例外処理を追加したLua処理系の実装

河原 悟

情報特別演習 報告会

January 12, 2016

情報特別演習 報告会

概要

実装

- parse
  - PEG について
- eval
  - CPS について

実行例

まとめ・課題

- プログラミング言語 Lua インタプリターの実装
  - 軽量なスクリプト言語
  - ゲーム開発から組み込み (FreeBSD、LuaLaTeX など) まで
  - 関数がファーストクラスで扱える
- 既存の処理系にはない例外処理機能の追加

Code 1: try-catch in Lua Sample

```
try -- try block
  for i = 1, 10 do
    if i > 9 then
      throw() -- goto catch block
    end

    print(i)
  end
catch --catch block
  print("thrown")
end
```

情報特別演習 報告会

概要

実装

- parse
  - PEG について
- eval
  - CPS について

実行例

まとめ・課題

- MoonScript で実装[*1]
  CoffeeScript に類似した簡潔な構文、Lua へ変換可能

Code 2: MoonScript Sample

```
f = (fun, e, n)->
  for i = 1, n
    fun e

f print, "hello", 10
-- print "hello" 10 times.
```

- 720 行程度
- CPS を用いた例外処理の実装

[*1] version 0.4.02(https://github.com/leafo/moonscript/tree/v0.4.0)

情報特別演習 報告会

概要

実装

- parse
  - PEG について
- eval
  - CPS について

実行例

まとめ・課題

- Parsing Expression Grammar の略
- 構文解析、字句解析を同時に行える
- Lua 向け PEG モジュールである LPeg[*2] を用いた[*3]

```
........
  Colonfunc: P':' * V'Space' * CV'Name' * V'Space' * V'Callargs' / lbl_tbl 'colonfunc', 'func', 'args'
  Call: V'Callargs' + V'Colonfunc' -- * V'Space' * V'Callargs' / lbl_tbl'colonfunc'
  Prefix: spaces(P'(', V'Exp', P')') + CV'Name'
  Suffix: V'Call' + V'Index'
  Var: (V'Prefix' * ast(V'Space' * V'Suffix' * #(V'Space' * V'Suffix')) * V'Space' * V'Index' + CV'Name') / gen_tblaccess
  Funcall: V'Prefix' * ast(V'Space' * V'Suffix' * #(V'Space' * V'Suffix')) / gen_tblaccess * V'Space' * V'Call' / lbl_tbl'funcall'
  Funcname: CV'Name' * ast(V'Space' * P'.' * V'Space' * CV'Name') * opt(V'Space' * P':' * V'Space' * CV'Name')

  Callargs:
    Ct(P'(' * V'Space' * opt(V'Explist' * V'Space') * P')' +
    (V'Tableconstructor' + V'String'))

  Funcdef: K'function' * V'Space' * V'Funcbody' * V'Space' * K'end' / lbl_tbl('annonymousfuncdef', 'args', 'body')
  Funcbody: spaces P'(', (opt(V'Parlist') / lbl_tbl'args'), P')', CtV'Block'

  Parlist: (V'Namelist' * opt(V'Space' * P',' * V'Space' * CP'...') + CP'...')
  Tableconstructor: P'{' * V'Space' * (opt(V'fieldlist' * V'Space') / lbl_tbl'constructor') * P'}'
  fieldlist: V'Field' * ast(V'Space' * V'Fieldsep' * V'Space' * V'Field') * opt(V'Space' * V'Fieldsep')
  Field:
    Ct(spaces(P'[', CtV'Exp', P']', P'=', V'Exp')) +
    Ct(spaces(CV'Name', P'=', V'Exp')) + V'Exp'
}
......
```

[*2] http://www.inf.puc-rio.br/~roberto/lpeg/

[*3] 実際には LPeg の Lua 実装である LuLPeg を用いた。https://github.com/pygy/LuLPeg

情報特別演習 報告会

- 継続渡し形式、Continuation Passing Style の略
- 例外発生時に継続を破棄することで例外処理を実現

```
......
    funstack\pushcresume -> eval def.body, nenv, k0
    ret = funstack\pop!

    ret[2] and unpack ret[2] or nil

  k nenv[def.name]

eval_args = (arglist, env, k0, k) ->
  unless arglist[1]
    k arglist
  else
    head = remove arglist, 1
    argf = -> eval_exp head, env, k0, (x) ->
      eval_args arglist, env, k0, (y) ->
        _insert y, 1, x
        k y

    unless arglist[1]
      if is_funcall head
        eval_funcall head[1], head[2], env, k0, (x) ->
          eval_args arglist, env, k0, (y) ->
            k with y do for i = 1, #x do _insert y, x[i]
      elseif head == "..."
        eval_args env[head], env, k0, (t) -> k t
......
```

例: 整数リストを受け取り、総乗を返す関数

Code 3: mul_list.moon

```
mul_list = (t) ->
  h = table.remove t, 1
  switch h
    when nil then 1
    else h * mul_list t
print mul_list {1, 2, 3} -- 6
```

⇒

Code 4: mul_list_cps.moon

```
mul_list_cps = (t, k = (x) -> x) ->
  h = table.remove t, 1
  switch h
    when nil then k 1
    else mul_list_cps t, (x) -> k x * h

print mul_list_cps {1, 2, 3} -- 6
```

Code 5: pseudo mul_list

```
  f [1,2,3]
= 1 * f [2, 3]
= 1 * (2 * f [3])
= 1 * (2 * (3 * f []))
= 1 * (2 * (3 * 1))
= 6
```

⇒

Code 6: pseudo mul_list_cps

```
  f' [1,2,3], ((x) -> x)
= f' [2,3],   ((x) -> 1 * x)
= f' [3],     ((x) -> 1 * 2 * x)
= f' [],      ((x) -> 1 * 2 * 3 * x)
= ((x) -> 1 * 2 * 3 * x) 1
= 1 * 2 * 3 * 1
= 6
```

関数に処理の内容を渡し、
これを継続

その関数が呼ばれるまで処理はしない

↓

この関数(継続)を捨てることで例外処理を実現

Code 3,4 の拡張: 整数リストを受け取り、総乗を返すが、0 が含まれている時に例外を発生する関数

Code 7: mul_list_excep.moon

```
mul_list_excep = (t) ->
  h = table.remove t, 1
  switch h
    when 0
      0, print "ZeroMultiple Error"
      -- error handling, but return 0
    when nil then 1
    else h * mul_list_excep t

print mul_list_excep {1, 2, 3}
-- 6
print mul_list_excep {1, 2, 3, 0}
-- ZeroMultiple Error
-- 0
```

⇒

Code 8: mul_list_excep_cps.moon

```
mul_list_excep_cps = (t, k = (x) -> x) ->
  h = table.remove t, 1
  switch h
    when 0 then print "ZeroMultiple Error"
    -- dump `k`
    when nil then k 1
    else mul_list_excep_cps t,
      (x) -> k x * h

print mul_list_excep_cps {1, 2, 3}
-- 6
print mul_list_excep_cps {1, 2, 3, 0}
-- ZeroMultiple Error
```

情報特別演習 報告会

概要

実装

- parse
  - PEG について
- eval
  - CPS について

実行例

まとめ・課題

- 対話型電卓
- ゼロ除算、パースエラーなどで例外処理を使用
- ソース全体はこちら[*4]

```
......
try -- catch zero division
  for i = 1, #x, 2 do
    local op = x[i]
    local n2 = eval(x[i + 1])

    if op == "/" and n2 == 0 then throw() end

    n1 = op_table[op](n1, n2)
  end
  return n1
catch
  print "ZeroDivision Error"
end
......
```

```
......
try -- parsed or not
  local t = parse(l)
  if not t then throw() end

  local e = eval(t)
  if e then
    print("ANS " .. tostring(e))
  end
catch
  print "!!parse error!!"
end
......
```

[*4] https://gist.github.com/Nymphium/aedeb3fab71fe7f1f412

情報特別演習 報告会

概要

実装

- parse
  - PEG について
- eval
  - CPS について

実行例

まとめ・課題

- まとめ
  - Lua サブセットのインタプリターの実装
    Lua のコア部分 (`while`、`for`、`if`、`function`、table など) の実装完了
  - 拡張
    - 処理系を CPS に変換
    - `try-catch`構文の追加
  - テスト･実行例
- 課題
  - 実行速度の改善 $\Rightarrow$ コンパイラの作成
  - Lua への完全対応