CL-Arrows for Emacs

This is a port of CL-Arrows to emacs lisp. The only real changes were using lexical binding and changing the test suite to ERT and adjusting the tests for the fact that emacs lisp doesn't have fractions.

Implements the -> and ->> threading macros in Clojure, as well as -<> and -<>> from the swiss-arrows library.

Documentation

[macro]
-> initial-form *&rest* forms => results

Inserts INITIAL-FORM as first argument into the first of FORMS, the result into the next, etc., before evaluation. FORMS are treated as list designators.

[macro]
->> initial-form *&rest* forms => results

Like ->, but the forms are inserted as last argument instead of first.

[macro]
-<> initial-form *&rest* forms => results

Like ->, but if a form in FORMS has one or more symbols named <> as top-level element, each such symbol is substituted by the primary result of the form accumulated so far, instead of it being inserted as first argument. Also known as diamond wand.

[macro]
-<>> initial-form *&rest* forms => results

Like -<>, but if a form in FORMS has no symbols named <> as top-level element, insertion is done like in ->>. Also known as diamond spear.

Examples

```
(-> 3
    /)  ; insert into designated list (/)
=> 1/3

(-> 3
    (expt 2))  ; insert as first argument
=> 9

(->> 3
     (expt 2))  ; insert as last argument
=> 8

(-<>> (list 1 2 3)
      (remove-if #'oddp <> :count 1 :from-end t) ; substitute <>
      (reduce #'+)                               ; insert last
      /)                                         ; list designator
=> 1/3

(let ((x 3))
```

```
  (-<> (incf x)      ; (let ((r (incf x)))
       (+ <> <>)))   ;   (+ r r))
=> 8
```

Todo

Future versions *might* include further ideas from rplevy's swiss-arrows.