Beastie: A new (and the old) bibliography processor

Norman Gray

Abstract

I describe 'Beastie', a bibliography processor designed to be highly compatible with BibTeX, but offering a different path for development.

1 BibTeX vs. Beastie?

BibTeX has been nearly finished for some decades now. Though BibTeX databases and style files are pervasive in many technical disciplines, and the program well-understood, the system as a whole has some points of discomfort:

- it is rather loosely documented, with some features described as being 'like Scribe' (a system which is now principally famous for being the thing that BibTeX is not);
- the (unnamed) implementation language for style files could most politely be described as 'esoteric' (even amongst those of us who regard TeX's syntax as perfectly normal);
- processing `.bib` databases with targets other than TeX is possible, but could never be said to be elegant or natural (see above); and
- to the extent that BibTeX is Unicode-aware (or at least Unicode-preserving) it is largely by accident.

Below, in Sect. 2, I expand on these points and mention some explicit goals for Beastie, and the ways in which they relate to other programs in this space.

In Sect. 3, I describe 'Beastie', the functionality of which runs parallel to BibTeX's in two separate ways. First, it is a parser for the BibTeX `.bib` format, and I discuss some of the existing ambiguities, and their possible resolutions. Second, it is a framework for processing the results of such a parse, and it uses that framework to provide an implementation of BibTeX's `.bst` language.

Beastie has explored the edges of BibTeX's syntax, and I have some views on what we find there, discussed in Sect. 4. It supports Scheme as an implementation language (the `.bst` support is just a Scheme program); while LaTeX output remains well-supported, of course, the program is intended to be neutral regarding output in other formats such as HTML or SQL; and it is Unicode-aware throughout.

Beastie is not intended to be an exact duplicate of BibTeX, and certainly not 'bug-compatible' with it, but instead to be easily mistakeable for it in 'normal' use, as well as supporting some simple extensions to the set of `.bib` structures and `.bst` primitive functions. There is of course an implied definition of 'normal' there, which I hope will provoke a discussion about what a future normative BibTeX might look like (even a BibTeX v2.0?).

The program's home online [4] includes a link to the current location of the public code repository, along with a detailed manual and some further discussion of edge-cases.

2 Goals

Beastie has several main goals:

- Explore ways of processing `.bib` files which aren't exclusively based on `.bst` (BibTeX is currently a single-implementation system).
- Support a means of processing `.bib` files which is not exclusively focused on outputting `.bbl` — or, put another way, which is intended to be friendly to arbitrary downstream processing.
- Provide an alternate implementation of the `.bst` language (resolving a couple of `btxhak` ambiguities [14], and aiming to provide better reporting of syntax errors).
- Initiate some sort of conversation about BibTeX standards, and their documentation.

As well as the above, we have some sub-goals.

As a functionality or interface goal, Beastie should be usable as a drop-in replacement for `bibtex`-the-program in most cases: the point is not to replace `bibtex`, but to have it supply our interface choices.

Beastie aims to be Unicode-friendly, of course. By this I mean not just that it can read Unicode input — a default expectation, nowadays — but that it can do Unicode-informed case conversions and sorting, and detect word boundaries in a compatible way [19]. In the current release (v0.12, at the time of writing) Beastie handles case and word-boundaries respectably, but sorting is still naïve.

The initial prompt for the project — the 'itch to be scratched' — was a desire to import the contents of a `.bib` file into an SQL database. This is possible with the existing BibTeX, but requires jumping through `.bst` hoops, and hacky TeX escaping.

The final, practical, goal is that we do want to release a v1.0!

2.1 Other related software

It is not a goal to immediately *replace* BibTeX. The original program manifestly works, and addresses a sufficiently large fraction of the problem that its use is nearly universal in the LaTeX world.

Should we wish to develop BibTeX's bibliographic support, however, we promptly run into challenges. The style-file `.bst` language is a stack-based

doi.org/10.47397/tb/46-2/tb143gray-beastie

language similar to Forth, which is elegant and amusing to use, but rather hard work. Most of the `.bst` files in circulation start with a recital of the source files, derivatives, and derivatives of derivatives, that they have been more or less hackily adapted from.

Although this is not a crowded space, in software terms, BibTeX's main current competition is BibLaTeX [9], which uses the same `.bib` databases (relying on either BibTeX or Biber to parse the bibliography), but relies on LaTeX as the formatting language. This has the advantage of familiarity. Separately from the implementation details, BibLaTeX defines a substantially enlarged repertoire of `.bib` entry types and fields: such an expansion in expressivity is valuable, but few of the types and fields have been implemented in `.bst` style files. It follows that a bibliography database curated with BibLaTeX in mind might be challenging to use with BibTeX for this reason.

Another limitation of BibTeX and its ecosystem is that, although it provides fairly comprehensive support for STEM-style bibliographies, it struggles with citations in fields much outside this area. For humanities or legal bibliographies, the most active support currently comes from BibLaTeX, possibly because the required support is too intricate to make `.bst` an attractive language choice (the `jurabib` BibTeX style aims for this domain also, and uses a version of the BibLaTeX technique). It's possible that Beastie's language could be helpful here.

Beastie is not the first attempt to address the frustrations here. Jean-Michel Hufflen comprehensively reviewed the range of BibTeX alternatives in 2008 [7]. One of these is Hufflen's program MlBibTeX, which is also, intriguingly, a BibTeX alternative implemented in Scheme. It is discussed in [8], but I am not aware of a location of a released version. There is also cl-bibtex [10], which has similar goals to Beastie, though written in Common Lisp; the program is unfortunately not well-known, and has not been recently updated.[1]

Switching programming tradition, Bibulous [5] is a `.bib` parser written in Python which, as well as providing programmatic access to the parse result, supports an attractively compact templating language for describing bibliography formats.

Outside of the LaTeX and STEM worlds, the MARC format [11] is well-established as a transfer format for bibliographic and library records, with no standard associated formatting mechanism; in contrast, the Citation Style Language[2] is a standardised specification for styling bibliographic information, descriptive rather than executable, implemented in a variety of reference-management products and their associated databases, both proprietary and open.

BibTool is a program for parsing and manipulating `.bib` databases, with the output being other `.bib` files rather than formatted bibliographies. It is built around a library which is intended to be repurposable. As such, BibTool overlaps with Beastie in some of its goals, and the collection of 'instructive examples' in its manual represents a collection of bibliography tasks which would be straightforwardly implementable as Beastie programs.

The BibTeX program has already been forked, first in `bibtex8`, to support multiple international alphabets, with consequent adjustments to `.bst` primitives; and subsequently in BibTeXu [6], which supports specifically Unicode (using the standard ICU library [18]), thus bringing in comprehensive support for Unicode-informed case-changing and sorting, and modest extensions of the set of `.bst` primitives.

Beastie addresses the problem of how, given a `.bib` database, one might do something interesting with it, in a language that is not painful to write. It is not a specific reaction to these other programs,[3] but it is distinguished from them in two ways. Firstly, because it supports the `.bst` language, it is literally a drop-in replacement for BibTeX, and aims to be almost completely compatible with it, whilst opening the door to alternative programming approaches. Secondly, and despite that, it does not aim to fully duplicate BibTeX, but makes some different choices with respect to details which have been points of friction in BibTeX's long history. If Beastie ends up replacing anything, it will be by reimplementing and expanding the use-cases covered by the standard program. Broadly, if there is a case for developing an immediate extension or successor to BibTeX, then Beastie might be a suitable place to start.

3 Beastie, the program

The most straightforward usage of Beastie is to use it as a drop-in replacement for `bibtex`:

```
% latex foo
% beastie foo    # reads .aux, writes .bbl
% latex foo
```

That is, the Beastie program contains an implementation of the `.bst` language, and if the program

[1] There appers to be a significant overlap between people driven to action by the irritation of `.bst` files, and people with a taste for parentheses. I have no ready explanation for this, but offer it as a meditation on the poetics of code.

[2] `citationstyles.org`

[3] Beastie started out as a weekend exploration ('how hard can it be?'), but expanded exactly as these things usually do. I had only a vague initial awareness of MlBibTeX and cl-bibtex, and discovered Bibulous only late in the process.

```
@string{kerala = "केरल"}

@book{माधव,   % mādhava,
  author  = {संगमग्राम के माधव},
  title   = {Venv{\=a}roha},
  शीर्षक    = "वेण्वारोह",
  address = kerala,
  year    = 1403
}
```

Figure 1: A sample BibTeX database, with some Unicode in it

is invoked giving an `.aux` file as argument, it will process it using the indicated BibTeX style, as usual.

The `.bst` implementation aims to be complete, but with the addition of a few extra primitives. The function `show.stack$$` displays the current stack, just like `stack$` does, except without popping it; and `printf$$`, `printf.push$$` and `printf.pop$$` allow some formatted printing, and redirection of output. These are not major changes, but might be welcome to anyone who's written `.bst` style files, and demonstrate that extensions to this language do not require major software effort. As further examples, Shell and Hoadley describe and implement a number of useful extension functions [16], several of which would be excellent candidates for implementation as new primitives. The implementers of BibTeXu [6] modified some primitives to make the program Unicode-aware, and added a few. Would anyone be interested in regexps in `.bst`?

As well, Beastie tries to be a little more informative about `.bst` syntax errors, and their location. Although BibTeX reports an error line when it can, it tends to be simply the line number of the `ITERATE` command, no matter which function the error actually happened in.

The sample bibliography in Fig. 1 contains a number of Unicode characters in illustrative places.[4]

Beastie requires entry keys and field names to be from a slightly restricted set of Unicode characters, which I will return to shortly; field values can be arbitrary Unicode. Also line comments are recognised starting with a `%` character.

The tail end of a Beastie program is in Fig. 2. The details of the functions here are in the Beastie documentation: I include this merely to illustrate the appearance of a Beastie program, and that it is

[4] Mādhava of Sangamagrāma was the early 15th century founder of the Kerala school of mathematics, associated with, amongst numerous other things, the development of infinite series for trigonometrical functions. Thanks to his successors in Kerala today for hosting TUG 2025, where a version of this paper was initially presented.

```
;; ...define *aux-file* as file,
;; and select print-entries! for
;; .bib/.crossref/.yaml
;; as the chosen output function

(print-entries!
  (call-with-aux-file
      *aux-file*
    (lambda (citations bibfiles _ignored)
      ;; call-with-aux-file guarantees
      ;; citations variable is a list
      (unless (list? bibfiles)
        (eprintf
          "no \\bibdata in file, ~a~%"
          *aux-file*)
        (exit 1))
      (assemble-bibliography
        citations
        bibfiles))))
```

Figure 2: A Beastie program (partial)

straightforwardly a high-level program in idiomatic Scheme.

Why Scheme? If we may repurpose the language of the Lua reference manual, Scheme is a powerful, efficient, lightweight embeddable language, with automatic memory management and a garbage collector (and tail-calls!). Though it does admittedly look unfamiliar to many people, it's also one of the few languages which it is a positive pleasure to write. There are many Scheme implementations; Beastie happens to be written on top of s7 [15], which is designed as an extension language, and all of the code for which is contained within two files in the Beastie source tree. There are numerous introductions to Scheme; regarding reference material, the terse classic 'R^{5}RS' [1] is the mostly widely known and implemented standard, and 'R^{7}RS' [3] the most recent.

Beastie is straightforward to build from source: since the Scheme implementation is embedded, there is no separate build system or runtime to install and wrangle. The download includes reference documentation discussing the program's functionality, and its relationship with BibTeX.

The distribution also includes a few example programs. One of them, `extract-bib.scm`, implements the 'hello world' of `.bib` processors, processing an `.aux` file to extract the cited references into a new `.bib` file; this is the program that Fig. 2 is the tail end of. Alternatively, the program can generate a block of Crossref output, as in Fig. 3, or `.yaml`, as an easy encore.

```
% beastie examples/extract-bib.scm \
  -O crossref examples.aux
<citation_list>
<citation key="माधव">
<author>संगमग्राम के माधव</author>
<volume_title>Venvāroha</volume_title>
<cYear>1403</cYear>
<unstructured_citation>माधव, संगमग्राम के (1403),
  ‘Venvāroha’,
  PUBLISHER.</unstructured_citation>
</citation>
</citation_list>
```

Figure 3: Example Crossref output, derived from Fig. 1 using the program in Fig. 2.

Another distributed example program gives a preliminary implementation of RMarkdown-style citations[5] using a `.bib` database in a LaTeX-less workflow. The example combines parsing the source Markdown (Beastie contains a basic Markdown parser) and integrating the results in an output HTML document, but Beastie would be as happy acting as the BibTeX-like step in a more elaborate workflow.

4 Thoughts on BibTeX, and `.bib` standards

As is well known, the BibTeX specifications are a little vague in places: there is no formal grammar for `.bib` databases, and the detailed documentation consists of the relevant part of the LaTeX book, and the distributed ‘btxdoc’ document [13]. This means that there are a number of cases where the question ‘is this valid or not?’ is very hard to answer.

4.1 BibTeX and its grammar

In 1993, Nelson Beebe discussed the then-current State of the BibTeX Nation [2]. In this article, he (i) suggested a formal grammar for `.bib` files and discussed its lexical space, (ii) announced some bibliography validation and transformation tools, (iii) proposed some modest syntactic extensions (`%`-comments, and an `@include{}` form), and (iv) made the obvious point that it would be useful to pin down what is and isn't permissible in a bibliography citation key, suggesting `[A-Za-z][-A-Za-z0-9:.+/']*`.

One notable thing about this discussion of the then-current state of `.bib` syntax is that it also describes the *now-current* state, 32 years later.

In writing Beastie, (i) I have suggested a formal `.bib` grammar, (ii) I offer the Beastie framework as a mechanism for writing transformation tools, (iii) I implement both Beebe's extensions, and (iv) stoutly concur with the desire for syntactical

[5] `bookdown.org/yihui/rmarkdown-cookbook/bibliography.html`

```
input: opt_interentry_text \
       stanza.list opt_interentry_text

stanza.list: stanza
  | stanza.list opt_interentry_text stanza

opt_interentry_text: /* empty */
  | opt_interentry_text INTERENTRYTEXT

stanza: entry | atpreamble | atstring
  | atcomment | atinclude

list.of.fields: field
  | list.of.fields ',' field
  | list.of.fields ',' /* max 1 trailing */

entry: ENTRYTYPE '{' NAME ',' \
         list.of.fields '}'
  | ENTRYTYPE '{' NAME '}'
  | ENTRYTYPE '{' NAME ',' '}' /* ditto */

field: NAME '=' string

atpreamble: ATPREAMBLE '{' string '}'

atstring: ATSTRING '{' NAME '=' string '}'

atinclude: ATINCLUDE '{' NAME '}'

atcomment: ATCOMMENT string

string: STRINGVALUE | NAME
  | string '#' STRINGVALUE
  | string '#' NAME
```

Figure 4: A candidate BibTeX grammar.
The linebreaks marked with trailing backslashes are added for formatting reasons.

explicitness, while making some slightly different suggestions about rules for names, below.

The suggested (yacc) grammar is as in Fig. 4. The terminals in this grammar are `ENTRYTYPE`, representing `@article` and the like; terminals representing the standard `@preamble`, `@string` and `@comment` forms, and a further terminal representing the extension `@include` form (all case-insensitive); `NAME` contains an entry key or field, `STRINGVALUE` contains as value the *content* of either `"..."` or `{...}`, and `INTERENTRYTEXT` contains any other text between entries, that does not contain `@`. The remaining terminals are `','`, `'='`, `'#'`, `'{'` and `'}'`, which have been stripped of surrounding whitespace by the lexer.

The latter two can be produced by the corresponding characters, or by `'('` or `')'`, in a modest (and sole!) concession to Scribe's continuing legacy in

extant `.bib` files, and to the fact that these do occasionally appear in distributed bibliographies. Beebe's article, incidentally, provides some hints about the Scribe format, which might be interesting to those of a historical inclination.

In fact, Beastie uses a slightly different (non-yacc) grammar framework to do its work, but it should be equivalent to this one, and I propose this one as a standard.

This grammar is simpler than that in [2] for a number of reasons. It is a goal of the Beebe grammar not to lose information, so that it can perform lossless conversion of the contents to other formats, and can prettyprint databases. Beastie is more willing to throw information away, and although the `INTERENTRYTEXT` and `@comment` productions do appear in the grammar, my expectation is that their values will be discarded in processing. Secondly, I propose (below) slightly different validity rules for entry keys and field names, but in Beastie this is enforced at a later stage of processing, and does not have to be represented in the lexer or grammar. Finally, this grammar offloads work onto the lexer, so that whitespace does not appear in this grammar, and the proposed `%`-style comments, being regarded as whitespace, do not appear either.

The BibTeX documentation indicates that constructions such as `{\"o}` — that is, a single-character command-sequence at brace-level 1 — should be treated specially in terms of character counting. Beastie also recognises these constructions, but transforms them into the corresponding Unicode character at parse time, using a fixed internal table. This is illustrated in the `title` field in Figs. 1 and 3, meaning that there are no residual TeX-isms in this XML output.

BibTeX restricts entry `crossref` fields to a single indirection. Beastie permits a chain of these, but (provisionally) retains the requirement that they be only forward references.

4.2 Syntactical questions: BibTeX

In this section, I gather a few of the more important ambiguities in this space, and Beastie's response to them. Beebe also investigates the behaviour of `bibtex`-the-program here [2, §9], and comes to compatible conclusions.

The original BibTeX is implicitly ASCII-only, though as noted above `bibtex8` supports a number of international alphabets, and BibTeXu supports specifically Unicode. Beastie expects input to be Unicode (only UTF-8 at present, but supporting UTF-16 and UTF-32 remains an exercise for the future).

The only concession Beastie makes to BibTeX's Scribe legacy is that it permits entries to be surrounded by `(...)` rather than exclusively `{...}`.

The BibTeX documentation is silent on which characters are permissible in an entry key, or field name (other than implicitly, by describing only a fixed repertoire of recognised fields). The `bibtex8` program seems willing to accept almost anything as an entry key. It is surely useful to make a rule explicit here, and in particular to tentatively restrict this, for the sake of whatever processors might handle the data downstream.

Beebe [2, §10] suggests that the name pattern be `[A-Za-z][-A-Za-z0-9:.+/']*`. The simplicity of such a rule is attractive, and it might not have been a problem if it had been adopted in 1993. The problem now, however, is that there are many bibliography files in circulation which use keys which do not match this and which, from their owners' points of view, work perfectly well with BibTeX. It would be both infeasible and pointless to break this.

Instead, I suggest two different validity constraints for entry keys and entry fields. I propose entry keys be a sequence of Base Multilingual Plane (BMP) characters in the following groups:

- 'word characters', processed case-sensitively,
- digits (Unicode character category 'N', obviously overlapping with ASCII `[0-9]`),
- symbols (Unicode 'S'; the ASCII characters with this property are `[$+<=>^|]`),
- or `[:./_&-]` (which Unicode regards as 'punctuation').

The 'word characters' here are those characters that may be regarded as part of a word. There isn't a formal definition of this set in Unicode, but a conventional definition is the union of characters which possess the Unicode properties 'Alphabetic', 'Diacritic', 'Extender' or 'Join Control' (see the Unicode Character Database (UCD) [17] for intricate details).

Field keys can reasonably be more constrained, and I suggest they be limited to word characters, digits and the hyphen, processed case-insensitively.

This definition of entry keys is well-defined (albeit with a repertoire which could in principle change as the UCD evolves), broad without simply admitting arbitrary bytes, and pragmatic where it overlaps with ASCII. There is of course a certain arbitrariness when it comes to selecting which symbols and punctuation should be included or excluded; though all of the non-A–Z characters would require some care if they appeared in a downstream pipeline, this list excludes those which are (it seems to me) egregiously likely to cause chaos. Characters such as `[\'"]`, which would

need escaping in most languages' string syntaxes, end up in the excluded list (with apologies to the citable O'Briens and al-Ma'muns of this world). It might be desirable to make the list of allowed characters smaller, at the risk of causing Beastie to reject `.bib` files which BibTeX is happy with. But while we might accept some modest breakage as collateral damage of a putative standardisation process, if we were determined to make things easy for downstream applications, we would have to exclude characters like colons and ampersands, where the breakage would be out of all proportion to the pain.

The definition of the field keys matters less, since in principle there is only a short pre-existing list of BibTeX-defined values. But BibTeX is perfectly happy to parse non-standard keys in databases, and I have seen digits and hyphens deployed in practice.

I stress that these restrictions refer to entry keys and field names only, and the field *values* should continue to accept any valid Unicode. I don't believe there is much of a case for accepting any other 'international alphabets', such as the various ISO-8859 encodings.

As noted above, Beastie diverges from BibTeX in permitting multiple `crossref` redirections, in treating `{\"o}` constructions specially, and in supporting the `%`-comments and `@include` forms which Beebe proposed.

4.3 Syntactical questions: `.bst` and btxhak

The language standard here is the 'Writing BibTeX Styles' document (the 'btxhak' document) contained in the BibTeX distribution [14]. It can be usefully supplemented by the observations and suggestions in Shell and Hoadley's BibTeX FAQ [16]. Nicolas Markey's 'Tame the BeaST' document [12] is an account of the whole BibTeX ecosystem, including the only other description of the `.bst` language that I know of. The `.bst` language's syntax and semantics are better documented than the `.bib` format, but there are still some ambiguities (some minor ones are enumerated in the Beastie documentation).

The btxhak document, in its final section, implies a grammar for the format-strings used by the `format.name$` primitive. I believe it is equivalent to that in Fig. 5. Here, token `NONALPHA` is a string of non-alphabetic letters, token `STRING` is a sequence of any characters other than `'{'` or `'}'`, and token `FMTSYM` is one of the name-part tokens `f`, `ff`, `v`, `vv`, `l`, `ll`, `j` or `jj`. The grammar here requires that the underlying lexer is sensitive to the brace-level, and will produce tokens `FMTSYM`, `NONALPHA`, `'~'` and `'~~'` only within brace level 1, and will object to any letters other than `[fvlj]` at that level. This

```
input: list.of.specs

list.of.specs: string.or.piece
  | list.of.specs string.or.piece

piece: '{' list.nonalpha.or.tie \
       fmt list.nonalpha.or.tie '}'

list.nonalpha.or.tie: /* empty */
  | nonalpha.or.tie
  | list.nonalpha.or.tie nonalpha.or.tie

fmt: FMTSYM | FMTSYM '{' maybe.string '}'

string.or.piece: STRING | piece

nonalpha.or.tie: NONALPHA | '~' | '~~'

maybe.string: /* empty */ | STRING
```

Figure 5: A candidate grammar for `format.name$` format strings. A 'piece' is a brace-level-1 element like `{f}`. This grammar depends on the lexer being sensitive to the brace-level.

grammar does not capture the semantic observation in the btxhak document that the tie `'~'` should be taken to be discretionary only when it is the last character of a 'piece'.

The BibTeX documentation (and its variants in *The LaTeX Companion*, for example) states that the separator between names is the string `"and"`. Somewhat to my surprise, however, BibTeX matches this *case-insensitively*, so that 'Andrew And and John Smith' is interpreted by BibTeX as one person with surname 'Andrew', a missing author, and a 'John Smith'. That seems sub-optimal to me: there are people in the world who labour technologically with the surname 'Null': we should attempt not to copy their troubles to people with surname 'And'. Beastie *currently* matches BibTeX here, but probably shouldn't.

There is also a problem, easy to miss but hard to ignore, that BibTeX's conventions for structuring a name into first-von-last-junior parts, though elegant, is probably insufficient for a fully international bibliographic solution, and for the wide range of locale-sensitive ways in which real names must be sliced up, processed and reassembled. The Unicode CLDR's 'person-names' schema [20] seems to strike a good balance between modest generality and widely deployed practicality.

Further, but related, the various bracing-and-escaping tricks described in the 'btxdoc' document, and in the Shell and Hoadley FAQ, suggest a notation which is trying to do too many things at once: the 'tricks' for getting BibTeX to behave in the way the

bibliography author wants are so tightly bound to the detailed way that BibTeX is implemented, that we seem to have lost the separation between semantics and execution that LaTeX is properly celebrated for. I don't know if there's a neat solution to this. It might be that more explicit markup would be required in an extended `.bib` format — would XML-structured fields be a terrible idea? — but at present, if a BibTeX-alternative is to be fully compatible with BibTeX-as-is, then it must inherit many ugly and barely intelligible special cases. It is not obvious that Beastie should follow BibTeX far down that particular path, and if there is scope for non-trivial amendments to the `.bib` format, then I suspect that name-structuring would be the first area to see evolution.

5 In conclusion

Beastie is an experiment in parsing and processing BibTeX's `.bib` files. It is not in a position to muscle BibTeX aside,[6] but it might provide a solid and adaptable foundation upon which to build extended or alternative bibliography processors.

Beastie is not yet complete. As of version 0.12, there is still work to do to complete the support for Unicode (the initial attempt to make the program completely standalone is probably infeasible in this area, so it seems inevitable I must engage with the ICU documentation at some point); too many braces survive into the output, as relic TeX-isms rather than being necessary for downstream LaTeX formatting; there will surely be some edge-cases of name-parsing and formatting still to be resolved. The list of unfinished business is not particularly long, but there is enough there to give ample opportunity for feedback and commentary, before a promised version 1.0.

References

[1] N.I. Adams, D.H. Bartley, et al. Revised[5] report on the algorithmic language Scheme. *SIGPLAN Notices*, 33(9):26–76, Sept. 1998. doi.org/10.1145/290229.290234

[2] N. Beebe. Bibliography prettyprinting and syntax checking. *TUGboat* 14(4):395–419, Dec. 1993. tug.org/TUGboat/tb14-4/tb41beebe.pdf

[3] S. Ganz, A.W. Hsu, et al. Revised[7] report on the algorithmic language Scheme, 2013. small.r7rs.org/attachment/r7rs.pdf

[4] N. Gray. Beastie. purl.org/nxg/dist/beastie

[5] N. Hagen. Bibulous. nzhagen.github.io/bibulous/

[6] Y. Haralambous, et al. BibTeXu. ctan.org/pkg/bibtexu

[7] J.M. Hufflen. Languages for bibliography styles. *TUGboat* 29(3):401–412, 2008. tug.org/tugboat/tb29-3/tb93hufflen.pdf

[8] J.M. Hufflen. MlBibTeX now handles Unicode. *TUGboat* 38(2):245–248, 2017. tug.org/TUGboat/tb38-2/tb119hufflen-mlbibtex.pdf

[9] P. Kime, P. Lehman. BibLaTeX. ctan.org/pkg/biblatex

[10] M. Köppe. cl-bibtex. github.com/mkoeppe/cl-bibtex

[11] Library of Congress. MARC 21 format for bibliographic data, 2025. Update 40. www.loc.gov/marc/bibliographic/

[12] N. Markey. Tame the BeaST: The A to X of BibTeX. CTAN, 2009. ctan.org/pkg/tamethebeast

[13] O. Patashnik. *Using BibTeX*. CTAN, 1988. ctan.org/pkg/bibtex

[14] O. Patashnik. *Writing BibTeX Styles*. CTAN, 1988. ctan.org/pkg/bibtex

[15] B. Schottstaedt. s7 scheme. cm-gitlab.stanford.edu/bil/s7

[16] M. Shell, D. Hoadley. *BibTeX Tips and FAQ*. CTAN, 2007. ctan.org/pkg/bibtex

[17] Unicode Consortium. About the Unicode Character Database. www.unicode.org/ucd/

[18] Unicode Consortium. ICU-TC home page. icu.unicode.org

[19] Unicode Consortium. The Unicode standard. Online, 2024. Version 16.0 is current as of mid-2025. www.unicode.org/versions/latest/

[20] Unicode Consortium. Unicode locale data markup language (LDML): Part 8: Person names, 2025. Version 47. www.unicode.org/reports/tr35/tr35-personNames.html

[6] It is, after all, wee, sleekit, cow'rin and tim'rous! I will avoid making too-explicit gags about 'prospects drear', and 'guess and fear'.

◇ Norman Gray
School of Physics and Astronomy
University of Glasgow
UK
norman.gray (at) glasgow dot ac dot uk
https://nxg.me.uk/
ORCID 0000-0002-1941-9202