Star \TeX{}: the Next Generation
Implementing \TeX{} in Common Lisp

Didier Verna

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

TUG 2012, July 16 – 18
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...
Don Knuth @ TUG 2010
Why did you design \TeX\ as a macro-expansion based system?

1. Wanted something simple to use for my secretary
2. Computational resources at the time were limited

Is \TeX\ simple to use, really?
1. Computational resources are not limited anymore
A better \TeX?  
What would that be?

\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
- Backward Compatibility
Wrap \TeX\ in a programming language

Wrap a programming language in \TeX\

Writing macros in another language

Getting rid of macros

Synchronous dual-process (std redirection / file I/O)

Multi-pass

What about a fully integrated approach?

I know, NTS is dead...
Outline

1. Introduction

2. Why Common Lisp?
   - Common Lisp
   - Built-in paradigms
   - Extensibility

3. How to do it?
   - API
   - Compatibility

4. Conclusion
Why Common Lisp?
A language that doesn’t get in the way

- Old language (≠ obsolete, = mature and modern)
- ANSI standard (1994) ⇒ stable
- Industrial-scale general purpose language
  - Multi-paradigm
  - Highly optimizable
  - Pletora of libraries
- Scripting / extension language
  - Highly dynamic
  - Highly reflexive
  - Easy to learn (no syntax)
Built-in paradigms

Free of charge

- key/value interface: functions lambda-lists
- Packages: ASDF systems
- Namespaces: Common Lisp packages
- Interactive behavior: conditions and restarts
- Dumping: Lisp images (idea: user-level dumping)
- Performance:
  - Interpretation / Compilation / JIT-Compilation
  - Static typing
  - And again, dumping

. . .
Extensibility / Customizability
Tweak at will

- Reflection (introspection / intercession)
- Structural:
  - Package internals (: :)
  - ...
- Behavioral:
  - Reader-macros
  - ...
Objectives
Remember them?

✗ A more modern and consistent API
✓ Real programming capabilities
✓ Still simple to use (at least for simple things)
✓ Extensibility / customizability
✗ Backward Compatibility
A more modern and consistent API
Programmatic \TeX\xspace primitives

- Parameters $\Rightarrow$ Lisp variables
  - badness

- Quantities $\Rightarrow$ Lisp objects
  - \begin{verbatim}
(setf baselineskip #g(b :plus x :minus y))
  \end{verbatim}

- Commands $\Rightarrow$ Lisp functions
  - (input file)
  - (hbox material)
  - (hbox material :to dim)
  - (hbox material :spread dim)
  - (hbox-to dim material)
  - (hbox-spread dim material)

- The \textit{typesetting} subset of \TeX
  No $\texttt{\textbackslash def, \textbackslash relax and friends}$
Backward Compatibility
With good’old $\TeX$

- Implement traditional $\TeX$ on top of procedural $\TeX$ (part of) $\TeX$’s digestive engine
- Provide a way to plug Lisp code in $\TeX$ files
  $\TeX$ macros written in Lisp or direct Lisp code
TiCL architecture
An overly simplified, extremely naive, totally wrong view

Procedural TeX

Traditional TeX

User level Lisp

Introduction
Why?
Common Lisp
Built-in paradigms
Extensibility

How?
API
Compatibility
What does it take to embed Lisp in TeX?
Provided TeX is written in Lisp, that is ::)

18/21
Expected problems
Let’s be realistic…

- Huge task
  - CFFI
  - Compatibility mode
- \( \text{T}_{\text{E}}\text{X} \)'s digestive engine is not really a pipeline
- Lisp / traditional \( \text{T}_{\text{E}}\text{X} \) interaction tricky
- Sandboxing
- Too much intercession…
- All the things I haven’t thought of yet (a lot)
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 $\LaTeX$ has gone before!
Live long and prosper!
Questions?