texp Manual

Max Rottenkolber

Sunday, 26 July 2015

Table of Contents

1 Abstract

`texp` is a domain specific language embedded in *Common Lisp* for printing *TeX expressions*. Because, let's face it, *TeX* is a horrible programming language. I created `texp` to ease the programatic generation of *TeX* documents and programs from within *Common Lisp*. By intertwining both languages, `texp` brings high level meta programming to *TeX* and enables you to produce high quality print media in an easy and scalable fashion.

If you have no experience with *TeX* then none of what follows will make any sense to you. `texp` has no model of *TeX's* semantics, besides escaping special characters. For an introduction to *TeX* I recommend Knuth's *TeXbook*.

2 A brief example

To utilize `texp` we need to intern the `texp` package and import the `texp:syntax` readtable:

```
(defpackage texp-example
  (:use :texp :named-readtables))

(in-package :texp)

(in-readtable texp:syntax)
```

`in-readtable` is provided by `named-readtables`.

Assume we need to generate a localized document. We could do that by using a function that accepts localized captions and fills in a *TeX* document.

```
(defun tex-menu (menu translations)
   (deftex item (caption price translation)
     ($ caption) " " ($ price)
     (br)
     ($ translation)
     (bigskip))
   (loop for (caption price) in menu
         for translation in translations
     do (tex (item {($ (escape caption))}
                   {($ (escape price))}
                   {($ (escape translation))})))
   (tex (bye)))
```

`deftex` enables us to use *TeX*'s `\def` with more descriptive parameter names. `br` prints a double newline and `escape` handles escaping of *TeX*'s many special characters. *TeX* macros can be expressed in a lispy way.

3 The `tex` macro

The `tex` macro translates its child expressions to *TeX* expressions. It is a very thin abstraction at the syntax layer. The translation rules are listed below:

s-expression	**translates to**
`(foo bar baz)`	`\foo barbaz`
`[foo bar]`	`[foobar]`
`{foo bar}`	`{foobar}`

`tex` translation rules. `foo` and `bar` could be strings or numbers as well. In case of symbols, the `symbol-names` are printed in lower case for convenience. Brackets and curly braces require the readtable `texp:syntax`. All expressions can be nested.

The `(br)` form prints two newlines, used to seperate paragraphs in *TeX*:

```
(tex (br))
▷
▷
```

The `$` form lets you interpolate values into `tex` bodies. Consider the following example:

```
(let ((x "Hello!"))
  (tex ($ x)))
▷ Hello!
```

The `escape` function can be used to escape characters treated specially by *TeX* in strings of input. A table of escape rules is bound to `*escape-table*`.

```
(tex ($ (escape "$$$")))
▷ \$\$\$
```

4 The `deftex` macro

The `deftex` macro simplifies writing *TeX*'s `\def` statements using `texp`. It's best described by example:

```
(deftex foo (parameter-a parameter-b)
  (bar ($ parameter-a) ($ parameter-b)))
▷ \def \foo #1#2{\bar #1#2}

(tex (foo {"hello"} {"world"}))
▷ \foo {hello} {world}
```

As you can see, `deftex` does not do much at all. Nevertheless, it provides a lispy syntax for *TeX* `\def`'s and enables you to use descriptive parameter names by hiding *TeX*'s natural parameter syntax with a `let`.