Lisp on TEX II

by HAKUTA Shizuya

Kurt Pagani
nilqed@gmail.com
August 8, 2022

Usage: `\usepackage{lisp-on-tex}` `\input{listfun.sty}`

List functions

```
\writeList \lst \b \sep \e
  % \lst the list to display
  % \b the begin character, e.g. '['
  % \sep the separator, e.g. ', '
  % \e the end character, e.g. ']'
Writes out a list in the form specified.
```

```
\equalQ \x \y
  % \x, \y any lisp type
  % Tests equality recursively.
  % Note that atomQ () -> /f, i.e () is not
      ↪ an atom.
```

```
\atomOrNilQ \x
  % Check whether \x is an atom or the empty
      ↪ list ().
  % Returns /f otherwise.
```

```
\append \x \y
  % Append \y to the list \x.
```

```
\subst \x \y \z
  % Substitute \x for \y in the list \z.
```

```
\memberQ \x \y
  % If \x is a member of \y then return /t
      ↪ else /f.
  % Note: \x may be a sublist, and atoms are
      ↪ members only on first level!
```

```
\pairlis \x \y \a
  % Give the list of pairs of corresponding
      ↪ elements of the lists \x and
  % \y, and appends this to the list \a. The
      ↪ resultant list of pairs, which
  % is like a table with two columns, is
      ↪ called an association list.
```

```
\assoc \x \a
  % If \a is an association list, then
      ↪ \assoc will produce the first pair
  % whose first term is \x. Thus it is a
      ↪ table searching function.
```

```
\sublis \a \y
  % Here \a is assumed to be an association
      ↪ list of the form
  % ((ul . v l ) . . . (un . v,)), where the
      ↪ u1's are atomic, and \y is
  % any S-expression. What \sublis does, is
      ↪ to treat the u1's as variables
  % when they occur in \y, and to substitute
      ↪ the corresponding v1's
  % from the pair list.
  % Note: \sublisXXX is the helper function
      ↪ sub2 in the LISP 1.5 Programmer
  % Manual (from where we have the info:).
```

```
\union \x \y
  % Union of the lists \x and \y
```

```
\intersection \x \y
  % Intersection of the lists \x and \y
```

```
\reverse \x
  % Reverse the list \x
```

```
\foldr \f \x \y
  % Fold right list \y with \f and start \x.
```

```
\foldl \f \x \y
  % Fold left list \y with \f and start \x.
```

```
\filter \f \x
  % Filter the list with the function \f
```

```
\allQ \f \x
  % f(x) true for all x?
```

```
\anyQ \f \x
  % f(x) true for any x?
```

For details and examples consult the manuals
https://github.com/nilqed/lisp-on-tex