LISP on TEX

A LISP Interpreter Written Using TEX Macros

HAKUTA Shizuya

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

Background

- Writing TEX 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 TEX and another programming language.

TEX with Other Languages

Pakin[TUGboat '03] showed four way to connect TEX and a foreign programing language;

1. using `\write18` to call an outer processor,
   - python package (CTAN:macros/latex/contrib/python)
2. embedding an interpreter in a TEX engine,
   - LuaTEX (CTAN:systems/luatex)
3. constructing macros that enable LATEX to communicate with an external interpreter,
   - PerlTEX (CTAN:macros/latex/conrtib/perltex)
4. creating a language processor with TEX macros
   - LATEX3 project created expl3: a new interface of TEX macros, but no ordinary language was implemented.

The Goal and the Mean

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.

LISP on TEX

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, pdfLATEX, LuaLATEX, XeLATEX, pLATEX, . . .

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

```
$10!=\lispinterp{(\texprint(\fact:10))}$
```

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

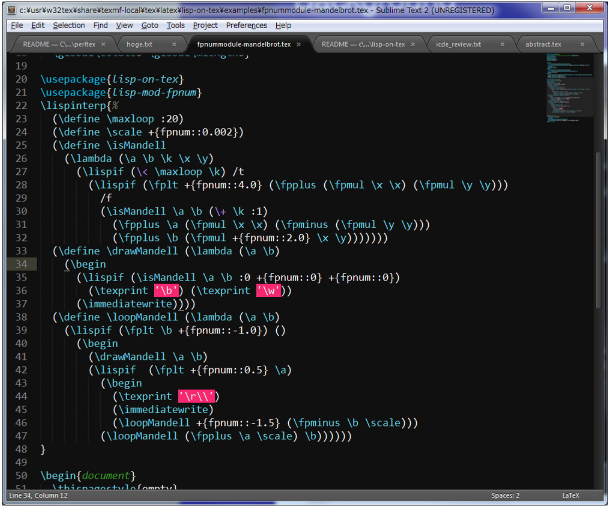
```
\usepackage{lisp-on-tex}

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

Result

$10! = 3628800$

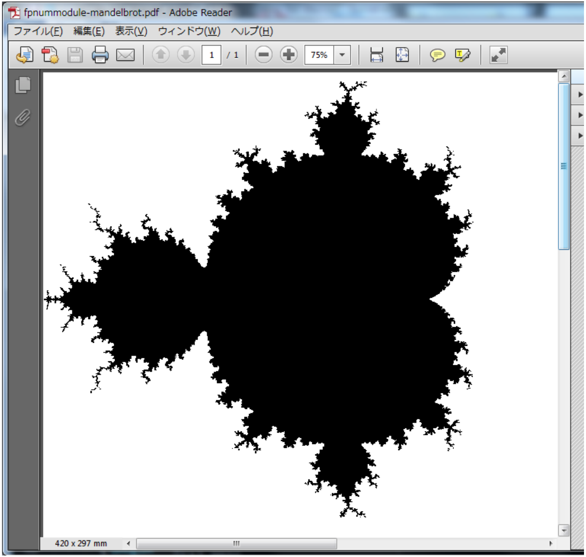
LISP codes were evaluated!

Examples (2/2)


c:¥usr¥w32tex¥share¥texmf-local¥tex¥latex¥lisp-on-tex¥examples¥fpnummodule-mandelbrot.tex - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
README — C\...\perltex
hoge.txt
fpnummodule-mandelbrot.tex
README — c\...\lisp-on-tex
icde_review.txt
abstract.tex
Line 34, Column 12
Spaces: 2
LaTeX

```

\usepackage{lisp-on-tex}
\usepackage{lisp-mod-fpnum}
\lispinterp{%
  (\define \maxloop :20)
  (\define \scale +{fpnum::0.002})
  (\define \isMandell
    (\lambda (\a \b \k \x \y)
      (\lispif (\< \maxloop \k) /t
        (\lispif (\fplt +{fpnum::4.0} (\fpplus (\fpmul \x \x) (\fpmul \y \y)))
          /f
          (\isMandell \a \b (\+ \k :1)
            (\fpplus \a (\fpmul \x \x) (\fpminus (\fpmul \y \y)))
            (\fpplus \b (\fpmul +{fpnum::2.0} \x \y)))))))
  (\define \drawMandell (\lambda (\a \b)
    (\begin
      (\lispif (\isMandell \a \b :0 +{fpnum::0} +{fpnum::0})
        (\texprint '\b') (\texprint '\w'))
      (\immediatewrite))))
  (\define \loopMandell (\lambda (\a \b)
    (\lispif (\fplt \b +{fpnum::-1.0}) ()
      (\begin
        (\drawMandell \a \b)
        (\lispif  (\fplt +{fpnum::0.5} \a)
          (\begin
            (\texprint '\r\\')
            (\immediatewrite)
            (\loopMandell +{fpnum::-1.5} (\fpminus \b \scale)))
          (\loopMandell (\fpplus \a \scale) \b))))))
}

\begin{document}
  \thispagestyle{empty}
```

Examples (2/2)


fpnummodule-mandelbrot.pdf - Adobe Reader
ファイル(F) 編集(E) 表示(V) ウィンドウ(W) ヘルプ(H)
1 / 1
75%
420 x 297 mm

Memory Management Problems

- 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

	tarai[sec]	asterisks[sec]	Mandelbrot[sec]
LISP on TeX	13	1.6×10^2	2.1×10^4
PerlTeX	1.0	1.0	1.6×10^2
LuaTeX	0.45	0.55	7.6
TeX macros	0.24	0.22	1.2×10^2
expl3	1.1	1.0	5.7×10^3

- 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.

Conclusion

- 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.