Linguagens de Programação

Fabio Mascarenhas - 2017.2

http://www.dcc.ufrj.br/~fabiom/lp

Chamada por valor vs chamada por nome

- O interpretador de *fun* está fazendo chamada por valor
- Mudá-lo para fazer chamada por nome é simples no entanto!
  - Apenas precisamos mudar a avaliação das chamadas de função para passar *expressões* ao invés de valores para a substituição
  - A substituição fica até mais simples! Não é preciso mais converter os valores primitivos em expressões para plugá-los no corpo de função
- Para não complicar o parser vamos adotar uma convenção léxica: parâmetros que começam com _ serão por nome, e os outros por valor

Nomes locais: let

- Vamos introduzir uma nova expressão em *fun*, para dar nomes para expressões

```
exp  : ...
     | LET ID '=' exp IN exp END
```

```
case class Let(nome: String, exp: Exp, corpo: Exp) extends Exp
```

- O `let` é parecido com o *val* de Scala; dentro do *corpo* do `let` o *nome* é associado ao valor de *exp*

- Podemos dar a semântica de *fun* com `let` via substituição também, mas a substiuição fica mais complicada

Substituição com *let*

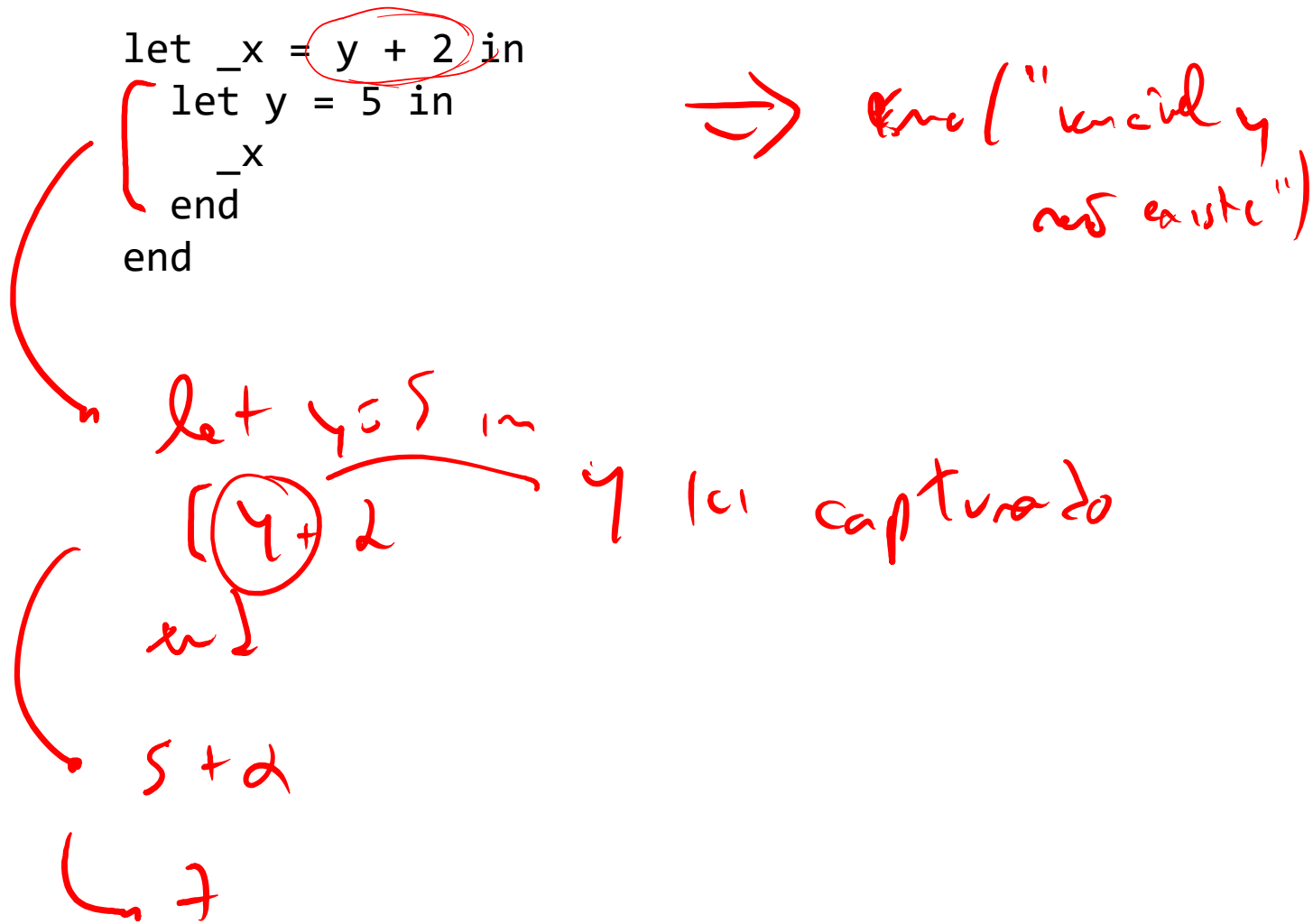
- Para substituir um identificador *x* em uma expressão *e* por um valor *v*, troque todas as **instâncias livres** de *x* em *e* por *v*

- Ou seja, a substituição do identificador `x` não “entra” em um termo `let x = ...`

- Essa definição funciona muito bem para substituição de valores (call-by-value), mas o que acontece com substituição de termos?

Substituição CBN

- Vamos avaliar essa expressão:



Substituição CBN

- Vamos avaliar essa expressão:

```
let _x = y + 2 in
  let y = 5 in
    _x
  end
end
```

- O resultado é 7!

- O termo pelo qual estamos substituindo não pode ter variáveis livres!