

Próximo:[Elementos da lista](#), Anterior:[Células Contrás](#), Acima:[Listas](#) [[Conteúdo](#)][[Índice](#)]

## 5.2 Predicados em Listas

Os predicados a seguir testam se um objeto Lisp é um átomo, se é uma célula cons ou uma lista ou se é o objeto distinto nil. (Muitos desses predicados podem ser definidos em termos dos outros, mas são usados com tanta frequência que vale a pena tê-los.)

### Função: *objeto consp*

Esta função retorna tse o *objeto* for uma célula contra, nil caso contrário. nil não é uma célula de contras, embora *seja* uma lista.

### Função: *objeto átomo*

Esta função retorna tse o *objeto* for um átomo, nil caso contrário. Todos os objetos, exceto as células contra, são átomos. O símbolo nil é um átomo e também é uma lista; é o único objeto Lisp que é ambos.

```
( objeto átomo ) ≡ ( não ( objeto consp ) )
```

### Função: *objeto listp*

Esta função retorna tse o *objeto* for uma célula cons ou nil. Caso contrário, ele retorna nil.

```
(listp '(1))
⇒ t
(listp '())
⇒ t
```

### Função: *objeto nlistp*

Esta função é o oposto de listp: ela retorna tse o *objeto* não for uma lista. Caso contrário, ele retorna nil.

```
( objeto listp ) ≡ ( não ( objeto nlistp ) )
```

### Função: *objeto nulo*

Esta função retorna tse o *objeto* for nile retorna nil caso contrário. Esta função é idêntica a not, mas por uma questão de clareza usamos null quando o *objeto* é considerado uma lista e not quando é considerado um valor de verdade (veja notem [Combinando Condições](#) ).

```
(null '(1))
⇒ nada
(nulo '())
⇒ t
```

### Função: *objeto lista-p adequada*

Esta função retorna o comprimento do *objeto* se for uma lista apropriada, nil caso contrário (veja [Cons Cells](#)). Além de satisfazer `listp`, uma lista adequada não é circular nem pontilhada.

```
(lista-própria-p '(abc))  
⇒ 3  
(lista-própria-p '(ab. c))  
⇒ nada
```

Próximo: [Elementos da lista](#), Anterior: [Células Contrás](#), Acima: [Listas](#) [[Conteúdo](#)][[Índice](#)]