

Próximo:[Condicional de correspondência de padrões](#), Anterior:[Condicionais](#), Acima:

[Estruturas de controle](#) [[Conteúdo](#)][[Índice](#)]

11.3 Construções para Combinar Condições

Esta seção descreve construções que são frequentemente usadas em conjunto com `if` e `cond` para expressar condições complicadas. As construções `and` e `or` também podem ser usadas individualmente como tipos de construções condicionais múltiplas.

Função: `not` condição

Esta função testa a falsidade da *condição*. Retorna `t` se a *condição* for `nil` e `nil` caso contrário. A função `not` é idêntica a `null`, e recomendamos usar o nome `null` se você estiver testando uma lista vazia.

Formulário especial: `e` condições...

O formulário especial testa se todas as *condições* são verdadeiras. Funciona avaliando as *condições* uma a uma na ordem escrita.

Se qualquer uma das *condições* for avaliada como `nil`, o resultado do `and` deve ser independente das *condições* `nil` restantes; então retorna imediatamente, ignorando as *condições* restantes. `and nil`

Se todas as *condições* forem diferentes `nil`, o valor da última delas se tornará o valor da `and` forma. Apenas `(and)`, sem *condições*, retorna `t`, apropriado porque todas as *condições* acabaram não sendo `nil`. (Pense nisso; qual deles não fez?)

Aqui está um exemplo. A primeira condição retorna o inteiro 1, que não é `nil`. Da mesma forma, a segunda condição retorna o inteiro 2, que não é `nil`. A terceira condição é `nil`, portanto, a condição restante nunca é avaliada.

```
(e (impressão 1) (impressão 2) nil (impressão 3))
- | 1
- | 2
⇒ nada
```

Aqui está um exemplo mais realista de uso `and`:

```
(if (e (consp foo) (eq (car foo) 'x))
    (mensagem "foo é uma lista começando com x"))
```

Observe que `(car foo)` não é executado se `(consp foo)` retorna `nil`, evitando assim um erro. `and` expressões também podem ser escritas usando `if` ou `cond`. Veja como:

```
(e arg1 arg2 arg3 )
≡
(se arg1 (se arg2 arg3 ))
≡
(cond ( arg1 (cond ( arg2 arg3 )))))
```

Formulário especial: ou condições...

O formulário especial testa se pelo menos uma das *condições* é verdadeira. Funciona avaliando todas as *condições* uma a uma na ordem escrita.

Se qualquer uma das *condições* for avaliada como um não nilvalor, o resultado do or deve ser não-nil; então or retorna imediatamente, ignorando as *condições* restantes. O valor que ele retorna é o não nilvalor da condição avaliada.

Se todas as *condições* ocorrerem nil, a or expressão retornará nil. Apenas (or), sem *condições*, retorna nil, adequado porque todas as *condições* acabaram nil. (Pense nisso; qual deles não fez?)

Por exemplo, esta expressão testa se x é nil ou o inteiro zero:

```
(ou (eq x nil) (eq x 0))
```

Como a and construção, or pode ser escrito em termos de cond. Por exemplo:

```
(ou arg1 arg2 arg3 )
≡
(cond ( arg1 )
      ( arg2 )
      ( arg3 ))
```

Você quase poderia escrever or em termos de if, mas não exatamente:

```
(se arg1 arg1
  (se arg2 arg2
    arg3 ))
```

Isso não é completamente equivalente porque pode avaliar *arg1* ou *arg2* duas vezes. Por outro lado, nunca avalia qualquer argumento mais de uma vez. (or *arg1 arg2 arg3*)

Função: xor condição1 condição2

Esta função retorna o booleano exclusivo-or de *condition1* e *condition2*. Ou seja, xor retorna nil se ambos os argumentos forem nil, ou ambos forem não-nil. Caso contrário, ele retorna o valor desse argumento que não é nil.

Observe que, ao contrário de or, ambos os argumentos são sempre avaliados.

Próximo: [Condicional de correspondência de padrões](#), Anterior: [Condicionais](#), Acima:

[Estruturas de controle](#) [Conteúdo][Índice]