Anterior:Predicados de igualdade, Acima:Tipos de dados Lisp [Conteúdo][Índice]

2.9 Mutabilidade

Alguns objetos Lisp nunca devem mudar. Por exemplo, a expressão Lisp `"aaa"`produz uma string, mas você não deve alterar seu conteúdo. E alguns objetos não podem ser alterados; por exemplo, embora você possa criar um novo número calculando um, Lisp não fornece nenhuma operação para alterar o valor de um número existente.

Outros objetos Lisp são *mutáveis* : é seguro alterar seus valores por meio de operações destrutivas envolvendo efeitos colaterais. Por exemplo, um marcador existente pode ser alterado movendo o marcador para apontar para outro lugar.

Embora os números nunca mudem e todos os marcadores sejam mutáveis, alguns tipos têm membros, alguns dos quais são mutáveis e outros não. Esses tipos incluem conses, vetores e strings. Por exemplo, embora `"cons"e (symbol-name 'cons)`ambos produzam strings que não devem ser alteradas, `(copy-sequence "cons")e (make-string 3 ?a)`ambos produzem strings mutáveis que podem ser alteradas por meio de chamadas posteriores para `aset`.

Um objeto mutável deixa de ser mutável se fizer parte de uma expressão avaliada. Por exemplo:

```
(let* ((x (lista 0,5)))
       (y (eval (lista 'cotação x)))))
  (setcar x 1,5) ;; O programa não deve fazer isso.
  e)
```

Embora a lista `(0.5)`fosse mutável quando foi criada, ela não deveria ter sido alterada via `setcar`porque foi dada a `eval`. O inverso não ocorre: um objeto que não deve ser alterado nunca se torna mutável depois.

Se um programa tenta alterar objetos que não devem ser alterados, o comportamento resultante é indefinido: o interpretador Lisp pode sinalizar um erro, pode travar ou se comportar de forma imprevisível de outras maneiras. [2]

Quando constantes semelhantes ocorrem como partes de um programa, o interpretador Lisp pode economizar tempo ou espaço reutilizando constantes existentes ou seus componentes. Por exemplo, `(eq "abc" "abc")`retorna `t`se o interpretador cria apenas uma instância da string literal `"abc"`e retorna `nil`se cria duas instâncias. Os programas Lisp devem ser escritos para que funcionem independentemente de esta otimização estar em uso.

Notas de rodapé

(2)

Este é o comportamento especificado para linguagens como Common Lisp e C para constantes, e isso difere de linguagens como JavaScript e Python, onde um interpretador é necessário para sinalizar um erro se um programa tentar alterar um objeto imutável. Idealmente, o interpretador Emacs Lisp evoluirá na última direção.

Anterior:Predicados de igualdade, Acima:Tipos de dados Lisp [Conteúdo][Índice]