3rd Clause

```
((_ (hd . tl) literals match-value fail-value success-expr)
     (cond ((eq? match-value fail-value)
            fail-value)
           ((exists-in? 'hd 'literals)
            (if (eq? (maybe-car match-value fail-value) 'hd)
                (match-pattern tl literals (maybe-cdr match-value fail-value)
                               fail-value success-expr)
                fail-value))
           (else
            (let ((hd (maybe-car match-value fail-value)))
              (if (eq? hd fail-value)
                  fail-value
                  (match-pattern tl literals (maybe-cdr match-value fail-value)
                                 fail-value success-expr))))))
```

failure under way

hd is a literal
try matching tail (tl)

hd is an atom
try matching tail (tl)

pattern is a list
deconstruct it into 2 parts
hd and tl

hd might be a literal, like
"*else*"

note to self: what are the quotes in
`(exists-in? 'hd 'literals)`
`[looks like a typo]`

4th Clause

```
((_ non-list literals match-value fail-value success-expr)
    (cond ((eq? match-value fail-value)
           fail-value)
          ((exists-in? 'non-list 'literals)
           (if (eq? 'non-list match-value)
               success-expr
               fail-value))
          (else (let ((non-list match-value)) success-expr)))
```

failure under way

pattern is a single literal
succeed only if match-value
is the same literal

pattern is a single
atom (non-literal);
bind match-value to it,
then eval success-expr
(probably creating more
macro output)

pattern is not a list
(all list cases have been weeded out at this point)
pattern is bound as "non-list"
deconstruct it into 2 parts
hd and tl