

Overview of the Contua programming language.

Jacek Olczyk

Feb 2019

1 Introduction

Contua is a strongly typed functional programming language. It is conceived on a basis of the idea that the language should force the programmer to write all the functions continuation-style. It is not immediately clear what measures needs to be taken to disallow regular functions. I needed to come up with some conventions to achieve that result.

2 The syntax

Since the language will be pretty complicated in the 'backend', I tried to keep the syntax as simple as possible. In this grammar I omitted the whitespace rules.

```
program = { typeDecl }, { funDecl }
funDecl = type, '::', id, { id }, "=", expr
typeDecl = 'type', ID, { id }, "=", typeCtor, { "|", typeCtor }
type = basicType | (type, { '->', type }) | "(", type, ")"
basicType = typeCtor | id | "[", id, "]"
typeCtor = ID, { type }
expr = id | ID | expr, "+", expr | expr, "-", expr | expr, "*", expr | expr, "-", expr |
      | "(", expr, ")" | expr, expr | 'fn', { id }, ".", expr | listExpr |
      | expr, 'where', funDecl, { 'and', funDecl } | 'let', funDecl, 'in', expr |
      | 'match', expr, 'with', { "|", expr, '=>', expr } |
      | 'if', bexpr, 'then', expr, 'else', expr
bexpr = id | bexpr, 'and', bexpr | bexpr, 'or', bexpr | 'or', bexpr | expr, '<=', expr
listExpr = [expr, {"", expr}] | expr, ":", listExpr | listExpr, "++", listExpr
```

3 Forcing continuation style