

Linguagens de Programação (CK0115)

Lista de Exercícios II (Capítulo 2)

Fernanda Costa de Sousa - 485404

1. Free and bound identifiers.

Bound. Pois, P está de certa forma presa ao valor do Proc. Quando traduzido para a linguagem núcleo, fica da seguinte forma:

```
local P in
  P = Proc { $ X }
  local T in
    T=X>0
    if T then {P X-1} end
  end
end
end
```

2. Contextual environment.

Supondo que o ambiente de chamada contenha {A -> 10, B ->x1}, {MulByN 10, x1}. Quando o corpo é executado, um mapeamento N -> 3 é adicionado ao ambiente.

Porque essa etapa é necessária?

Porque N pode ser uma referência que pode estar pendente.

OBS: Professor, no livro é chamado de *dangling references*, não sei se em português faz o mesmo sentido.

Em particular, poderia ter uma referência N -> 3 em algum lugar do ambiente?

Sim, pois os resultados mudam em cada execução.

E se isso não for suficiente para garantir que o identificador N mapeie para 3?

Dê um exemplo onde N não existe no ambiente no momento da chamada.

```
local A=10 B in
  local MulByN in
    local N=3 in
      proc {MulByN X ?Y}
        Y=N*X
      end
    end
  end

  {MulByN A B}
  {Browse B}
  % Mostra 30
```

```

    {Browse N}
    % N não existe

end
end

```

Dê um exemplo em que N existe, mas está vinculado a um valor diferente de 3.

```

local A=10 B in
  local MulByN in
    local N=3 in
      proc {MulByN X ? Y}
        Y=N*X
      end
    end
    local N=10 in
      {MulByN A B}
      {Browse B}
      {Browse N}
    end
  end
end
end

```

3. Functions and procedures.

4. The if and case statements.

(a)

```

if <x> then <s>1 else <s>2 end
case <x>
of true then <s>1
[ ] false then <s>2
end

```

(b)

```

case <x> of <label>(<feat>1:<x>1 ... <feat>n:<x>n) then <s>1 else <s>2
end

if {Label <x>}==<label> then
  if {Arity <x>}==[<feat>1 ... <feat>n] then
    local <x>1=<x>.<feat>1 ... <x>n=<x>.<feat>n in
      <s>1
    end
  end
end

```

```
end  
else <s>2 end  
else <s>2 end
```

5. The case Statement.

```
proc {Test X}  
case X  
of a|Z then {Browse 'case'(1)}  
[ ] f(a) then {Browse 'case'(2)}  
[ ] Y|Z andthen Y==Z then {Browse 'case'(3)}  
[ ] Y|Z then {Browse 'case'(4)}  
[ ] f(Y) then {Browse 'case'(5)}  
else {Browse 'case'(6)} end  
end
```

Predições:

{Test [b c a]}
Nesse teste, o 4

{Test f(b(3))}
Nesse teste, o 5

{Test f(a)}
Nesse teste, o 2

{Test f(a(3))}
Nesse teste, o 5

{Test f(d)}
Nesse teste, o 5

{Test [a b c]}
Nesse teste, o 1

{Test [c a b]}
Nesse teste, o 4

{Test a|a}
Nesse teste, o 1

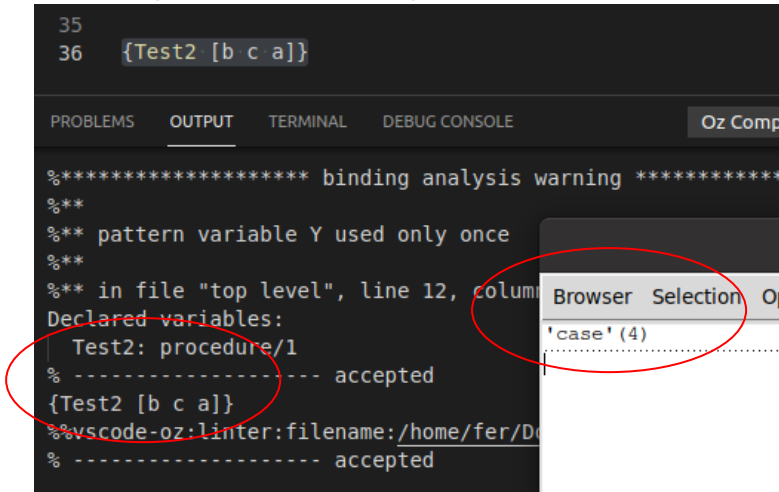
{Test |(a b c)}
Nesse teste entra no else, o 6.

```
declare
proc {Test2 X}
  case X of '|' (a Z) then {Browse 'case'(1)}
  else
    case X of f(a) then {Browse 'case'(2)}
    else
      case X of '|' (Y Z) then
        if Y==Z then {Browse 'case'(3)}
        else {Browse 'case'(4)}
        end
      else
        case X of f(Y) then {Browse 'case'(5)}
        else
          {Browse 'case'(6)}
          end
        end
      end
    end
  end
end
end
```

```
% Para fazer os testes no exemplos dados,
% testei feed line em cada linha
```

```
{Test2 [b c a]}
{Test2 f(b(3))}
{Test2 f(a)}
{Test2 f(a(3))}
{Test2 f(d)}
{Test2 [a b c]}
{Test2 [c a b]}
{Test2 a|a}
{Test2 '|' (a b c)}
```

um exemplo da saída com o primeiro teste no meu ambiente:



The screenshot shows a VS Code terminal window with the following content:

```
35
36 {Test2 [b c a]}
```

The terminal output includes:

```
%***** binding analysis warning *****
%**
%** pattern variable Y used only once
%**
%** in file "top level", line 12, column 12, warning:
Declared variables:
Test2: procedure/1
% ----- accepted
{Test2 [b c a]}
%vscode-oz:lint:filename:/home/fer/D
% ----- accepted
```

A "Browser Selection" dialog box is open, showing a list with the entry "'case' (4)".

6. The case statement again.

declare X Y {Test f(X b Y)}

Acredito que não dá certo e nada vai ser mostrado no Browser.

declare X Y {Test f(a Y d)}

Entra no else e no Browse é mostrado 'case'(2).

declare X Y {Test f(X Y d)}

Esse também não dá certo. Nada apareceria no Browser.

Testando o exemplo abaixo, como sugerido no livro, aparece no Browser o 'case' (2).

```
declare X Y
if f(X Y d)==f(a Y c) then {Browse 'case'(1)}
else {Browse 'case'(2)} end
```

Me parece que se houver uma variável não ligada, o programa pula por um momento e verifica a próxima, mas no case, ele verifica no começo e se tiver uma variável não associada, ele pula. Parece que já bloqueia sem verificar.

7. Lexically scoped closures.

A minha predição sobre o que acontece quando se executa um feed line sobre {Browse [{Max3 4} {Max5 4}]}, é:

{Max3 4} - Neste seria 4

{Max5 4} - Neste seria 5

então ao executar o comando:

{Browse [{Max3 4} {Max5 4}]} - Seria 4 5

8. Control abstraction.

(a)

Parece que o resultado é igual para os dois. Quando BP1 é falso, o BP2 não é chamado. Por exemplo, if EXPRESSION1 then EXPRESSION2 else false end. EXPRESSION2 não é chamada e termina, end.

(b)

```
declare
fun {OrElse BP1 BP2}
  if {BP1} then true else {BP2} end
end
{Browse {OrElse fun {$} false end fun {$} 'Isso é um teste!' end}}
fun {Test EX1 EX2}
  EX1 orelse EX2
end
{Browse {Test false 'Isso é um teste!'}}
```

9. Tail recursion

10. Expansion into kernel syntax.

11. Mutual recursion.

12. Exceptions with a finally clause.

Seguindo as orientações da questão, executando o <s> 1 e, logo depois, executando <s> 2 , se ocorrer uma exceção, a exceção precisa ser lançada. Fica da seguinte forma:

```
try <s>1 finally <s>2 end

local Boolean = {NewCell false} Exc in
  try
    <s>1
  catch X then
    Exc = X
    Boolean := true
  end
  <s>2
  if Boolean then raise Exc end end
end
```