Haskell

un'implementazione in SML

Cicio Ionut

Sommario

Haskell (context-free grammar)

Hello darkness my old friend [1]

$$M, N \Coloneqq \text{integer} \mid \text{float} \mid \text{string} \mid$$

Parser

- comments:
  - "–" single line
  - "{- -}" multiple line
- keywords:
  - case
  - class
  - data
  - deriving
  - do
  - else
  - if
  - import
  - in
  - infix
  - infixl
  - infixr
  - instance
  - let
  - of
  - module
  - newtype
  - then
  - type
  - where
- strings:
  - "abc" unicode string (basically a list of chars)
  - 'a' single character
  - "multi
    line
    string" multiline string
- numbers:
  - 1 integer
  - 1.0 floating point
- enumerations:

  - [1..10] 1, …, 10
  - [100..] 100, 101, 102, …
  - [110..100] ∅
  - [0, −1 ..] negative integers?
  - [−100..-110] syntax error, should be [−100.. −110]
  - [1,3..100], [−1,3..100] list from 1 to 100 by 2,-1 to 100 by 4
  - each value in the Enum class can be used?? What is a class?

- lists & tuples:
  - [] empty list
  - [1,2,3] list of three numbers
  - 1 : 2 : 3 : [] “cons”(:) and “nil”([])
  - ‘a’ : ‘b’ : ‘c’ : [] same as “abc”
  - (head, tail, 3, ‘a’, “abc”) tuple of different elements

- “Layout” rule, braces and semi-colons ??????
  - basically python-like indentation for scopes, don’t even think about “;” and “{}”

- function definition
  - square x = x * x
  - square x = x2 where x2 = x * x

- let
  - square x = let x2 = x * x in x2

- case
  - TODO:
    - nesting, capture, matching order, guards

- class
  - TODO:
    - class + instance
    - overloading?
    - defaults

- data
  - algebraic data types
    - a.k.a. algebre induttive
    - constructors with arguments
    - type and constructor names
    - type variables
    - record syntax

- deriving????

- do
  - monads?????????????????

    - if and io
- let
    - deconstruction????????
- of (riguarda le classi?)
- module
    - yay!
    - imports???
- data
    - creates a new type
- type
    - just aliases another type, they can be used interchangeably
- newtype
    - basycally create a new type, but behaves exactly like another type

Semantica operazionale lazy static

Monad

Bibliografia

[1] J. Bailey, «Haskell Cheat Sheet». [Online]. Disponibile su: https://hackage.haskell.org/package/CheatSheet-1.5/src/CheatSheet.pdf

https://github.com/shwestrick/smlfmt https://smlhelp.github.io/book/docs/ TODO: smlnj TODO: millet