

— \TeX Blue Note* —

Proposed changes on the *dennis*-branch in \TeX

Dennis Müller
Computer Science, FAU Erlangen-Nürnberg

October 19, 2020

Abstract

Bla.

1 Guiding Principles

- Simplify macro syntax as much as possible
- ⇒ Ideally, have every functionality be governed by a single macro, e.g. only `\begin{module}`[...] instead of `module`, `modnl`, `modsig`, `mhmodsig`, `mhmodnl`, etc. – analogously for `\importmodule`, `\usemodule`, etc.
- get rid of e.g. `mh`-options - \TeX should be able to figure out if we are in a mathhub/smglob repository on its own and resolve paths accordingly
- ⇒ as a user, include a single package that provides everything needed and works with *and* without MathHub, smglob, etc.
- macros corresponding to content with a URI should carry their respective URIs and be accessible *via* their URI, so that overloaded macros/names can be systematically dealt with.
- ⇒ massively simplifies bindings (I hope) by having \TeX itself do the brunt of the work.
- No external tools required (`localpaths`, `sms`)

2 Modules

- `\begin{module}` picks a namespace and an id (if not explicitly provided) ⇒ \TeX knows the full URI of this module.
- The namespace is either provided explicitly (via `[ns=http://mathhub.info/example]`), or take from a `MANIFEST.MF` if in a current repository, or computed from the current file, e.g. `file:///foo/bar/`, if the current file is `/foo/bar.tex`.
- The name of the module is either provided explicitly (via `[id=foo]`), or by default enumerated (`module0`, `module1`, etc.)

*Inspired by the “blue book” in Alan Bundy’s group at the University of Edinburgh, \TeX blue notes, are documents used for fixing and discussing ϵ -baked ideas in projects by the \TeX group (see <http://github.com/sLaTeX/sTeX>). Unless specified otherwise, they are for project-internal discussions only. Please only distribute outside the \TeX group after consultation with the author.

- `\begin{module}[ns=http://mathhub.info/example, id=foo]` will create a new macro `http://mathhub.info/example?foo` and a shorter, accessible macro (maybe, but probably not, `\foo` because of name clashes with symbols in the module) that both expand to `\invoke@module{http://mathhub.info/example?foo}`, which will allow for e.g. accessing symbols in the module, in order to disambiguate e.g. `\natarith{plus}` and `\intarith{plus}`.
- `\begin{module}` sets `\this@module` to `\module@defs@http://mathhub.info/example?foo`.

variants that need to be dealt with:

- `modnl/mhmodnl` - we can replace `\begin{modnl}{title}{lang}` by `\begin{module}[title=...,lang=...]`. Analogously for `mhmodnl`.
In the `smglom`, we might want to consider removing the module-environments entirely, since the same information is already contained in `filename + MANIFEST.MF`.
- `modsig/mhmodsig` - `mhmodsig` isn't even used anywhere. `modsig` could be replaced by e.g. `\begin{module}[title=...,lang=sig]` and analogously to `modnl` could maybe be removed in the `smglom` entirely.

3 Symbols/Notations

Assume we're in module `http://mathhub.info/example?foo`.

- deprecate `\symdef`, `\symvariant` and `\symi` and variants, and replace them by:
- `\symdecl[id=foo]{bar}` – declares a new symbol with URI `http://mathhub.info/example?foo?foo`, creates macros `\http://mathhub.info/example?foo?foo` and `\bar` that expand to `\invoke@symbol{http://mathhub.info/example?foo?foo}`.

If `id` is not provided, the name of the symbol is the macro name, e.g. `bar`.

- Design question: Types for `\symdecl` ? Separate macro or key in the [...] part? Should `TeX` do anything with it? Definientia, too? e.g. via `\abbrdef` ?
- `\notation[lang=...,arity=...,variant=...,arg=...,prec=...]{foo}[n]{...}` declares a new notation for symbol `foo`. `lang`, `arity` and `variant` are keys for different notation “types”, that can ultimately be used via e.g. `\foo[lang=en,arity=2,variant=op]`. `\foo[op]` is shorthand for `\foo[variant=op]`.

`arg` is a sequence of `is` and `as`, where `i` is a simple argument and `a` a flexary/associative argument. If `arg` is not given, the optional `[n]` is used instead (i.e. then `arg` is `in`). If neither is given, but the `arity` keyword is set, then the `arity` is used. If neither is given, the `arity` is 0. Probably we'll also need something like `b` (in addition to `a` and `i`) for “bound” arguments (i.e. bound variables), and maybe others as well? Should `TeX` do anything with `b` arguments, or only `LaTeXML`? Maybe explicitly mark macros that are “allowed” as “head symbols” in `b`-arguments, such as `\setin` ?

`prec` is a string of numbers `psym;p1x...xpm`, where `psym` is the precedence of the symbol (upwards) and the `pi` are the precedences of the individual arguments (downwards)(see below). The default precedence is 0 except if the macro has `arity` 0, in which case it is `-\infprec`.

`\notation` takes an additional argument for each `a` in its `arg` for (the infix-notation of) an associative argument.

Design question: Currently (as in `\symvariant`) the `foo` argument stands for the *macro name* of the symbol that will be given a notation. This is fine in most cases, but not very semantic and weird whenever `id` and macro name of a symbol differ, or `\foo` has been redefined as something entirely different. I propose that instead, `foo` should either be a macro that ultimately expands to `\invoke@symbol{URI}`, in which case `URI`

is the uri of the symbol (this would preserve the current syntax), *or* `foo` is the name of a symbol in the *current* module, or `foo` is a full URI of a symbol in the same or a different module. This URI could externally be accessible via `\invoke@module`, e.g. in `\notation[variant=foo]{\intarith{?plus}}` (which technically wouldn't give the URI of `?plus`, but rather ultimately expand to `\invoke@symbol{...?intarith?plus}`, which is also covered).

- precedences/bracketing: An argument n with precedence p_n in a notation is wrapped in a `\notation@argprec{p_n}{...}`, the *whole* notation is wrapped in a `\notation@symprec{p_sym}{...}`. E.g. `\notation[prec=50;20x20]{plus}{#1 + #2}` would actually have notation `\notation@symprec{50}{\notation@argprec{20}{#1} + \notation@argprec{20}{#2}}`. Associative arguments are wrapped in a `\notation@assoc`, e.g. `\notation[prec=50;20,args=a]{plus}{#1}{+}` results in `\notation@symprec{50}{\notation@argprec{20}{\notation@assoc{+}{#1}}}`
- The initial “downwards” precedence p is `-\infpred`. `\notation@symprec{n}{...}` takes care of inserting brackets, by comparing n with p . If $n \leq p$ (and $p \neq -\infpred$), brackets are inserted. `\notation@argprec{n}{...}` sets the downwards precedence to n . This subsumes `\mixfix-variants`, `\prefix`, `\suffix`, etc.
- For bracketing, the values of `\notation@lparen` and `\notation@rparen` are used. `\withbrackets{a}{b}{...}` temporarily changes those to a and b (`\notation@symprec` changes them back afterwards, so that the changed ones are only used at the specific point `\withbrackets` is used).

Design question: `\notation` probably needs a key `[withbrackets={a,b}]`, because the `\withbrackets`-macro needs to be *outside* of the `\notation@symprec`-macro, which `\notation` wraps around the *whole* notation.

Brackets are prefixed with `\left/\right` in display mode only.

Construction sites:

- `\setnotation[key=value]` globally/locally sets e.g. `lang=de` for all notations.
Tricky: what to do if a symbol doesn't have a `lang=de` notation? How should that interact with explicitly provided notation variants, *both* other e.g. languages (I suggest explicitly provided variants override those of a `\setnotation`) *and* others (e.g. `\foo[variant=op]`) if the *combination* (e.g. `\foo[variant=op,lang=de]`) doesn't exist?
- `\symi` and friends should be deprecated by `\symdecl`. For that, I'd need to figure out what the exact difference is between `\symi{a}{b}` and `\symi{a-b}` and `\symi{a b}`, and rethink `\trefi`-variants analogously.
- `\vardecl` should be like `\symdecl`, but have exactly one notation (**I guess?**) and expand to something like `\invoke@variable{n}` (which gets LaTeXified to an OMV!) rather than having a full URI.
Needs design: Local/global variables? variables that are theory parameters? Universally/existentially bound? Types?
- `\symdecl` should be allowed outside of a `module`-environment, in which case e.g. the *filepath* (or `ns:-field` in the MANIFEST.MF + subfolder in *source*) could provide the namespace and the *filename* the module name. Since this is what e.g. `smglom` consistently does (and to some extent `MiKoMH` as well?), it seems like we should make that the default, which would allow getting rid of the boilerplate in `smglom` files.

4 `\importmodule/\usemodule/\inputref` and variants

- Currently all these commands take care of setting `\this@module`, `\mh@currentrepos`, etc. As a result, we get *some* trouble, e.g. with `\TeX` macros in TOC lines. It seems to me that every module should be in charge of its own location, based on folder/archive-MANIFEST/file, rather than setting and resetting them on `\includemodule`.
- Again, lots of variants that should be unified, e.g. one could use `\includemodule[mhrepo=A/B]` instead of `\gimport` or `\mhinclude` or what else is around there...
- `mh-package` variants currently require `\mh@currentrepos` to be set to something containing a `/`, which is awful.
- All these macros should copy files to a “local cache” (in the PWD probably?), which e.g. can be submitted to `git(labs)/Springer/arxiv` etc. so that the document folder is always self-contained. Consequently, all these macros should check a local cache if e.g. the `MATHHUB` system variable is not set, or the required repo doesn’t exist, or a requested module file can’t be found for whatever reason.

This “cache” should only be a fallback to make submission/collaboration easier, never the *preferred* source for a module. This is something that a package option could turn on/off.

TODO: by default, `argprec = symprec`.

`\imoprmodule{mhrepos=MikMH/AI}{fragments/en?foo}` = module foo in file en or module foo in file foo in directory en