

# LISP on T<sub>E</sub>X マニュアルもどき

@hak7a3

2012 年 7 月 24 日

## 1 はじめに

こんなネタを投げてみたことがある．

T<sub>E</sub>X によるプログラムは，文書を効率的に見た目よく組版するために必須の技術であるが，T<sub>E</sub>X の文法や意味論などにおける複雑さから敷居<sup>1</sup>ハードルが高い．以前からこれに対する解決案として，他言語に解を求めるものが示されている．LISP による T<sub>E</sub>X のプリプロセッサ [Iwasaki '02] や LuaT<sub>E</sub>X などがそれにあたる．前者の研究は，既存の T<sub>E</sub>X 資源を有効に使える利点があるが，Emacs に依存するため TeXShop などの近代的な編集環境に適用しづらい．後者は pT<sub>E</sub>X の資源を使うためにやや工夫が必要である．そこで本研究では外部環境に依存しない，既存の環境を有効利用できる平易な T<sub>E</sub>X プログラミング環境として，T<sub>E</sub>X 上での言語処理系の実装手法を提案する．本手法によりユーザ，特にプログラミング言語の知識があるプログラマはより簡単に T<sub>E</sub>X を効率的に利用することができる．また，本研究は T<sub>E</sub>X がチューリング完全な言語であるという（忘れられた）事実をすべての T<sub>E</sub>X ユーザに啓蒙するための手段となる．

やってみた．

## 2 文法

LISP on TeX の文法は，（まだ変わるかもしれないけど）次のとおりです．

```
s-expr ::= symbol | nil | list | dot-expr | int | srting | bool
symbol ::= control-sequence | control-symbol
nil ::= ()
list ::= (s-expr +)
dot-expr ::= (s-expr.s-expr)
int ::= :integer
string ::= 'TEX-token * '
bool ::= /{t | f}
```

---

<sup>\*1</sup> 木枝祐介氏の指摘により修正．感謝

integer やら control-sequence やらは  $\text{\TeX}$  の世界の言葉として認識してくれると嬉しいです .

現状 , スペシャルフォーム的な何かとして機能するものは次のとおりです .

```
quote, if, define, lambda
```

このうち , 注意が必要な物は `define` と `lambda` です . `define` は常にグローバル環境の変更としてみなされます . ローカル環境へは手出ししません . `lambda` は一引数関数のみサポートしています . そのため , 適当にカーリー化する必要があります . また , 一引数関数しか定義できない関係上 , 引数部の括弧を省略します . すなわち , `(\lambda \ x (\ + \ x \ 1))` などと記述することになります .

### 3 サンプル

インタフェースとして , “`lispinterpl`” 命令を定義してあります . 引数に任意個の `s-expr` を書けます . `dvi` への出力は “`print`” 関数を用いると嬉しいです . 簡単な例を示します .

```
\lispinterpl{
  (\define \sum (\lambda \n (\if (\ = \n :0) \n (\ + \n (\sum (\ - \n :1)))))
  (\print (\sum :10))}
```

:55