedule 2 (Form 1040A), <ITL>Child
and Dependent Care Expenses for Form 1040A Filers,</ITL>
the name and taxpayer identification number (generally
the social security number) of each qualifying person.
See <ITL>Taxpayer identification number</ITL> under
<ITL>Qualifying Person Test,</ITL> later.</para></PARA></LI>
```

IRS publication fragment

addressing semantic problems adequately requires that we situate markup interpretation within a broader interpretive problem. Markup must be understood as one of several sources of evidence brought to bear in processing digital documents correctly. After motivating discussion through some simple examples, we illustrate our approach to the fusion or synthesis of evidence in an ongoing project on the metrical analysis of Russian poetry.

The research described in this paper grew out of two projects. The first was by David J. Birnbaum (Department of Slavic Languages and Literatures, University of Pittsburgh) and Lawrence D. Adams (Department of Computer Science, University of Pittsburgh) on the automated extraction of rhyme schemes from Russian verse [djb97:TT]. The second basis for this work is the BECHAMEL Markup Semantics project [renear02:doceng] which is developing a formal framework for the interpretation of markup, and a multi-layer knowledge representation and inferencing environment (in Prolog) with which to express theories of markup semantics. In earlier papers we have described the development of BECHAMEL at the syntactic [cmsmcq02:extreme] and mapping [dubin03:extreme] levels; in this paper we illustrate modeling at the object layer, and how BECHAMEL can be used in a synthesis of different semantic models.

§ Problems of Interpretation

Figure 1 shows a fragment of markup from an IRS publication [IRS02:P503]. The example shows several instances of a presentational element `ITL` which is used to highlight text in an italic typeface. Italics is used for only a few specific purposes in this publication series, and in the body of the document italics is used only for cross-references and citations to other publications.

One approach to markup interpretation is to limit the inferences to those licensed by the markup alone [cmsmcq00]. Under these criteria, the meaning of italics in these IRS documents is usually a disjunction: either a cross-reference or a citation. However, another rule of style for these publications demands that section names be identified with specific key words before and after the italicized section name (e.g., “See,” “under,” “earlier,” and “later”). The italicized title of another IRS publication will always be preceded by a form or publication number (indeed, this number is part of the citation, although no specific citation element encloses both number and title).

In this example it is possible to distinguish between the two uses of italics by scanning the surrounding text, and for that reason we can adopt a definition of meaning based on the author's understanding of the element's meaning in context, rather than the set of possible meanings allowed by the prose tag set documentation. In practice, of course, it is often much more difficult to find reliable evidence for this kind of disambiguation, particularly if the author is using the element idiosyncratically.

The fragment shown in figure 2 is taken from a scholarly paper [huitfeldt00:nachlass]. In the extract we see an ordered list marked structurally, and references to elements within that list cued using the expected labels and punctuation markup, rather than an ID/IDREF link. It is easy for a human reader (and even fairly easy for a computer program) to recognize that these parenthesized characters are cross references, but such an inference is impossible if text nodes are considered opaque. We believe that examples such as this one are commonplace, though some may come about by the author's choice, and some arise as consequences of the editorial cycle.

Our last example in figure 3 consists of metadata for a conference paper. In interpreting the markup we note first that the syntactic relationships among the elements are to some degree arbitrary. In this case,

Figure 2

```
<p>According to the requirements of the normalised version, one will
not only have to account for the fact that "große" is the correct
spelling of "grosse", but also to sort out the different possible
readings of the text in question. Taken out of context, the example
may seem intuitively to have at least the following possible readings:
<list type="ordered">
  <item n="a">das große weiße Haus</item>
  <item n="b">das große weiße Schloß</item>
  <item n="c">das große Schloß</item>
  <item n="d">das weiße Haus</item>
  <item n="e">das große Haus</item>
  <item n="f">das weiße Schloß</item>
</list></p>
[...]
<p>According to the editorial principles employed at the Wittgenstein
Archives, the example above has exactly two readings: (a) and (b) --
these and no others, neither more nor less (unless interpretational
considerations decide otherwise).
```

scholarly paper fragment

Figure 3

```
<PAPER SECNUMBERS="0"><FRONT
><TITLE>Object Mapping for Markup Semantics</TITLE
><AUTHOR CONTACT="1"
><FNAME>David</FNAME
><SURNAME>Dubin</SURNAME
><ADDRESS
><AFFIL>University of Illinois</AFFIL
><SUBAFFIL>Graduate School of LIS</SUBAFFIL
><ALINE>501 E. Daniel Street</ALINE
><CITY>Champaign</CITY
><STATE>IL</STATE
><POSTCODE>61820</POSTCODE>
```

article metadata fragment

for example, the `AFFIL` and `SUBAFFIL` elements are children of the `ADDRESS` element, but under (for example) the DocBook schema, organization name, division name, and address would all be children of an `AFFILIATION` element. Users of either schema infer the correct substantive relationships without effort. These inferences concern objects in a number of different domains:

1. There are *structural* inferences, such as the fact that the title and author's name are part of the front matter for the article.
2. There are *functional* inferences, such as that the front matter serves to identify the source of the article.
3. There are inferences about *entities in the world* outside the document, such as persons and organizations.

Because authors and markup language designers are able to make easy cognitive shifts from one level or domain to another, markup may exhibit *ontological variation in reference* [renear02:doceng], where something might be tagged as (for example) both a person and a proper name.

We cite these examples to illustrate how in the interpretation of even straightforward document markup, authors, readers, and application programmers rely on knowledge and evidence that goes beyond the markup itself. They bring to the task models of the document at different levels: strings of characters, hierarchies of content objects, graphical objects arranged and aligned in a plane. They bring and use assumptions about the larger world, including other authors and other documents. These models can accommodate varying expressions of the same information, marked up in ways that are equally understandable. As with the interpretation of natural language, omitting crucial markup often presents little problem to a human author or reader; the meaning of a semantically ambiguous tag (or even a missing tag) can be understood using evidence in the text or other contextual cues.

Figure 4

```
<POEM OPID="S1.100" LINESPAN="1-4" COLPAGE="1.261" YEAR="1817"
MASCRHYME="2" FEMRHYME="2" OTHERRHYME="0" MASCUNRHYME="0"
FEMUNRHYME="0" OTHERUNRHYME="0" NOENDWORD="0" COL13="2.75">
<TITLE>Надпись на стене больницы</TITLE>
<LINE LINENO="001">Вот здесь лежит больной студ<STRESS>е</STRESS>нт;</LINE>
<LINE LINENO="002">Его судьба несумол<STRESS>и</STRESS>ма.</LINE>
<LINE LINENO="003">Несите прочь медикам<STRESS>е</STRESS>нт:</LINE>
<LINE LINENO="004">Болезнь любви неизлеч<STRESS>и</STRESS>ма!</LINE>
</POEM>
```

Poem fragment

§ Markup and other evidence

Accounting for the meaning of document markup — even to the degree necessary for correct processing — will typically depend on information that *might* have been explicitly tagged using XML syntax, but which may only be apprehended by means of other evidence. The pursuit of that evidence will require programs to emulate the kind of cross-domain inferences that a human reader would make. For example, determining the reason why a particular text span is italicized may require reasoning based on a model of how the text will look when formatted or how it will sound when read aloud.

We illustrate the application of modeling across semantic domains with examples from an ongoing project to develop a system for extracting rhyme schemes from Russian verse. Our corpus consists of a collection of the poetry of Aleksandr Sergeevich Pushkin (1799–1837). Figure 4 shows an example of one such poem, expressed in one of several encodings that we wish to process. The main output of the analysis is a rhyme scheme (e.g., of the type **AbAb**) indexed to the lines of the poem. An earlier implementation of this analysis method in C is described in [djb97:TT]. One motivation for reimplementing the analysis using BECHAMEL is a desire to synthesize evidence from both markup and content sources.

Russian rhyme is based on pronunciation, rather than orthography, and one needs to convert the written text to a representation of its pronunciation in order to determine which lines rhyme. The markup is crucial because the pronunciation of a Russian words depends on the place of stress not only for the obvious reason (greater expiratory force on the stressed syllable), but also because the pronunciation of a written vowel letter varies according to whether or not it is stressed. The analysis thus requires converting at least the end of each line to a representation of its pronunciation, which is partially dependent on the marked-up final stress.

The algorithm adopted in the earlier version is (in broad outline) as follows. After the place of stress in text is determined:

1. Identify the portion of each line that is checked for rhyme (the STRESS module).
2. Determine the pronunciation of the text susceptible to rhyme (the ORTH module).
3. Determine which lines rhyme perfectly or imperfectly (the WEIGHT module).
4. Identify the rhyme scheme in a stanza (the REPORT module).

Russian rhyme [unbegaun56] [scherr86] [nabokov64] is determined by looking for phonetic matches in two ways:

1. For masculine open rhyme (words that end in a stressed vowel, with no following consonant), perfect rhyme requires that the stressed vowels match, and also that the preceding (“supporting”) consonant match. In English, “see” and “be” rhyme, but they don't in Russian because they lack agreement in the supporting consonants.
2. All other rhyme is like English: the stressed vowels and everything that follows them must match. No supporting consonant is required. This involves all closed rhyme (where a consonant follows the rhyming vowel) and all non-masculine rhyme (where the stressed rhyming vowel is not in the last syllable of the line).

Linguists operating within an anthropological tradition have traditionally de-emphasized orthography under the (not mistaken, but limited) assumption that language is primarily oral, and writing an imperfect representation of speech. The ORTH module rules for mapping from orthography to pronunciation depend in part on traditional notions of phonology (and occasionally morphology), but they are primarily orthographic transformations[1].

1. Russian obstruents are devoiced in final position, so that, for example, a final letter “d” (U+0434) is pronounced like “t” (U+0442).
2. Russian consonant clusters assimilate in voicing to the final member of the cluster. Thus, orthographic “sd” (U+0441, U+0434) is pronounced like phonetic “zd” (U+0437, U+0434), where the “s” (U+0441) becomes voiced because it is followed by voiced “d” (U+0434).
3. A few idiosyncratic consonantal peculiarities must be handled. For example, final “ogo” (U +043E, U+0433, U+043E) and “ego” (U+0435, U+0433, U+043E) are usually represent a particular grammatical ending and are pronounced “ovo” (U+043E, U+0432, U+043E) and “evo” (U+0435, U+0432, U+043E) (subject to the vowel changes described below). Occasionally, however, this sequence does not represent the grammatical ending in question, in which case the “g” letter is pronounced like a “g”, rather than a “v”.
4. All soft vowel letters convert to a sequence of soft sign (U+044C) plus the hard vowel letter counterpart. There are ten vowel letters, five “hard” (a [U+0430], eh [U+044D], y [U+044B], o [U+043E], u [U+0443]) and five soft (ja [U+044F], e [U+0435], i [U+0438], jo [U+0451], ju [U +044E]). The vowels are paired in the order given, so that, for example, “ja” is the soft counterpart of “a”, “i” is the soft counterpart of “y”, etc.
5. Unstressed vowels change as follows:
    1. Unstressed y and u never change.
    2. Unstressed e changes to y.
    3. Unstressed a and o change to y when they follow a soft sign or when they follow ch (U +0447) or shch (U+0449).
    4. Unstressed o changes to a when it does not follow a soft sign or one of the two consonants mentioned immediately above.

Following the transformations of the ORTH module, string identity represents perfect rhyme. The WEIGHT module applies rules that encode exceptions, allowing deviation from perfect rhyme. This information is used by the REPORT module functions to determine the rhyme scheme for the poem. Further details on the C implementation can be found in [djb97:TT]. The present discussion serves to describe the domain entities that participate in the analysis.

§ Modeling objects, properties, and relations

The earlier metrical analysis program combines analysis at the phonemic, morphological, and orthographic levels. At each stage of the analysis character strings not only carry content, but also serve as data structures that encode meaningful correspondences and track the state of the analysis. The current project aims to encode the same knowledge within the framework of BECHAMEL, but in the form of explicit class instances, properties, and relations. In other papers we have described BECHAMEL’s predicates for processing markup at the syntactic level, and for creating object instances via the execution of rules of inference [dubin03:llc]. In the present paper we describe the object classes, properties, and relations necessary to model directly much of what had been implicit in the previous implementation of the metrical analysis algorithm.

As shown in figure 4, the XML markup for the current version of corpus mainly identifies the structure of each poem as a sequence of lines. Accordingly our declarations include a *poem* class, a *stanza* class, and a *line* class, where the lines stand in part/whole relation to the stanza, the stanzas to the poem, and a *follows* relation for line or stanza instances that follow one another in sequence. The poem class takes a *title* property whose value is to be set to the content of the `title` XML element. In modeling the poem and its title and lines, we intend this representation to be abstract, representing neither the written nor the spoken expression of the poem, but simply the poem. The rhyming relation is defined on lines as well as on words, and takes the properties of open vs. closed and masculine vs. feminine as described earlier.

In the orthographic domain, the lines of the poem are written as a sequence of letter tokens, white space tokens, and punctuation marks. Our declarations include a class of *character* objects for representing these tokens. A *follows* relation is defined on pairs of character instances, a *string* class is defined in whole/part relation to the characters, and a *written* relation is for connecting corresponding line and string instances.

As described earlier, the phonetic matching rules use word boundaries (in, for example, the obstruent devoicing rule). Therefore our morphology domain declarations include a class definition for morphological words that participate in a *written* relationship with orthographic word instances.

The phonemic domain is the most complex, in part because it contains the largest number of class and subclass definitions (phonemes, vowels, consonants, obstruents), properties (stressed vs unstressed, voiced vs. voiceless, palatalized vs. non-palatalized) and relations (consonant *supports* vowel, voicing assimilation between consonants, etc.). Moreover, two different type/token distinctions must be supported: phoneme exemplars (/d/ vs /t/, etc.) and the individual instances of the phonemes in the poem. How these are to be supported depends in part on how the phoneme objects are defined: as feature sets (manner and place of articulation, etc.) or as groups of atoms standing in correspondence relations based on shared features. In either case our rules must infer matching sounds on the basis of relations such as assimilation. The phoneme instances themselves must be mapped from the letter objects using the same mechanisms we use to generate content object instances from XML elements and attributes.

§ Example: devoicing and palatalizing rules

The following declarations and sample output demonstrate how our application works across semantic domains. We employ the same mapping rules and blackboard architecture described in [dubin03: extreme], but in this case the mapping rules have one foot in each of two domains at the object level [dubin03:llc], while in our earlier mapping demonstrations, rules mapped from SGML/XML syntactic relations into a single object level domain.

We demonstrate the encoding and results of applying two rules mentioned earlier: devoicing of word-final obstruents on the word муж, and the (non) effect of the soft sign at the end of the word глушь on its adjacent consonant. Mapping the string representations of these words first creates a pair of orthographic word objects, composed of character objects:

```
?- isa(X,orthword),describe_object(X).
o1
Class= orthword
Models: [u0433, u043B, u0443, u0448, u044C]
  part_of: [o2, o1]
  part_of: [o3, o1]
  part_of: [o4, o1]
  part_of: [o5, o1]
  part_of: [o6, o1]

X = o1 ;
o7
Class= orthword
Models: [u043C, u0443, u0436]
  part_of: [o8, o7]
  part_of: [o9, o7]
  part_of: [o10, o7]

X = o7 ;
```

Prior to the execution of the devoicing rule, the mapping rules have instantiated a voiced consonant object standing in a “written” relation with the letter “zhe” at the end of the word муж:

```
?- describe_object(o10).
o10
Class= character
Models: [u0436]
  id = u0436
  name = zhe
  charclass = consonant
  wordfinal = true
  part_of: [o10, o7]
  follows: [o10, o9]
```

```
  written: [o17, o10]

Yes
?- describe_object(o17).
o17
Class= consonant
Models: [x7]
  nonpalatalizable = true
  voicing = voiced
  place = alveopalatal
  manner = fricative
  written: [o17, o10]
```

The devoicing rule states that word-final obstruents that are not already voiceless are to take that value on their voicing property. Membership in the object class *obstruent* is governed by a Boolean expression ("stop OR fricative OR affricate"), and so the rule can be encoded quite straightforwardly:

```
/* devoice word-final obstruents */
mrule :- isa(O, obstruent),
    relation_applies(written,[O,L]),
    property_applies(L,wordfinal,true),
    not(property_applies(O,voicing,voiceless)),
    apply_property(O,voicing,voiceless),!.
```

Note that the property of being word final applies to letters, not to phonemes, and so the devoicing rule uses both phonological and orthographic information: if an obstruent is written with a letter that occurs in word-final position, and that consonant is not already voiceless, then it needs to be. After application of this rule, the value of the voicing property has changed:

```
?- blackboard2.

Yes
?- describe_object(o17).
o17
Class= consonant
Models: [x7]
  nonpalatalizable = true
  place = alveopalatal
  manner = fricative
  voicing = voiceless
  written: [o17, o10]
```

The consonant corresponding to the letter "sha" in the word глушь is already voiceless, and so the devoicing rule will not apply. But that letter is followed by a soft sign that would ordinarily palatalize the adjacent consonant:

```
?- describe_object(o6).
o6
Class= character
Models: [u044C]
  id = u044C
  name = soft
  charclass = mark
  part_of: [o6, o1]
  follows: [o6, o5]

Yes
?- describe_object(o5).
o5
Class= character
Models: [u0448]
  id = u0448
  name = sha
  charclass = consonant
  wordfinal = true
  part_of: [o5, o1]
  follows: [o6, o5]
  follows: [o5, o4]
  written: [o14, o5]
```

However, the rule for the palatalizing effect of the soft sign includes an exception for certain consonants (such as these fricatives) that are never palatalized:

```
/* soft sign palatalizes consonant, unless unpalatalizable */
mrule :- isa(M,mark),
    property_applies(M,name,soft),
    relation_applies(follows,[M,L]),
    relation_applies(written,[C,L]),
    isa(C,consonant),
    not(property_applies(C,palatalized,true)),
    not(property_applies(C,nonpalatalizable,true)),
    apply_property(C,palatalized,true),!.
```

And so the consonant written as “sha” remains unchanged by the action of this rule:

```
?- describe_object(o14).
o14
Class= consonant
Models: [x4]
  nonpalatalizable = true
  voicing = voiceless
  place = alveopalatal
  manner = fricative
  written: [o14, o5]
```

There are many advantages of an efficient data structure, such as the string representation in our C implementation, compared to the more elaborate and redundant scheme emerging from our port of the algorithm to Prolog. The same can be said in general of applications that act directly on markup or on the data structures emerging from a parser, as compared with the knowledge-based applications we develop using BECHAMEL. Disentangling the sounds of the poem, the words, the letters used to write the words, and the abstract structure of the poem itself is a complex process, and offers little in terms of execution speed or scalability to large data sets.

But in addition to identifying the many semantic levels at which metrical analysis takes place, we argue that our current approach promises greater flexibility for extending the project for other analyses, generalizing to languages other than Russian, and the ability to process a wider range of digitally encoded poetry. There are, of course, other approaches to tagging verse using XML. Some, such as the schema used in Wendell Piez's Sonneteer Library, tag the rhyme structure directly [piez03:sonnet]. Others might mark linguistic features only. In all cases both the markup language designer and the encoder exercise considerable freedom in deciding which characteristics to highlight, which to leave implicit, and what form the cues will take.

It is our belief that this is true to at least some extent in most document markup applications. No aspiration to deep semantic content analysis is required in order to find oneself confronted with complex modeling issues. Of course, decisions on how exhaustively to model the domain or domains has to be informed by many issues, efficiency issues included. We just recommend keeping in mind that processing markup appropriately is almost never a matter of the markup alone, regardless of whether the tagging is detailed and expressive or minimal and simplistic.

§ Concluding remarks

We don’t intend to suggest that markup presents any more of an inherent problem than other methods of encoding and representing information. All the distinctions that we’re able to explicate using BECHAMEL could, in principle, guide the re-tagging of documents with richer markup that would eliminate the need for an inferential step in each case. Or BECHAMEL’s network of properties and relations could be serialized in the form of RDF or a topic map.

Our point is simply that having the ability to create RDF expressions (or to follow best practice guidelines in the application of conventional markup) does not in itself contribute to the work of locating a particular document’s position in a vast space of tagging choices. Tasks such as the federation of collections and the migration of document content across systems over time demand robust markup processing in challengingly wide regions of that space.

In developing applications such as the metrical analysis project described above, it’s interesting how quickly we had to look beyond the markup for evidence. BECHAMEL’s mapping layer was designed as a mechanism for bridging overloaded syntactic relationships in markup (parent/child, ID/IDREF, etc.) with the distinctive semantic relationships that they represent. But in the case of our rhyme analysis, the information contributed by markup is only one in an array of domains in which we are trying to automate

reasoning. It seems to us that any markup language for poetry would pose similar problems, unless uses and analyses of the documents were very limited.

Structural markup's many practical benefits must certainly shape the way we understand these challenges, and how we frame solutions to them. One way this may be happening is in attention to markup and to the abstractions and formalisms that form the basis of markup itself—rather than attending to, understanding, and modeling those things people are using markup to encode. There continues to be room for proposals to make markup itself more useful and powerful, and even for debates on what the *right abstraction* behind XML is or should be. But we should also look for better ways that information in markup can be synthesized with other sources of evidence. In evaluating how markup is used in actual practice, we can respond with our views on how the tagging could be better. But it would also pay to ask ourselves how markup — as it is used — contributes to meaning in and interpretation of documents in a broader sense.

Notes

1. These rules for mapping orthography to phonetics are not sufficient for a full transcription of Russian pronunciation, but they are generally adequate for the features and domains that are relevant for the analysis of rhyme.

Acknowledgements

The authors would like to thank Allen Renear, Claus Huitfeldt, the other members of the UIUC Electronic Publishing Research Group, participants in the GSLIS Research Writing Group, and the anonymous reviewers of this paper for their helpful comments and suggestions.

Bibliography

[bayerl03] Bayerl, P. S., Lüngen, H., Goecke, D., Witt, A., and Naber, D. Methods for the semantic analysis of document markup. In *Proceedings of the 2003 ACM symposium on Document engineering* (2003), ACM Press, pp. 161–170.

[cmsmcq00] Sperberg-McQueen, C. M., Huitfeldt, C., and Renear, A. Meaning and interpretation of markup. *Markup Languages: Theory and Practice 2*, 3 (2000), 215–234.

[cmsmcq02:extreme] Sperberg-McQueen, C. M., Dubin, D., Huitfeldt, C., and Renear, A. Drawing inferences on the basis of markup. In *Proceedings of Extreme Markup Languages 2002* (Montreal, Quebec, August 2002), B. T. Usdin and S. R. Newcomb, Eds.

[cover00] Cover, R. The SGML/XML aversion to semantics. Technology report, Cover Pages, 2000. Published on the Worldwide Web at **http://xml.coverpages.org/sgmlEschewsSemantics.html**.

[djb97:TT] Adams, L. D., and Birnbaum, D. J. Perspectives on computer programming for the humanities. *Text Technology 7*, 1 (1997), 1–17.

[dubin03:extreme] Dubin, D. Object mapping for markup semantics. In *Proceedings of Extreme Markup Languages 2003* (Montreal, Quebec, August 2003), B. T. Usdin, Ed.

[dubin03:llc] Dubin, D., Sperberg-McQueen, C. M., Renear, A., and Huitfeldt, C. A logic programming environment for document semantics and inference. *Literary and Linguistic Computing 18*, 2 (2003), 225–233. (This is a corrected version of an article that appeared in 18:1 pp. 39–47).

[huitfeldt00:nachlass] Huitfeldt, C. Editorial principles of Wittgenstein's Nachlass — the Bergen electronic edition. In *Humanities computing: philosophy and digital resources* (London, 2000), G. Pancaldi, H. Short, and D. Buzzetti, Eds., Office for Humanities Communication Publications.

[IRS02:P503] Internal Revenue Service, US Treasury Department. *Child and Dependent Care Expenses*. Washington, DC, 2002. Publication 503.

[nabokov64] Nabokov, V. *Notes on Prosody and Abram Gannibal*. Bollingen Series. Princeton University, Princeton, 1964. page 87. (Originally published by Princeton as part of Nabokov's critical translation of and commentary to Pushkin's "Eugene Onegin.")

[piez03:sonnet] Piez, W. The sonneteer: A demonstration of structured form. Published on the World Wide Web at http://xmlshoestring.com:4442/sonneteer/index.xml, July-August 2003.

[raymond96] Raymond, D. R., Tompa, F. W., and Wood, D. From data representation to data model—Meta-semantic issues in the evolution of SGML. *Computer Standards and Interfaces 18*, 1 (January 1996), 25–36.

[renear02:doceng] Renear, A., Dubin, D., Sperberg-McQueen, C. M., and Huitfeldt, C. Towards a semantics for XML markup. In *Proceedings of the 2002 ACM Symposium on Document Engineering* (McLean, VA, November 2002), R. Furuta, J. I. Maletic, and E. Munson, Eds., Association for Computing Machinery, pp. 119–126.

[scherr86] Scherr, B. P. *Russian Poetry: Meter, Rhythm, and Rhyme.* University of California, Berkeley, 1986. page 193.

[unbegaun56] Unbegaun, B. O. *Russian Versification*. Clarendon, Oxford, 1956. pages 136–37.

[vlist04:RELAXNG] van der Vlist, E. *RELAX NG*. O'Reilly and Associates, Inc., Sebastopol, CA, 2004.

The Authors

David Dubin
University of Illinois, Graduate School of Library and Information Science
501 E. Daniel Street
Champaign
IL
61820
USA
ddubin@uiuc.edu
tel: 217-244-3275
fax: 217-244-3302

David Dubin is a senior research scientist on the staff of the Information Systems Research Lab at the University of Illinois Graduate School of Library and Information Science. He is a member of the Electronic Publishing Research Group.

David Birnbaum
University of Pittsburgh, Department of Slavic Languages and Literatures
1417 Cathedral of Learning
Pittsburgh
PA
15260
USA
djbpitt+@pitt.edu
tel: +1-412-624-5712
fax: +1-412-624-9714

David Birnbaum is Associate Professor and Chairman of the University of Pittsburgh Department of Slavic Languages and Literatures.

Extreme Markup Languages 2004
Montréal, Québec, August 2-6, 2004
This paper was formatted from XML source via XSL by Mulberry Technologies, Inc.