

Extracting Mathematical Expressions From Postscript Documents

Michael Yang and Richard Fateman EECS Department, University of California, Berkeley

January 6, 2003

Abstract

Full-text indexing of documents containing mathematics cannot be considered a complete success unless the mathematics symbolism is extracted and represented in a standardized form permitting searching for formulas, and further use of this information. Most documents produced in the past and digitally encoded, and even most those in current journal production are—at best—encoded in a printer form such as Adobe Postscript [1], in which mathematics is not explicitly marked or easily identifiable. While one might look forward in the future to other document encodings such as XML, the current journal or textbook product is essentially without semantic content: a jumble of odd characters. We demonstrate an approach to decoding, to recognizing and extracting mathematical expressions, from a Postscript document. We plan to produce a syntactic representation of the extracted expressions which can be used to generate various forms. For example, if we can extract \TeX , we can re-typeset the expression. Additionally, starting from this still-ambiguous representation, we can consider combining it with additional contextual processing to assign semantics, including encoding in XML, or computer algebra system languages such as Maple or Mathematica. In this first stage we concentrate on extracting the mathematics parts from text.

1 Introduction

It is common today to see complex documents on the internet posted as Postscript (PS) or Portable Document Format (PDF). Sometimes there is additional information that allows you to immediately extract the ASCII text for that document, and other material including so-called *meta-data* such as title, author, creation date, etc. Sometimes (perhaps most of the time) there is no such data, and short of printing the material on a page and re-scanning it, even determining the sequence of words (for example in a multi-column document) is not trivial. Nevertheless, extracting this information is important for indexing and search. It is possible to approximate the solutions to these meta-data extraction problems for many conventional paper and journal layouts. This is demonstrated very effectively by NEC's ResearchIndex project (formerly CiteSeer, [8]). Consistent with their philosophy (and our belief), the web site provides substantial online links to associated descriptive papers.

An alternative representation used for some newer documents as html or (better) xml goes further toward a structured document in which mathematics can be represented, but the use of this representation for new, much less legacy documents, is not common. Thus the postscript (or merely scanned OCR) equivalent material is where we start.

The material digested by NEC does not make special efforts for understanding mathematical formulas. Either in-line or displayed, they are not understood: if they are extracted at all they appear as jumbled characters in odd sizes and fonts. While a human can consider viewing them as images by reviewing the display form of a page, we believe it would generally be handy to have the formulas decoded into an indexable ASCII encoding of the mathematics suitable for search, re-use, or even insertion into a computation. Other organizations dealing with indexing of mathematics journals (for example JSTOR [6]) face similar unsolved problems with legacy documents, most of which are *not* solved by documents “born digital” yet stored as PS, PDF, or similar appearance-based encodings.

2 Why do we want to view mathematical documents online?

Historically, the research community used conventional paper journal or proceedings publication as a means of propagating findings, establishing priorities of discovery, and (through refereeing), establishing a scholarly reputation for authors. Today it appears that for many purposes and for many scientific communities it is more effective to have papers freely available in the journal/library on-line repository, in an author's individual repository, a non-refereed collection such as ArXiv.org where authors submit preprints, or allow papers to be indexed/copied over into an archive of free papers such as NEC ResearchIndex.

A recent study [7] by Lawrence shows that the mean number of citations to an article that is freely available online is 7.03 as opposed to an offline article which has a mean number of citations of only 2.74, 2.6 times less (aside: in fact, we found these statistics in a paper that was freely available online!). Lawrence's research suggests that the ease in accessing an online document as well as the availability increases the readership of the article. An increase in the effectiveness of "data mining" of scientific articles can provide a useful service to society [2].

The advantages for access, review, cross-reference, and essentially all uses of published research benefit from on-line internet availability [10]. The major downside appears to be the loss of revenue for publishers.

Given this background, any improvement in technology for encoding accuracy stands to benefit all related disciplines. At Berkeley, our digital library research activity could use these encodings to improve the accuracy of the semantic layer in Multivalent Documents (MVD) [11]. Mathematical expressions could be represented in some intermediate form, in some cases \TeX is sufficient, but we prefer a more semantically rich encoding. Options include the Lisp form we produce, or XML, MathML, OpenMath, or a command-string in a computer algebra system language. Interpreters, or MVD "behaviors," can be written to convert the richer intermediate representations to those that are more superficial (say, for printing). Our best current hope for clear semantics is a translation into an unambiguous utterance in a computer algebra system, from which the other forms can be derived. Not all published mathematics can be managed in this way.

3 Improving math access

To increase the ease of access and availability of a paper online, the paper needs to be indexable and searchable. With plain text and html documents, this is easily done by most commercial search engines such as Google or Excite. However, complex documents such as those formatted in Postscript, where the document is essentially stored as a program whose execution on a stack-based interpreter results in a page image, provide much more of a challenge than text. NEC's ResearchIndex bridges much of this gap by processing citations it finds in Postscript/PDF documents and linking papers to each other through these citations, making it easier to find and retrieve related journal and conference papers. It is also of note that Google not only searches through webpages, but is also able to search through postscript and pdf documents on those webpages as well.

In a research context, being able to recognize and process mathematical expressions from a scholarly document can be very important for searching and indexing these documents as they will contain much mathematics. If mathematics were contained on their own math semantics layer in a MVD [11], researchers could cut and paste mathematical expressions into a computer algebra system saving time over hand-typing in the expressions while also avoiding the introduction of errors. Currently, however, ResearchIndex fails to process mathematics in a document correctly.

We illustrate with a simple example. Searching for citations for G. N. Watson's *A Treatise on the Theory of Bessel Functions*, ResearchIndex returns 73 citations. Here is one citation context¹, only slightly edited by inserting newlines.

```
...0.002 0.0025 200 400 600 800 1000 k Figure 3.  
Left: The standard Airy distribution.  
Right: Observed frequencies of core sizes k 2 [20; 1000]  
in 50,000 random maps of size 2,000, showing the bimodal  
character of the distribution. variety of integral or power
```

¹a paper by Cyril Banderier and others, "Random Maps, Coalescing Saddles, Singularity Analysis, and Airy Phenomena," *Random Structures and Algorithms*, 19 3-4, 194-246 (2001)

series representations including (see [1, 45]) 1)

$$\text{Ai}(z) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i(z t + t^3/3)} dt = \frac{1}{\pi 3^{2/3}} \sum_{n=0}^{\infty} (3^{1/3} z)^n \frac{\Gamma((n+1)/3)}{n!} \sin \frac{2(n+1)\pi}{3}$$

Equipped with this definition, we present the main character of the paper, a probability distribution closely related to the Airy function. Definition 1. The standard

Clearly, there are some problems here with the context, and especially the representation of the mathematical expressions². The user is not given very good information from this citation. It would be far more useful and convenient to the reader if this output could mathematical formula in the document in some legible form. system. We see two possible solutions, each first requiring that we locate the place on the page where the recognition process goes haywire:

1. Take a snapshot of the mathematical expressions (perhaps as a gif image file) from the document at that location. Intersperse plain text and the snapshots of the mathematical expressions for context. The solution also works for line art, graphs, half-tone photos, and any other unsolved parts of the page image.
2. Parse the mathematical expressions in the document and return a pre-typeset version in some commonly used format such as \TeX .

The first solution, while easily done, would reduce many mathematics articles to essentially a sequence of images: and none of the math content would be indexed. In fact, given the nature of OCR, it is likely that the short intervening text would not be recognized accurately either. The meta-data of article title, author, and some words in the abstract might be the limit of the automatic indexing.

Therefore we prefer the latter solution when possible: it provides a much more flexible representation where we can apply some semantic analysis to the returned expressions and even pass it to a computer algebra system for processing. *If we cannot make sense of particular formulas, the first approach could always be a fallback. In either case, it is necessary to detect the mathematics in the document first.*

In this particular case, extraction of the document image shows two formulas in the middle of the citation:

$$\begin{aligned} \text{Ai}(z) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i(z t + t^3/3)} dt \\ &= \frac{1}{\pi 3^{2/3}} \sum_{n=0}^{\infty} (3^{1/3} z)^n \frac{\Gamma((n+1)/3)}{n!} \sin \frac{2(n+1)\pi}{3}. \end{aligned}$$

Our goal is to encode these in an internal form that could, if need be, transformed to XML, MathML, or through the intermediary of \TeX shown below, the rendering above.

$$\begin{aligned} \text{\mbox Ai}(z) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{i(z t + t^3/3)} dt \\ \text{\} &= \frac{1}{\pi 3^{2/3}} \sum_{n=0}^{\infty} (3^{1/3} z)^n \\ &\quad \frac{\Gamma((n+1)/3)}{n!} \sin \frac{2(n+1)\pi}{3} \end{aligned}$$

4 What to Do With the Mathematics Once We've Detected It

An obvious target is a typesetting language such as \TeX in which the formulas are treated as the display objects computed by \TeX . Figuring out the exact notation that was used by an author to produce a \TeX formula is not possible in general, since there are many possible distinct utterances for the same output. (That is, \TeX output is ambiguous). But for many formulas a reasonably simple form can be put together. A similar version would be the MathML presentation form, now understood by several browsers.

An important step further would be to find a correct semantic encoding, say an expression in a computer algebra system language like Maple or Mathematica, or an encoding in content MathML.

²there is also a problem with ligatures, since "definition" is rendered as "de nition". The numbers along the top are from a graph. The document seems also to have changed since being indexed, since the reference to Watson's treatise is now citation 50, rather than 45!

5 Detecting Mathematics in Postscript

Our technique is to use as much information as we have about the glyphs, fonts and positions to find mathematics in a postscript document using the heuristics described in Fateman [5]. In a \TeX document, the regular font for typesetting text paragraphs is often Times-Roman while mathematics is typeset in a combination of Computer Modern Roman font (CMR10 below) and Computer Modern Math Italic font (CMMI10 below). Unfortunately, the FontInfo and FontName fields, which provide us with these details, may not survive the processing. They are optional in a Postscript font dictionary. However, at the very least we can detect when a new font is set and, further, we can assign a number to each new unique font that has been defined, and use this information to make a guess at when we are typesetting mathematics.

To give an example of an optimal case, suppose we have the following typeset expression $f(x) = y$ which we will be able to process into the following:

```
f, type: text, font: CMMI10, bbox: (8555.47 6222.22 9111.06 7222.22)
(, type: 4, font: CMR10, bbox: (222.168 -888.889 485.379 333.306)
x, type: text, font: CMMI10, bbox: (555.474 -452.203 1111.11 -0.0271267)
), type: 4, font: CMR10, bbox: (1245.06 -888.889 1555.53 333.333)
=, type: 1, font: CMR10, bbox: (333.36 -444.444 1111.08 -111.111)
y, type: text, font: CMMI10, bbox: (339.084 -453.857 849.392 222.249)
```

Each row begins with the character or string which was detected in the postscript document. Here we have used the fact that \TeX will typeset a function in the following manner to detect math:

1. the function name is in Computer Modern Math Italic
2. the opening and closing parenthesis are typeset in Computer Modern Roman
3. the simple arguments of the function are typeset in Computer Modern Math Italic

In a tougher version of the same information we may have the following:

```
f, type: text, font: 1, bbox: (8555.47 6222.22 9111.06 7222.22)
(, type: 4, font: 2, bbox: (222.168 -888.889 485.379 333.306)
x, type: text, font: 1, bbox: (555.474 -452.203 1111.11 -0.0271267)
), type: 4, font: 2, bbox: (1245.06 -888.889 1555.53 333.333)
=, type: 1, font: 2, bbox: (333.36 -444.444 1111.08 -111.111)
y, type: text, font: 1, bbox: (339.084 -453.857 849.392 222.249)
```

Here, the font names are not present and we will need to make guesses based on the font changes and fontnumbers as to which expressions are mathematics and which are plain text.

6 Implementation

Imagine we have found a document on the internet that is of some interest, perhaps following in the footsteps of NEC. Or that we have found the document *BECAUSE* of NEC's citation, but we wish to index it better.

Our initial processing of the Postscript document is done by a modified version of Prescript [9], a program which modifies operators of the Ghostscript interpreter to output various information about strings and bounding box information about the typeset expressions. Our modified version will report the font name of the current font that is being used when this information is available to us. In addition, we also incorporate an important feature (important to our endeavors in finding mathematics in a Postscript document) which is found in pstotext [3]: the assigning and reporting of unique numbers to each font. This can be later used in typesetting the subject Postscript document.

For example, from the expression $\frac{a+b^c}{de}$, the modified postscript interpreter will return:

```

#S(command :type scale :x 0.009 :y -0.009)
#S(command :type translate :x 8000 :y -80000.0)
#S(command :type moveto :x 0 :y 0)
#S(command :type moveto :x 6401 :y 6783)
#S(command :type setfont :x 1 :y (1134 1000))
#S(box :font CMMI8 :fontnumber (1) :baseline 6782.99 :intlist (97 )
  :y1 6777.81 :x1 6873.02 :y0 6436.77 :x0 6444.42)
#S(command :type setfont :x 2 :y (1106 1000))
#S(box :font CMR8 :fontnumber (2) :baseline 6782.99 :intlist (43 )
  :y1 6888.86 :x1 7579.29 :y0 6222.22 :x0 6953.21)
#S(command :type setfont :x 1)
#S(box :font CMMI8 :fontnumber (1) :baseline 6782.99 :intlist (98 )
  :y1 6777.83 :x1 8000.05 :y0 6172.82 :x0 7666.69)
#S(command :type moveto :x 8036 :y 6470)
#S(command :type setfont :x 3 :y (1230 1000))
#S(box :font CMMI6 :fontnumber (3) :baseline 6470.0 :intlist (99 )
  :y1 6444.42 :x1 8444.42 :y0 6212.32 :x0 8110.43)
#S(command :type moveto :x 6444.44 :y 7000.0)
#S(command :type rlineto :x 2061 :y 0)
#S(command :type rlineto :x 0 :y -45)
#S(command :type rlineto :x -2061 :y 0)
#S(command :type fill :x 6444.44 :y 6955.02)
#S(command :type moveto :x 6971 :y 7764)
#S(command :type setfont :x 1)
#S(box :font CMMI8 :fontnumber (1) :baseline 7764.0 :intlist (100 101 )
  :y1 7777.83 :x1 7889.76 :y0 7112.01 :x0 7000.0)

```

From this output, initial processing will proceed through these steps:

1. **Basic word assembly.** In postscript, strings are sometimes broken up into fragments so they can be typeset better. so we use the same algorithm as `pstotext` and `prescript` to assemble the string fragments into words. We make one modification to their algorithm in that we require that the fonts of the two string fragments begin combined have the same font.
2. **Find types.** Determine the type of all objects (from the heuristics discussed in Fateman [4])
3. pattern match certain groups of commands to find glyphs that are created by multiple commands, for example: square roots, integral signs, and (in our example) horizontal lines

```

#S(command :type moveto :x 6444.44 :y 7000.0)
#S(command :type rlineto :x 2061 :y 0)
#S(command :type rlineto :x 0 :y -45)          ---> horizontal line
#S(command :type rlineto :x -2061 :y 0)
#S(command :type fill :x 6444.44 :y 6955.02)

```

Although our program faces some of the difficulties of optical character recognition (OCR) programs (more properly “document image analysis”), some components are much easier. Consider the task of determining the meaning of horizontal lines; during the initial processing of an OCR program, the horizontal lines could be one of many different things such as a divide bar, a subtraction sign, or, perhaps, part on an equal sign. In a postscript document, however, the latter two are encoded in a string and have been separated out during the preprocess phase. Therefore such tasks as determining the meaning of a horizontal line are easier.

The result of our processing is as follows:

```

a, type: text, font: CMMI8, bbox: (6444.42 6436.77 6873.02 6777.81)
+, type: 1, font: CMR8, bbox: (6953.21 6222.22 7579.29 6888.86)

```

```

b, type: text, font: CMMI8, bbox: (7666.69 6172.82 8000.05 6777.83)
c, type: text, font: CMMI6, bbox: (8110.43 6212.32 8444.42 6444.42)
hline, type: hline, font: nil, bbox: (6354.48 6955.02 8595.399 7000.0)
de, type: text, font: CMMI8, bbox: (7000.0 7112.01 7889.76 7777.83)

```

The processed data is then stable-sorted vertically, then horizontally via the bounding box information of each item. The result is a list of elements that are ordered by their left-most horizontal position, with ties favoring text that is higher up in the page. This aids in the clumping part of the program where we create box structures of the recognized math. The sorted version of the example expression is below:

```

hline, type: hline, font: nil, bbox: (6354.48 6955.02 8595.399 7000.0)
a, type: text, font: CMMI8, bbox: (6444.42 6436.77 6873.02 6777.81)
+, type: 1, font: CMR8, bbox: (6953.21 6222.22 7579.29 6888.86)
de, type: text, font: CMMI8, bbox: (7000.0 7112.01 7889.76 7777.83)
b, type: text, font: CMMI8, bbox: (7666.69 6172.82 8000.05 6777.83)
c, type: text, font: CMMI6, bbox: (8110.43 6212.32 8444.42 6444.42)
1, type: text, font: CMR12, bbox: (25698.9 69111.1 26122.6 70015.0)

```

From the sorted data, we apply the clumping heuristics based on the Math/OCR programs used by Fateman [4] and obtain the following built up expression, a lisp data structure:

```
(vbox (hbox a + (superbox b c)) hline de)
```

One of the problems we had with building up expressions is that some programs that generate postscript files will modify the user coordinate space of the document. We cannot assume that the units in the postscript document in question are the standard point (1/72 inch). This can be remedied by seeing what changes are made to the current transformation matrix in the graphics state of the postscript document. Calls to operators such as `translate`, `rotate`, `scale`, and `concat` need to be tracked and accounted for in the calculations for building up expressions (e.g. how many “units to the right” should I look for the next lexeme).

As a check, the built up expression can be run through a simple recursive printer to generate the final typesetting commands to typeset the expression:

```
$a + b ^ c \over de $
```

which is typeset into: $\frac{a+b^c}{de}$, our original expression!

7 Further Work to be done

The current prototype of the system works for postscript documents generated by \LaTeX and `dvips`. There are many other programs that generate postscript document (for example, Acrobat Distiller or `pdf2ps`) and additional work may be needed so that the system can best process Postscript documents created by these other programs.

The current program also requires that the fontname fields of the fonts used in the postscript documents to be present (as given in the “optimal” case document). Additional heuristics must be devised in the case where fonts are not known.

We intend to benchmark our program on some substantial papers. For example, there are hundreds of thousands of possible test cases available from digital library collections such as the NEC Research Index. We have corresponded with the NEC project scientists, and hope to be able to augment their indexing capabilities.

Interestingly, it appears that NEC is processing new documents that are available in postscript by essentially printing them out (virtually) and re-recognizing them with “OCR” or document understanding systems. The advantage to this is that the OCR programs have been refined over a decade or more to extract layout information (e.g. title, author, paragraph ordering on multi-column pages) in a far more sophisticated fashion than is plausible on “raw” postscript, which may lay out characters on a page in any

order whatsoever, and in particular cannot be trusted for column ordering. The OCR of mathematical formulas is unfortunately not a commercially important task and thus the OCR programs do not solve our problem particularly well. In fact, reducing the problem to OCR of math returns us to a earlier research task, only partially solved. (The decoding of postscript at least gives us 100% of the characters, presuming that we can figure out the font situation.)

8 Conclusions

Searching for some semantic meaning to Postscript math is a challenging, and like the similar task of optical character recognition (OCR), one that we will not solve perfectly³

Even though our problem is more constrained in scope than OCR: we are given the glyphs, positions, and (probable) fonts, there still numerous difficulties that prevent us from truly “understanding” mathematics on a page.

How best to do this encoding? For new documents the role of the author and editor of a mathematical journal article should be enhanced beyond that of mere \TeX . In the future we hope that authors who have become used to typesetting in the “publication” process will go one step further and include at least a start on semantic encoding, and that the publication process will maintain this encoding instead of obscuring it through the production of page proofs and printed pages devoid of their digital origins.

9 Acknowledgments

This research was supported in part by NSF grant CCR-9901933 administered through the Electronics Research Laboratory, University of California, Berkeley. Michael Yang was supported in part by an NSF undergraduate research grant.

10 Appendix: Using the programs

This section gives a quick overview on how to use the Postscript mathematical expression locating/extraction system. We have tested the system using the following programs on a Sun/ Solaris platform:

- Aladdin Ghostscript 6.0
- Allegro Common Lisp 6.0
- \LaTeX Web2C 7.3.1
- `dvips` 5.86

10.1 Preprocess

The processing begins with a Postscript document. For experiments it is convenient to create a document starting with \TeX so the result can be compared with the original document. In the Preprocess step, a sample \TeX document is converted to a `dvi` file via \LaTeX or a related program and then to Postscript via `dvips`.

The Postscript document is run through a modified version of Ghostscript, a common Postscript interpreter available on many different platforms. The modifications to Ghostscript are achieved by simply adding a prolog and epilog to the main Postscript document as it is run through the interpreter. The modifications generate output that includes the bounding boxes, `currentpoints` of each string (which we are taking as the baselines for most text), the font (if it exists), and the string itself, which in our case we change to integer representation of each of the individual characters. This transformation is done to address issues of portability in the next phase. In addition, commands such as `rlineto`, `moveto`, and `stroke` are also outputted for further analysis.

³Of course, it usually doesn't pay to reduce a problem to a harder one: as just mentioned, if the OCR of math were to be perfected, one could reduce our problem to it!

The following command does the initial processing:

```
gs -q -dNODISPLAY -soutfile=output mlypre.ps psfile mlypre2.ps quit.ps
```

where *output* is the destination file and *psfile* is the source Postscript document that needs to be examined.

The output generated is a textfile that constitutes definitions of a series of Lisp lists with structures, `box` and `command`, as its elements. `boxes` are structures for strings that have been found and `commands` are structures for Postscript commands such as `moveto` or `rlneto`. The variable `pages` tells how many pages were processed from the Postscript document. Each list is named `page[N]` where `N` is a integer such that $0 < N \leq \text{pages}$.

10.2 Process

The data from the preprocess phase is then processed by lisp programs `pass1` and `pass2` which implement phase 1 and phase 2 of three phase algorithm described by Fateman [5] respectively. This can be done by loading `util.cl` into Lisp and calling `(process page[N])`.

To view the actual bounding boxes and baselines on the Postscript document in question, `(ps-draw (process page1))` can be run to generate the necessary Postscript commands to draw the boxes. We currently do not have an automated way of displaying the boxes in the page, but the commands can be cut and pasted into the Postscript file directly on the page in question, just before the `showpage` operator (n.b. most of the time, Postscript documents have common Postscript commands aliased at the beginning of the document for efficiency issues. Therefore, you may not see an explicit `showpage` command; for example, `latex` uses `eop`).

10.3 Typesetting

To output \TeX commands from the built up expression generated by `process`, run the lisp program `typeset`. Ideally this would reproduce the typesetting command for the formula, but in general all one can expect is one of the (many) possible forms that will produce the formula or an approximation of it.

References

- [1] Adobe Systems Incorporated. Postscript Language Reference, Third Edition. Addison-Wesley Publishing, 1999.
- [2] Center for Information Technology Research in the Interest of Society (CITRIS) - <http://www.citris.berkeley.edu/>
- [3] Digital Equipment Corporation - <http://www.research.compaq.com/SRC/virtualpaper/pstotext.html>
- [4] Fateman, Richard, Taku Tokuyasu, Benjamin P. Berman, and Nicholas Mitchell "Optical Character Recognition and Parsing of Typeset Mathematics." *Journal of Visual Communication and Image Representation* vol 7 no. 1 (March 1996), 2–15.
- [5] Fateman, Richard. "How to Find Mathematics on a Scanned Page," in Daniel P. Lopresti; Jiangying Zhou (eds). *Proc. SPIE Vol. 3967, Document Recognition and Retrieval VII*, Dec.1999, 98–109.
- [6] JSTOR - <http://www.jstor.org/about/bibliography.html>
- [7] Lawrence, Steve. "Online or Invisible." *Nature*, Volume 411, Number 6837, p. 521, 2001. <http://www.neci.nec.com/lawrence/papers/online-nature01/online-nature01.pdf>
- [8] NEC Research Institute - ResearchIndex <http://www.neci.nec.com/lawrence/researchindex.html>
- [9] Neville-Manning, Craig and Reed, Todd. "A Postscript to Plain Text Converter." June 5, 1996. - <http://www.nzdl.org/html/prescript.html>
- [10] Andrew Odlyzko, "The Future of Scientific Communication," Access to Publicly Financed Research: The Global Research Village III, Amsterdam 2000, P. Wouters and P. Schroeder, eds., NIWI, 2000, pp. 273–278

- [11] Thomas A. Phelps, and Robert Wilensky. "Multivalent Documents: Anywhere, Anytime, Any Type, Every Way User-Improvable Digital Documents" Communications of the ACM, June 2000. - <http://http.cs.berkeley.edu/~phelps/Multivalent/papers/index.html>