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