LISP on TeX
A LISP Interpreter Written Using TeX Macros

HAKUTA Shizuya

The 34th Annual Meeting of the TeX Users Group, 2013
Background

- Writing $\text{\LaTeX}$ macros is useful.
  - e.g. Calculating some small numeric expressions.
- However, it is difficult for novice users.
- To improve, there are some researches that combine $\text{\LaTeX}$ and another programming language.
Pakin[TUGboat ’03] showed four ways to connect \TeX{} and a foreign programming language:

1. using $\texttt{\write18}$ to call an outer processor,
   - python package (CTAN:macros/latex/contrib/python)
2. embedding an interpreter in a \TeX{} engine,
   - Lua\TeX{} (CTAN:systems/luatex)
3. constructing macros that enable \LaTeX{} to communicate with an external interpreter,
   - Perl\TeX{} (CTAN:macros/latex/contrib/perltex)
4. creating a language processor with \TeX{} macros
   - \LaTeX{}3 project created expl3: a new interface of \TeX{} macros, but no ordinary language was implemented.
Our goals are

- Implementing a language’s interpreter with \TeX macros, and
- Comparing its performance with other approaches.

We take two design choices;

1. Choosing LISP as a ordinary language, and
2. Creating the product as a \LaTeX package.
We name the our product **LISP on \TeX**.

- It was already archived on CTAN and \TeXLive.
  - `macros/latex/contrib/lisp-on-tex`
- We constructed all parts of LISP on \TeX with \TeX macros;
  - parser, recognizing LISP expressions,
  - evaluator, calculating a expression, and
  - environment, mapping symbols to LISP objects.
- The code is written with traditional \TeX macros only, so it works in all \LaTeX engines,
  - \LaTeX, pdf\LaTeX, Lua\LaTeX, Xe\LaTeX, p\LaTeX, \ldots
Examples (1/2)

Source

The Preamble of the Slides

\usepackage{lisp-on-tex}

\lispinterp{
  (\define \fact
    (\lambda (\n)
      (\lispif (\= \n :0) :1
        (\* (\fact (\- \n :1)) \n)))))
}

Result

$10! = \lispinterp{\text{\(\text{\textbackslash texprint(\text{\textbackslash fact:10})}\)}}$
Examples (1/2)

Source

The Preamble of the Slides

```
\usepackage{lisp-on-tex}

\lispinterp{
  (define \fact
    (\lambda (\n)
      (\lispif (\= \n :0) :1
        (\* (\fact (\- \n :1)) \n))))
```

Result

10! = 3628800
Examples (1/2)

Source

The Preamble of the Slides

\usepackage{lisp-on-tex}

\lispinterp{
  (\define \fact
    (\lambda (\n)
      (\lispif (\= \n :0) :1
        (\* (\fact (\- \n :1)) \n))))}

Result

10! = 3628800

LISP codes were evaluated!
Examples (2/2)
Examples (2/2)
LISP on \TeX uses a lot of memory.
- Yato showed that LISP on \TeX stalls when using a lot of LISP objects\(^1\).
- It is caused by spending a lot of control sequences.
- Building a garbage collection system is one of our future work.

\(^1\)http://d.hatena.ne.jp/zrbabbler/20121116/1353068217
(Japanese Only)
Comparison to other approaches

We compared LISP on TeX and other approaches by three benchmarks.

- CPU Core i7 2.2GHz, 8GByte Memory, W32TeX

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LISP on TeX</td>
<td>13</td>
<td>$1.6 \times 10^2$</td>
<td>$2.1 \times 10^4$</td>
</tr>
<tr>
<td>PerlTeX</td>
<td>1.0</td>
<td>1.0</td>
<td>$1.6 \times 10^2$</td>
</tr>
<tr>
<td>LuaTeX</td>
<td>0.45</td>
<td>0.55</td>
<td>7.6</td>
</tr>
<tr>
<td>TeX macros</td>
<td>0.24</td>
<td>0.22</td>
<td>$1.2 \times 10^2$</td>
</tr>
<tr>
<td>expl3</td>
<td>1.1</td>
<td>1.0</td>
<td>$5.7 \times 10^3$</td>
</tr>
</tbody>
</table>

It shows that LISP on TeX is too slow... :-(

- It is caused by reading TeX tokens repeatedly.
  ⇒ We can make LISP on TeX faster with improving the code.
We implemented LISP on TeX, a LISP interpreter written only with TeX macros.

It works well, but the product has problems about memory usage and speed.
Why LISP is Selected?

There are two reasons why we select LISP.

1. LISP is Turing complete, so it contains all essence of programming languages.
2. Because LISP has simple syntax and semantics, we can implement LISP easily.