

Próximo:[Expansão repetida](#), Anterior:[Vars locais surpreendentes](#), Acima:[Problemas com macros](#) [[Conteúdo](#)][[Índice](#)]

14.5.4 Avaliando Argumentos Macro em Expansão

Outro problema pode acontecer se a própria definição de macro avaliar qualquer uma das expressões de argumento de macro, como por meio de chamada `eval` (consulte [Eval](#)). Se o argumento deve se referir às variáveis do usuário, você pode ter problemas se o usuário usar uma variável com o mesmo nome de um dos argumentos da macro. Dentro do corpo da macro, a ligação do argumento da macro é a ligação mais local dessa variável, portanto, quaisquer referências dentro do formulário que está sendo avaliado se referem a ela. Aqui está um exemplo:

```
(defmacro foo (a)
  (list 'setq (eval a) t))
(setq x 'b)
(foo x) → (setq bt)
      ⇒ t ; e b foi definido.
;; mas
(setq a 'c)
(foo a) → (setq em)
      ⇒ t ; mas este conjunto a, não c.
```

Faz diferença se a variável do usuário é nomeada `aou` `x`, porque `a` entra em conflito com a variável do argumento da macro `a`.

Outro problema em chamar `eval` numa definição de macro é que ela provavelmente não fará o que você pretende em um programa compilado. O compilador de bytes executa definições de macro enquanto compila o programa, quando os próprios cálculos do programa (que você pode ter desejado acessar com `eval`) não ocorrem e suas ligações de variáveis locais não existem.

Para evitar esses problemas, **não avalie uma expressão de argumento ao calcular a expansão da macro**. Em vez disso, substitua a expressão na expansão da macro, para que seu valor seja calculado como parte da execução da expansão. É assim que os outros exemplos neste capítulo funcionam.

Próximo:[Expansão repetida](#), Anterior:[Vars locais surpreendentes](#), Acima:[Problemas com macros](#) [[Conteúdo](#)][[Índice](#)]