CrossTEX: A modern bibliography management tool

Robert Burgess and Emin Gün Sirer
Cornell University, Ithaca, NY
http://www.cs.cornell.edu/People/egs/crosstex

Abstract

CrossTEX is a new bibliography management tool that eases database maintenance, style customization, and citation. It is based on an object-oriented data model that minimizes redundant information in bibliographic databases. It enables a work to be cited not only through arbitrarily-assigned object keys but through semantic information that uniquely identifies the work. It automates common tasks in order to avoid human errors and inconsistencies in bibliographies, while providing users with fine-grained control. CrossTEX's other features include support for modern reference objects such as URLs and patents, direct generation of HTML documents, the ability to write styles in a modern programming language, and extensive databases of published works included in the distribution. It is backwards-compatible with existing BibTEX databases, and, overall, builds on BibTEX's strengths while fundamentally fixing the design restrictions that lead to errors in BibTEX-formatted documents.

1 Introduction

Bibliography management and typesetting play a critical role in publishing. Since its introduction in 1985, BibTEX has become the dominant tool for professional bibliography management with LATEX. It has achieved this dominance due to several good design decisions, such as tight integration with LATEX, a human-readable format for bibliographic databases, and overall ease of use. However, two decades of experience with BibTEX have revealed several fundamental weaknesses that require re-evaluating what features a modern bibliography management system must provide. CrossTEX is such a tool, which learns from the example of BibTEX and provides backwards compatibility while moving forward with new models of bibliographic data and stylistic control.

1.1 What's wrong with BibTEX?

First, BibTEX interprets databases with a single-table *relational* model in which every bibliographic entry contains all of the information that can appear in it. This redundancy is a challenge to those who maintain bibliographic databases because they must correctly look up, enter, and maintain all that information for every reference; it is easy to enter or modify entries separately and use slightly different versions of the name of a journal, conference, or even author.

BibTEX's `crossref` field provides a simple, specialized form of "inheritance", allowing information to be factored out into a single other object, but is not a generalized feature — it is insufficient, for example, for an author who wants to create a bibliography of all of his or her own works, with a common note or URL associate with each entry. Although `@string` objects could help prevent spelling mistakes, there is no practical way around adding the fields explicitly to each entry.

Furthermore, meeting publication requirements in a professional setting that requires consistent abbreviations, names, and formatting guidelines requires users to edit the database. To abbreviate a journal name, the user must edit a copy of the database, find each occurrence of the name, and change it to the desired value. Even if the database takes advantage of `@string` objects, a feature included in BibTEX to attempt to circumvent the restrictive relational model, the values of the strings must be changed, because information appears in a BibTEX-formatted bibliography as it appears in the database. This fact alone prevents large, common databases from being useful save for looking up citation information to copy-and-paste to smaller, per-document bibliographies that authors must manage.

BibTEX citations within documents are based on arbitrary keys attached to each entry in the database. In the best case, authors establish site-wide rules for creating keys from entries so they can correctly guess the required key based on the publication information of the article to be cited. This is still quite fragile, as a single typo may cause the wrong work to be cited; when databases have been assembled by cutting and pasting entries from different sources, the keys are unlikely to follow a convention, and the author must instead look up the appropriate key for each citation.

BIBTEX style files are written in their own, arcane programming language. Few authors or database maintainers have the skill to create or accurately modify a bibliography style to meet requirements handed to them by publishers — they must count on an appropriate style file being provided or already existing, or they must blindly change one that is "close" until it seems to produce the correct typeset appearance.

In addition, BIBTEX does not easily permit the introduction of new kinds of objects. Database maintainers represent newly emerging referenced works, such as URLs, and unsupported objects, such as patents, with the all-purpose `@misc` object whose appearance is difficult to control because it is too general.

Finally, BIBTEX provides very little automation besides formatting the references as specified in the database. For example, it does not check capitalization in titles, ensure that an author's name appears consistently, or abbreviate journals. This can lead to inconsistent spelling, capitalization, or accents for authors' names or titles of papers. A modern tool should be able to automate the tasks of checking and fixing these common errors by enforcing consistency by default.

1.2 A modern bibliography tool

We have developed a new bibliographic management tool, CrossTEX, that addresses these problems. It unifies features from modern programming languages, databases, and bibliography management tools to solve problems authors and database maintainers have struggled with for two decades, and also adds convenient new features while being completely backwards-compatible to allow authors to use old BIBTEX databases unmodified.

Authors and maintainers have very different needs from a tool such as CrossTEX. Authors must conform to a variety of requirements on appearance, fields, and formatting, and so must have great flexibility with their bibliographic data. Database maintainers, on the other hand, should be able to specify everything just once, so they need look up and correctly enter conference locations, author names, and book editors just once, and can find them in one place to update and verify. Separating their jobs is also very important. Authors should need to do very little searching through the database for keys, because their concern is writing. Maintainers should be able to manage databases without concern for document styles, because independence from individual documents allows databases to be shared widely.

CrossTEX is designed not only to enable, but to encourage large, common databases so that authors can get on with their writing while the maintainers have clean databases to manage. As a start, it comes distributed with many large databases of the papers published at major computer science conferences, converted from the DBLP [13] project.

The most important aspect of CrossTEX is that an *object-oriented model* replaces the underlying relational model of BIBTEX. Every entry is an object, which contains fields and ultimately has a value itself; in BIBTEX, entries have fields, and `@string` objects have values, but this notion is not taken to the level of principle. One object can use the value of another by assigning its key to a field, by extension of the syntax for using `@string`s in BIBTEX. In addition, if the containing object leaves any fields unspecified, it will inherit them, if possible, from the other objects it refers to. For example, an object representing a conference could specify not only a value corresponding to the name of the conference, but include fields such as editor or location. Any articles that appear in the conference will simply use that value as a book title and then automatically inherit an editor and location without specifying anything extra. Those very familiar with BIBTEX will note that this is similar to the behavior of the special 'crossref' field, but has become a part of the way data is interpreted across the board rather than a special-purpose feature.

The object-oriented model enables CrossTEX databases to be concise and easily customized. For instance, most authors refer to conferences by their full names in formal journal papers, but abbreviate them otherwise; in CrossTEX, objects are flexible and have both long and short values. For example, "OSDI" and "Symposium on Operating System Design and Implementation" are two very different strings, but they are both names for the same conference. In BIBTEX, the database maintainer would have to choose just one value — but in CrossTEX, both can appear in the same `@conference` object, and the choice can wait until the author chooses stylistic options for each document.

Objects can also specify fields conditionally. Some information depends on context, especially for often-reused objects such as conferences and authors. If the location of a conference changes every year, it can be specified one way for 2006, and another for 2007. Conditional fields are inherited along with their conditions, allowing richly specified objects to adapt to the contexts where they are used. This enables powerful new idioms such as the ability to express everything about a conference over time in a single object.

CrossTEX supports new kinds of objects, including `@url`, `@patent`, and `@rfc`, that help modernize databases and allow them to be more precise, rather than depending on overly permissive, non-specific `@misc` objects. Adding other entirely new objects is easy as well. Due to the more precise taxonomy authors can apply stylistic options with better granularity because each kind of object represents one specific kind of reference.

CrossTEX provides fine-grained control over every aspect of the bibliography's presentation. From choosing between short and long conference names, states or journals to abbreviating author names, capitalizing titles or even pulling out hyperlinks in fields, the author has control with command-line switches. The LATEX files need not change at all and there is no need to write new style files for each combination of options.

CrossTEX provides new ways to keep track of citation keys to make them more meaningful and easier to find and remember. Objects are not restricted to just one key, but can be assigned any number of shorter or more descriptive keys, and even extended with new aliases after their definition. An even more powerful technique inspired by the `NbibTeX` [15] project is constrained citation: An author can cite a paper by Sirer and Walsh in 2006 as `\cite{!sirer-walsh:2006}` without worrying at all about what its key is in the database.

In addition to making consistency of data easy through object inheritance, CrossTEX automatically enforces consistency in other ways to avoid common errors in databases, such as inconsistent capitalization in titles. By default, it processes each title and re-capitalizes them to a consistent style that can be controlled by the author. CrossTEX carefully handles accents in author names, math in titles, and other complications to maintain consistency and a professional appearance even if the database is not perfect.

Maintainers will find many small but potent features for managing large databases, such as the ability to piece together many files with `@include` statements or specify the same field for groups of objects with `@default`. CrossTEX also allows objects to be updated far from their original definition, to permit anyone to correct and extend shared databases entirely without copying and pasting, even if they do not have permission to change the database itself.

CrossTEX is structured as a drop-in replacement for BIBTEX. Simply replace invocations of `bibtex` in the typesetting process with `crosstex`, and then incrementally update your databases to allow more and more consistency and reusability over time using CrossTEX's new features. The rest of this introductory paper will summarize some examples of these features and how they fit together to simplify the typesetting process.

2 Objects, inheritance, and conditions

CrossTEX enables a new idiom for managing conference information that exemplifies the usefulness of its object-oriented model, inheritance, and conditional fields. Information that is always true is specified first; fields specified in square brackets introduce conditions, such as a particular year for which the 'location' or 'month' fields have a particular value. The result is readable and intuitive, but more importantly collects information into a single object that can adapt to its context:

```
@conference{nsdi,
  shortname = "NSDI",
  longname = "Symposium on Networked System
              Design and Implementation",
  [year=2007] address=CambridgeMA, month=apr,
  [year=2006] address=SanJose, month=may,
  [year=2005] address=Boston, month=may,
  [year=2004] address=SF, month=mar,
}
```

The example defines the object `nsdi`, which typically has the value "Symposium on Networked System Design and Implementation" when assigned to a field. Additionally, however, this object is aware that if the referring context includes, for example, the year 2006 and does not already specify an address, the address will be assigned `SanJose`, a reference to a `@location` object representing the city in California. Thus, the papers in the proceedings of that conference need not specify that information at all, but simply pull in all redundant information from `nsdi`:

```
@inproceedings{credence,
  title = "Experience with an Object Reputation
           System for Peer-to-Peer Filesharing",
  author = "Kevin Walsh and Emin {G\"un} Sirer",
  booktitle = nsdi,
  year = 2006,
}
```

The assignment `booktitle = nsdi` is like specifying either `booktitle = "NSDI"` or `booktitle =`

`"Symposium ..."`, depending on whether the author has chosen long or short names to be used for `@conference` objects. Additionally, precisely because the `credence` object does not specify an address or month, those fields will appear exactly as they do in the conference. This concision and consistency make conferences a perfect example of CrossTEX's flexibility from both the perspective of the database and the author.

Because CrossTEX encourages centralized objects with a lot of information, it can be very important to group them into meaningful databases and include them wherever they are needed with the `@include` statement. CrossTEX includes files from a standard path, including such databases as `dates`, `locations`, `conferences-cs` (which contains entries such as the one above for many of the important computer science conferences), `journals-cs`, and so forth. The `standard` database is always included when CrossTEX begins, which allows the administrator to include system-wide databases that must always be available. By default, `standard` includes `dates` (for backwards compatibility with BibTEX and also because dates are so universally useful) and the databases containing the default rules for capitalizing titles.

Databases can also be found in the same directory as the document being processed. Because that directory is searched before the system directories, users can easily supply their own databases, even a `standard` database to control their own defaults. This flexibility allows maintainers and authors to work together to easily find, re-use, and adapt helpful databases.

3 Consistency and automation

Title case is one of the most common inconsistencies when using BibTEX. Often, some papers appear with lower-case titles, some with all upper-case, and some with mixed title-case. Entries haphazardly capitalize key acronyms such as "BGP", and proper nouns such as "Internet".

CrossTEX ensures that all titles follow the same uniform capitalization standard, even if they appear in a wild variety of styles in the database. By default, the first letter of each word will become capitalized, the rest lower; this is the system known as "titlecase". CrossTEX is very careful to ensure the titles come out looking "good": It retains as-is words in StudlyCaps or CAPITALS, LaTEX commands, and anything in math mode; compound words such as "Peer-to-Peer" are split into words, capitalized correctly, and re-assembled; and finally a list of known phrases are found and formatted. For example, any appearance of a string that is, regardless of case, equivalent to "Internet" will be capitalized as "Internet". CrossTEX determines these phrases at run-time in using `@titlephrase` commands:

```
@titlephrase "USENIX"
@titlephrase "Internet"
```

The standard include files define certain common computer science phrases such as these, but they can appear in any database. Small words, such as "a", "an", "the", etc. are also handled specially: They are made lower-case except at the beginning of the title or after certain punctuation, such as long dashes or colons. These, too, are determined at run-time by `@titlesmall` commands:

```
@titlesmall "a"
@titlesmall "the"
```

Again, the standard include files define common English small words.

Thus an example title with the default capitalization might appear as "Aardvark: A System for Peer-to-Peer BGP Routing on the Internet", despite messy or inconsistent capitalization in the database.

CrossTEX provides other capitalization options: With `--titlecase lower`, only the first letter of the title and those following punctuation are capitalized, the rest put into lower-case. All of the special cases for the default title-case still apply. Thus, the example title would appear as "Aardvark: A system for peer-to-peer BGP routing on the Internet".

With `--titlecase upper`, everything, even known phrases and small words, are put into upper-case, thus: "AARDVARK: A SYSTEM FOR PEER-TO-PEER BGP ROUTING ON THE INTERNET". Commands and math mode are still parsed and protected.

Finally, `--titlecase as-is` tells `crosstex` to allow titles to appear as they are specified in the database.

This philosophy of enforcing consistency by default makes it easier to achieve clean, professional appearance in bibliographies without any interference. At the same time, CrossTEX grants the user control with styles and run-time options and even fine-grained control over how specific phrases should appear throughout the bibliography.

4 Default fields

When databases get large and many elements have very similar fields — if, for example, they are all in the same conference or have the same informative `category` field — the CrossTEX command `@default` can help make them more concise and prevent typos by allowing the maintainer to specify that field just once for the whole group. For example, in the `nsdi.xtx` database, which contains the entries for papers published in NSDI, every entry will obviously have the same booktitle: `nsdi`. Because all of the entries are from the DBLP [13] project, they also share bibsource fields. Finally, they are grouped in the database by year. With default values for fields, maintainers can significantly shorten the database and save effort and typos with the following (line break is editorial):

```
@default booktitle = nsdi
@default bibsource
          = "DBLP, http://dblp.uni-trier.de"
@default year = 2004
```

From this point in the file until the end, entries will by default contain these default fields where they are relevant. Because of this factorization and the inheritance from `nsdi`, most papers now simply state author, title, and pages fields. Later in the file, however, are entries with different years. A new `@default` command takes precedence over the first:

```
@default year = 2003
```

The booktitle and bibsource defaults are unchanged, but now the year defaults to 2003. As with field values inherited from referenced objects, field values inherited from `default` definitions have lower precedence than those specified in the object. If it is desirable to allow an object to appear with no year but not use the default, either put it before the first default specifying the year field, or remove the default by explicitly assigning `@default year = ""`. Defaults are easily overridden and serve only to simplify and shorten the database.

5 Extending objects

Occasionally it is useful to add information to an object that already exists. For example, an author must cite a paper that appeared in NSDI 2005, but the system database only has information about the NSDI conference up to 2004. Obviously, the best solution is to add the following line to fill in the entry in the system conferences database:

```
[year=2005] address=Boston, month=may,
```

However, the author might not have permission to edit the database. Two options come to mind: Cut-and-paste the `nsdi` object into some local database with a new name so there is no conflict, or put the address and month directly into the paper's entry rather than relying on inheritance. Neither is a good solution. More desirable is to be able to add information to the `nsdi` object, even though it has already been defined in another database, using the CrossTEX `@extend` feature:

```
@extend{nsdi, [year=2005] address=Boston,
        month=may}
```

An `@extend` entry looks just like an object definition. However, rather than defining a new object, CrossTEX will find the specified object and re-build it with the information provided, retaining its old fields where they are not changed.

Just as object definitions can specify multiple keys to alias the same object, so can `@extend` statements. If any keys specified do not yet refer to an object, they will be created; however, CrossTEX will report an error rather than change a key that already points to a different object than the one being extended, so it is not possible to accidentally break extant keys. Authors can take advantage of this to define shorter, easier-to-remember names for database objects even when no fields need changing.

6 Constrained citations

CrossTEX borrows from `nbibTeX` the very useful notion of *constrained citation*. Constrained citations enable an author user to cite a work by specifying pieces of information that uniquely identify it. For instance, consider a reference to a paper written by Emin Gün Sirer in 1999 on how to split up virtual machines, which appeared at SOSP. The author could search the database for some terms that will appear in the entry (e.g. 1999, sosp, sirer), copy the key for the entry, and issue a plain citation using that precise key. This is what many BibTEX users do without thinking; however, with constrained citation, CrossTEX will search on your behalf.

A constrained citation begins with an exclamation point, and specifies a series of colon-separated terms that identify the reference being cited. Some examples of constrained citations (line break is editorial):

```
\cite{!author=sirer:title=virtual:year=1999}
\cite{!author=sirer:title=virtual
      :title=machines:year=1999}
\cite{!author=sirer:author=walsh:year=2006}
```

Colons separate constraints. Each constraint identifies a field that the reference must have, as well as a string that should appear somewhere within that named field. So `author=smith` will match both "Smith" and "Smithson".

Sometimes, multiple constraints apply to the same field. Specifying the same field multiple times, as in the second and third examples above, is perfectly acceptable, but tedious. Instead, CrossTEX provides a way to specify multiple constraints for the same field: Every word separated by a "-" sign is treated as a separate constraint. Thus the examples above can appear as:

```
\cite{!author=sirer:title=virtual:year=1999}
\cite{!author=sirer:title=virtual-machines
      :year=1999}
\cite{!author=sirer-walsh:year=2006}
```

Multiple constraints within a given field are not ordered and can appear anywhere in the string, so "virtual-machines" will match "virtual machines", as well as "machines virtual", and even "building a machineshop virtually".

Several shorthands make constrained citations even easier to specify by providing defaults for fieldnames. If the fieldnames are missing, the first constraint defaults to "author". The second constraint defaults to "title" if the value is not numeric; if it is, it defaults to "year". Finally, the last constraint defaults to "year". So the examples above can be even shorter:

```
\cite{!sirer:virtual:1999}
\cite{!sirer:virtual-machines:1999}
\cite{!sirer-walsh:2006}
```

Two caveats are worth remembering about constrained citations. First, the citation needs to be uniquely identifiable. If the constraints match more than one object, CrossTEX prints an error and identify the matching objects. The author can then specify more constraints until the reference is correct or switch to a plain citation based on the search CrossTEX performed. Second, due to a limitation in LATEX, referring to the same work through different constraints (e.g. `!sirer:virtual:1999` and `!sirer:virtual-machines:1999`) will cause CrossTEX to flag an error so the citation does not appear twice in the references section, because LATEX would think it was two different works. For each work, one must decide on a set of constraints and use them consistently throughout a document.

Overall, constrained citations are a convenient way to cite papers without having to look anything up; they fit naturally to the way people recall citations.

7 Extending CrossTEX

CrossTEX is designed to be easy to extend with only trivial knowledge of Python. New objects or fields are defined by editing the standard objects module `crosstex.objects`, which will be found wherever CrossTEX's library files are installed as `crosstex/objects.py`.

To create a new field for a particular object type, find its definition (e.g. the section defining the `@string` object begins `class string`). Most objects already define some fields; simply copy that syntax for your own field. To create an entirely new class `@foo` which is identical to a current one named `@bar`, add the following to the end of the list of objects:

```
class foo(bar):
    pass # 'pass' is only necessary
         # if no fields are defined.
```

Fields are defined as optional or required by assigning them the values `OPTIONAL` and `REQUIRED`, respectively. To make an optional field required or a required field optional, simply assign it the new value in the class where you want the change. To allow a field to inherit its value from another field in the same object if left blank, assign a string containing the name of the other field. A list containing `OPTIONAL`, `REQUIRED`, and one or more string field names will be processed and define several sibling fields and the given requirement level. For example:

```
class foo(bar):
    baz = REQUIRED
    blah = OPTIONAL
    quux = [REQUIRED, 'baz', 'blah']
```

This defines a new kind of object `@foo`, which behaves the same as `@bar`; additionally, the 'baz' field is required, the 'blah' field is optional, and the 'quux' field is required but if unspecified will try to take its value from 'baz' or 'blah' in that order.

Styles are defined in small Python modules in the `style` directory in the same place you found `objects.py`. There you will find the default styles, `plain.py`, `full.py`, etc. Styles are built up with small filter functions, many of which are provided in `crosstex.objects`. Each field is filtered through four phases:

- Production, in which an initial value is generated from the object itself;
- List filtering, if the value is a list;
- List formatting, to turn the list into a string for the final step; and
- Filtering, in which zero or more filters modify the value into its final form.

As a starter, consider the following statements taken from existing styles:

```
misc._addproducer(emptyproducer, 'label')
conference._addfilter(proceedingsfilter,'value')
misc._addfilter(emphfilter, 'fullpublication',
                'booktitle')
```

The first line states that the label attribute of any `@misc` object (or any object of a type derived from `@misc`) can be produced by `emptyproducer` if that function returns anything other than `None`. (`emptyproducer` is defined in `crosstex.objects` and always returns an empty string, which in the case of labels causes LaTeX to default to numeric citation.)

The second line causes the value of objects derived from `@conference` to be filtered through a function that prepends 'Proceedings of the' to the value.

The last line filters the 'booktitle' field of objects derived from `@misc`, but only when used within the 'fullpublication' field (which happens to be a virtual field defined solely by attaching producers to it).

It is important to note that filters and producers are applied starting from the most recent, so later producers will take precedence and later filters will be nested inside earlier filters. The standard styles are well-commented and should provide a good start towards extending CrossTeX with new stylistic features.

Finally, because CrossTeX also searches for styles in the same directory as the file being processed, one can develop styles without editing anything installed system-wide. This can be useful for personal, per-paper, or experimental styles, or even extending CrossTeX without the privileges of the system administrator.

8 Related work

BibTeX is the dominant tool in the TeX world to automatically format bibliographies. The corresponding tool in `roff` typesetting is `refer`, which uses a similar relational model and a more concise but slightly less user-readable database format. `refer` also introduced the notion of semantic (constrained) citation to computer typesetting. `nbibTeX` [15] is a true drop-in replacement for BibTeX that supports the same database language and style files, but adds support for semantic citations in order to assist multiple authors working together from a large database. EndNote [3] is a commercial product that manages databases for Microsoft Word; it uses a GUI database editor and its own database format, adapted from that of `refer`.

Because of BibTeX's dominance but lack of easy automation, the community has developed numerous database editors and other tools to support it. Database editors such as Pybliographer [7], KBibTeX [6], and JabRef [5] are relevant to CrossTeX because they attempt to solve some of the same problems without changing BibTeX and also because, since CrossTeX is backwards-compatible with their BibTeX output, one can take advantage of both.

Other tools for formatting bibliographies for publication on the web exist, such as the BibTeX-XML-HTML [9] project, which provides a tool for converting bibliographies to HTML documents by first converting them into XML. `xtx2html` has the advantage of integrated styling using the same methods as styling documents themselves.

Many projects address the need for large central databases, including the DBLP [13] project, which has so far assembled bibliography entries in BibTeX format for more than 870,000 publications in computer science, some of which have been converted to be distributed with CrossTeX. Other large databases and integrated search engines include CiteSeer [2], Google Scholar [4], and arXiv [10]. RefDB [8] is an approach to the actual sharing of databases, which allows users to share bibliographies over the network using SQL databases and the RIS bibliography format. However, all of these efforts are focused on providing authors with the ability to find a reference in order to copy-and-paste it into their own local database — in short, assisting with looking up the information but not solving the problem of centralizing information because of the limitations of BibTeX. CrossTeX is designed to address this problem and allow vast databases such as these projects to be easily incorporated directly into a document, as well as assisting those who must keep the databases up-to-date.

9 Conclusions

We have presented CrossTeX, a modern bibliography management tool based on and replacing BibTeX. CrossTeX solves a number of problems in BibTeX, including its relational model that requires duplication of information, the dependence of presentation details such as abbreviation on choices made in the database, arbitrary object keys, and an impenetrable style language.

The primary contribution of CrossTeX is its object-oriented database language, which brings the power of inheritance to bear on the goal of specifying information in only one place, to be used and adapted everywhere it is needed. The many kinds of CrossTeX objects allow useful fields to be bundled together, assigned both long and short names, and inherited in different forms throughout the database. New kinds of objects allow precise semantics and self-explanatory databases. Providing full author names and both long and short versions of strings allow typesetting-time decisions about style without modifying the database.

Conditional fields allow data that depend on context, such as locations of a conference by year, to be collected together into one place so they are visible and useful together. Because such conditional fields are also inherited, objects automatically adapt to the contexts in which they are used.

CrossTEX enforces consistency by using a sophisticated and flexible algorithm to guarantee consistent capitalization in paper titles and applying abbreviation decisions across every object. Automation of tedious tasks such as abbreviation and capitalization prevents human error while being easy to customize.

Backwards compatibility with BIBTEX combined with fundamentally new idioms and powerful semantics make CrossTEX easier to use and less error-prone than its predecessor. Overall, CrossTEX makes consistency and professional appearance easy to achieve, both in databases and typeset documents.

References

[1] CS Bib. `http://liinwww.ira.uka.de/bibliography/index.html`. Accessed June 15, 2007. The Collection of Computer Science Bibliographies.

[2] CiteSeer. `http://citeseer.ist.psu.edu/`. Accessed June 15, 2007. Scientific literature digital library.

[3] EndNote. `http://www.endnote.com/`. Accessed June 15, 2007. A PC product that does what BIBTEX does, for Word on Windows.

[4] Google Scholar. `http://scholar.google.com/`. Accessed June 15, 2007. Search engine dedicated to scientific publications on the web.

[5] JabRef. `http://jabref.sourceforge.net/`. Accessed June 15, 2007. A graphical database editor for BIBTEX based on Java.

[6] KBibTEX. `http://www.unix-ag.uni-kl.de/~fischer/kbibtex/screenshots.html`. Accessed June 15, 2007. A graphical database editor for BIBTEX.

[7] Pybliographer. `http://www.pybliographer.org/`. Accessed June 15, 2007. A Python tool for managing bibliographic databases.

[8] RefDB. `http://refdb.sourceforge.net/`. Accessed June 15, 2007. RefDB is a reference database and bibliography tool for SGML, XML, and LATEX/BIBTEX documents that allows users to share databases over a network.

[9] BIBTEX-XML-HTML Project. `http://www.authopilot.com/xml/home.htm`. Accessed June 15, 2007. Transforms BIBTEX databases into HTML by way of XML.

[10] arXiv. `http://arxiv.org`. Accessed June 15, 2007. E-prints in physics, mathematics, computer science and quantitative biology.

[11] Donald E. Knuth. *The TEXbook.* Addison-Wesley, Reading, Massachusetts, 1984.

[12] Leslie Lamport. *LATEX: A Document Preparation System.* 2nd Edition, Addison-Wesley, 1994.

[13] Michael Ley. DBLP. `http://www.informatik.uni-trier.de/~ley/db/`. Accessed June 15, 2007. DBLP is a huge effort by a dedicated team that has so far assembled bibliographic entries for 830,000 publications in computer science. The databases shipped with CrossTEX are derived from DBLP.

[14] Oren Patashnik. BIBTEXing. February 1988.

[15] Norman Ramsey. NbibTEX. `http://www.eecs.harvard.edu/~nr/nbibtex/`. Accessed June 15, 2007. The origin of constrained citation.