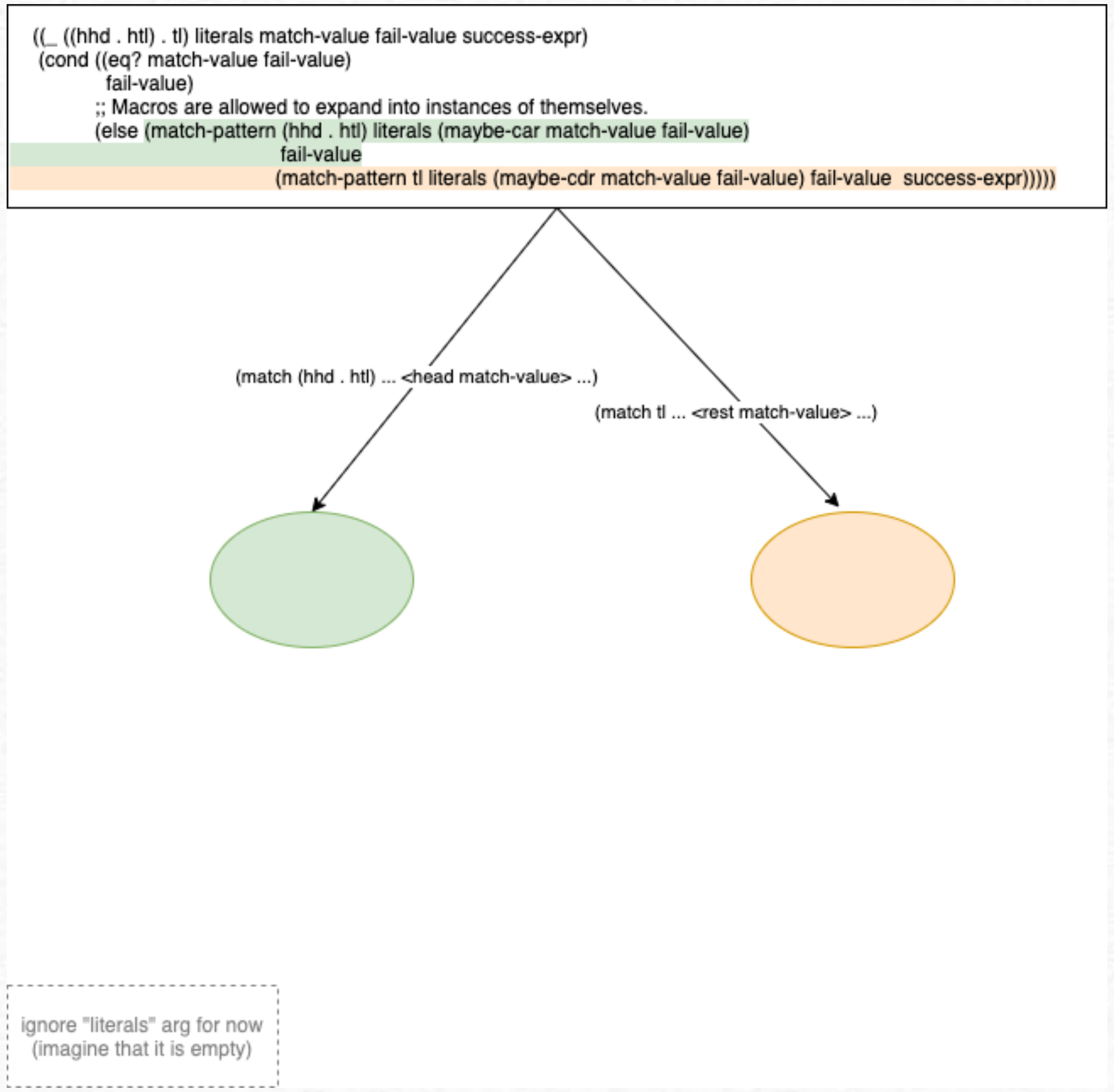
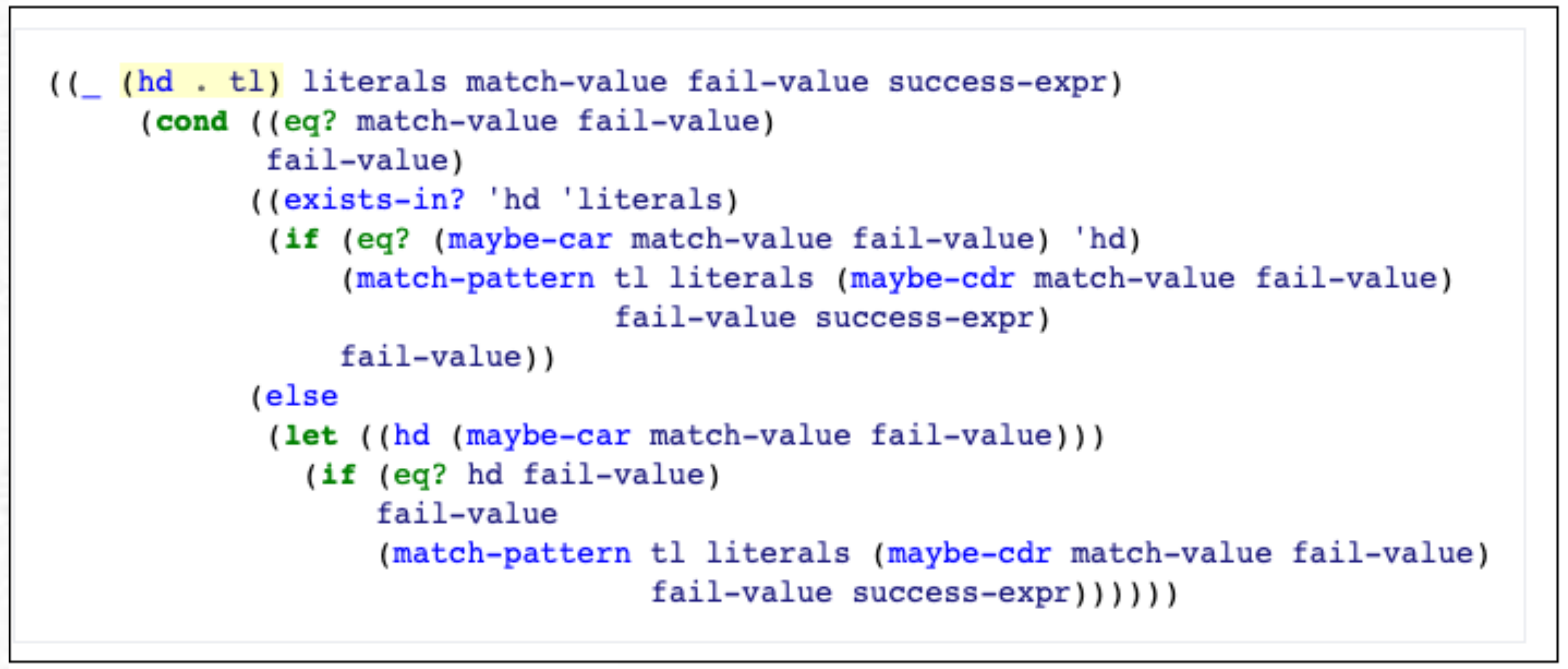
2nd Clause



```
((_ ((hhd . htl) . tl) literals match-value fail-value success-expr)
 (cond ((eq? match-value fail-value)
          fail-value)
         ;; Macros are allowed to expand into instances of themselves.
         (else (match-pattern (hhd . htl) literals (maybe-car match-value fail-value)
                              fail-value
                              (match-pattern tl literals (maybe-cdr match-value fail-value) fail-value  success-expr)))))
```

(match (hhd . htl) ... <head match-value> ...)
(match tl ... <rest match-value> ...)
ignore "literals" arg for now (imagine that it is empty)

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]`