Formatted references

A set of Scheme functions for formatting references

in a TeXmacs document - v 0.22

Description

Are formatRef and formatList the names we want?

This program is inspired by the LaTeX package `prettyref`.

It provides the macro `formatRef` (`<formatRef|lab>`) which formats a reference choosing in a list of formats saved in the global variable `formatList`. Each format is a list consisting of two members; the first member is a format code, which is a short string, whicle the second member is the corresponding format to apply to the reference. The function selects the format by comparing the format code with the beginning of the label: the first element of `formatList` that matches is selected and the corresponding format string `formatString` is prepended to the TeXmacs reference, so that the reference obtained with the macro `<formatRef|lab>` looks like

`formatString<reference|lab>`

The labels must be assigned according to the format codes so that the macros have an effect. If no format code matches, the macro returns

`<reference|lab>`

TeXmacs macros and their use

I have defined the TeXmacs macros in the preamble of this document

formatRef

`<formatRef|lab_string>`

Formats a reference so that it looks like

`formatString <reference|lab>`

where `formatString` is the format string of the first matching format (empty string if there is no match).

addToFormatRef

`<addToFormatRef|formatCode_string|formatString_string>`

Adds to the list of formats `formatList` (which is a global Scheme variable) the new format which is composed of the strings `formatCode` and `formatString`.

Example:

`<addToFormatRef|fig:|Figure >` makes the new format `("fig:" "Figure")` available, so that the application of the macro

`<formatRef|fig:figure1>`

returns

Figure `<reference|fig:figure1>`

deleteFromFormatRef

`<deleteFromFormatRef|formatCode_string|n_integer>`

Deletes from `formatList` the n-th format that matches `formatCode`. If no format that matches format code remains after deletion, then

`<formatRef|lab>`

will return

`<reference|lab>`

Example:

`<deleteFromFormatRef|fig:|1>`

deletes from `formatList` the first occurrence of the format with format code `"fig:"`

replaceInFormatRef

`<replaceInFormatRef|formatCode_string|formatString_string|n_integer>`

Replaces in `formatList` the n-th format with matching `formatCode` with the format `("formatCode" "formatString")`.

Example:

Let the initial value of `formatList` be `(list (list "eq:" "eqn. ") (list "Sec:" "Section ") (list "eq:" "equation "))`

`<deleteFromFormatRef|eq:|Equation >`

transforms `formatList` into

`(list (list "eq:" "eqn. ") (list "Sec:" "Section ") (list "eq:" "Equation "))`

in this case the application of the macro

`<formatRef|eq:equation1>`

returns

eqn. `<reference|eq:equation1>`

like it did originally, as the matching format is the first one that matches the format code.

Load Scheme functions

```
Scheme] (load "manipulateList.scm")
Scheme] (load "selectFormat.scm")
Scheme] (load "interfaceToTeXmacs.scm")
Scheme]
```

Test of TEXMACS macros

```
Scheme] (set! formatList (list (list "eq:" "Equation ") (list "fig:" "Figure ")))
Scheme] (extractFormatString "eq:" formatList)
   "Equation "
Scheme]
```

Test format application

$$\sin(x)^2 + \cos(x)^2 = 1 \tag{1}$$

Equation 1

Manipulation of format list

Test addition to format list

addToFormatRef sec: Section

?

```
Scheme] formatList
   (("eq:" "Equation ") ("fig:" "Figure ") ("sec:" "sect. "))
Scheme]
```

Test

sect. 4.2

Test deletion from format list

deleteFromFormatRef sec: 1

?

```
Scheme] formatList
   (("eq:" "Equation ") ("fig:" "Figure "))
Scheme]
```

4.2

Test replacement in format list

addToFormatRef sec: Section

?

replaceInFormatRef sec: Section

?

```
Scheme] formatList
   (("eq:" "Equation ") ("fig:" "Figure ") ("sec:" "sect. "))
Scheme]
```

sect. 4.2

To do

- `formatList` should perhaps be a tuple
  - In this way it could be easy to initialize it
- I need to load the Scheme functions automatically
  - Is it possible?
- Certain TEXMACS macros - the ones that manipulate the list - should be called only once.
  - How can I make sure of this?
  - Should I put them in the preamble?
- General error checking
  - For example, what happens if I input a non-integer number in the macros for list manipulation?
- Error messages are sent to the standard output. I would need nice messages in TEXMACS too.
- The space after the format string should be dealt with well