TiCL: the prototype
Star \TeX: the Next Generation (Season 2)

Didier Verna

didier@lrde.epita.fr
@didierverna
facebook/didier.verna
http://www.lrde.epita.fr/~didier

TUG 2013, October 23 – 26
**TEX**

*The [final] frontier.*

*These are the voyages,*

*Of a software enterprise.*

*Its continuing mission:*

*To explore new tokens,*

*To seek out a new life,*

*New forms of implementation.*
TEX’s strength is in the quality of its typesetting, \textit{not} in its programmatic interface.

Keep the typesetting functionality but provide... 
- A more modern and consistent API
- Real programming capabilities
- Still simple to use (at least for simple things)
- Extensibility / customizability
Fitness of Lisp
Previously, on Star TEX TNG...

- Existing approaches are heterogeneous
- What about a fully integrated approach?
- Industrial-scale general purpose language
  - Multi-paradigm
  - Highly optimizable
  - Pletora of libraries
- Scripting / extension language
  - Highly dynamic
  - Highly reflexive
  - Easy to learn (no syntax)
TiCL: the prototype

Current architecture
This is where we start
Not very \TeX y…

cl-typesetting Hello World

```cl
(defun first-document (&key (file "~/tmp/texput.pdf"))
  (tt:with-document ()
    (let ((content (tt:compile-text ()
             (tt:paragraph () "This is some text.")))
          (tt:draw-pages content)
          (when pdf:*page* (typeset:finalize-page pdf:*page*))
          (tt:write-document file)))
```

8/23
Global variables: *title*, *author* etc.

Functions: `document-class`, `make-title` etc.
Note: keywords arguments

Macros: `with-document`, `with-section` etc.
Non lispy accessors (although for constants)
Non lispy function (macro) names
Raw symbols instead of keywords
  ▶ for constant arguments
  ▶ Requires a macro layer

Example

;; Before:
(document-class :article :paper :letter :pt 12)

;; After:
(documentclass article :paper letter :pt 12)
Symbol macros
Turning 0-ary function calls into mere symbols
More non lispy function (macro) names

Example

;; Before:
(make-title)
(table-of-contents)

;; After:
maketitle
tableofcontents
- A \texttt{(par)} command instead of the \texttt{with-par} macro
- And the \texttt{par} symbol macro that goes with it.

**Example**

\begin{verbatim}
;; Before:
(with-par "bla_bla_bla")
(with-par "bla_bla_bla")

;; After:
"bla_bla_bla" \texttt{par} "bla_bla_bla"
\end{verbatim}
- Standalone sectionning commands
- Need to get rid of underlying macros
- Require explicit state management
  
  e.g. PDF outline levels

Example

;;; Before:
(with-section "Title"
  "bla_bla_bla" par
  "bla_bla_bla")

;;; After:
(section "Title")
"bla_bla_bla" par
"bla_bla_bla"
Replacing \texttt{par} with empty lines in Lisp strings

- Requires overriding \texttt{cl-typesetting}'s behavior
- But that's easy (it's Lisp)!

\textbf{Example}

\begin{verbatim}
;; Before:
(section "Title")
"bla bla bla" par
"bla bla bla")

;; After:
[section "Title")
"bla bla bla
bla bla bla"
\end{verbatim}
- When impossible to get rid of the macro layer
- Syntax extension through (read-time) macro characters

### Example

```
;; Before:
(with-document
  "bla_bla_bla"
  "bla_bla_bla")

;; After:
{begin document}
  "bla_bla_bla"
  "bla_bla_bla"
{end document}
```
**LaTeX-like textual layer**

**The ultimate goal**

- **Idea**
  - Remain text-driven instead of program-driven
  - Convert automatically to the programmatic layer

- **Implementation**
  - Use \ as an escape (to Lisp) character
  - Everything else is accumulated into Lisp strings

**Example**

```latex
;; Before:
(section "Lorem Ipsum")
"Lorem_" (textbf "ipsum") "_" (textit "dolor") "sit_amet,...

;; After:
\(\text{section} \text{"Lorem Ipsum"}

Lorem_\(\text{textbf} \text{"ipsum"}) \_\(\text{textit} \text{"dolor"}) \_\text{sit_amet,...}
```
- **Rivers detection**
  - Requires introspection (boxes internal representation)
  - But that’s easy (it’s Lisp, and OO)!

- **Implementation**
  - New kind of box (subclass of *vbox*)
  - Collects content subject to rivers detection
  - Additional stroking method for this new box
  - User-level “rivers” environment
Conclusion

- Last year: fitness of Lisp for a TeX implementation
- This year: proof of concept
- Remember the objectives?
  - A more modern and consistent API
  - Real programming capabilities
  - Still simple to use (at least for simple things)
  - Extensibility / customizability
These were the voyages,
Of a software enterprise.
Its continuing mission:
To explore new tokens,
To seek out a new life,
New forms of implementation.
To \textbf{go},
Where no \TeX has gone before!
Live long and prosper!
Questions?