Gustavo Lopes Rodrigues, Lucas Santiago, Pedro Souza, Thiago Henriques

Uma Introdução a Haskell

Belo Horizonte

2020

Sumário

1 Introdução

Em 1930, Alonzo Church , matemático estadunidense apresentou o Cálculo Lambda, como parte da investigação dos fundamentos da matemática. O Cálculo Lambda é um sistema que estuda funções recursivas computáveis, e foi utilizada como base para as teorias e fundamentos matemáticos por trás do paradigma da Programação Funcional. Ele também pode ser considerado a primeira linguagem programação funcional, todavia, não foi projetada para ser executada em computadores, sendo apenas um modelo que descreve relações entre funções simples, permitindo criar funções mais complexas.

Com o passar dos anos, varias linguagens funcionais foram criadas, sendo alguns exemplos a linguagem LISP em 1955 e a ML no final da década de 70. Porém, não havia um padrão para as linguagens desse paradigma, e quando chegou a segunda metade da década de 80, havia uma necessidade para criar uma única linguagem, que englobasse as melhores práticas de projeto, além de implementar as técnicas funcionais que estavam em alta na época.

Em Setembro de 1987, aconteceu a conferência *Functional Programming Language and Computer Architecture* em Portland - Oregon, para que fosse discutido justamente tais questões.

Dessa conferência, surgiu as seguintes diretrizes:

- Ser viável para o ensino, pesquisa e aplicações, incluindo sistema de larga escala;
- Ser completamente descritiva via publicação no tocante à sua sintaxe e sua semântica;
- Não ser proprietária, tal que qualquer um pudesse implementá-la e distribuí-la;
- Basear-se em ideias que envolvessem o senso comum;
- Reduzir a diversidade desnecessária de outras linguagens funcionais.

Além disso, eles decidiram não começar do zero, e elegeu a linguagem Miranda, como base do projeto, por ser a mais amadurecida e bem projetada entre as demais. E foi a partir dessa linguagem, que nasceu Haskell.

2 Histórico sobre a linguagem, com sua cronologia

Como dito anteriormente, Haskell nasceu a partir da necessidade de criar uma linguagem funcional mais padronizada. Seu nome é uma homenagem ao matemático Haskell Brooks Curry, que contribuiu com teorias fundamentais para o paradigma.

A primeira versão foi definida em 1 de abril de 1990 e com o passar dos anos, a linguagem foi recebendo atualizações(1.1 em 1991, 1.2 em 1992, 1.3 em 1996 e 1.4 em 1997). Em janeiro de 1999, foi juntado todos as versões anteriores para fazer o Haskell 98, que especifíca uma versão mínima, estável e portável da linguagem e das bibliotecas para ensino. A linguagem mesmo assim continuo evoluindo, recebendo novas atualizações em 2003, 2006 e por fim , a última atualização em 2010, com o Haskell 2010.

3 Paradigma a que pertence

O paradigma oferece e determina a visão que o programador possui sobre a estruturação e a execução do programa. Um exemplo bem famoso de paradigma é o POO (Programação orientada a objetos). Já a Programação funcional é um paradigma que descreve uma expressão matemática a ser avaliada, mapeando dos valores de entradas nos valores de retorno, por meio de funções. Em outras palavras: a programação funcional só funciona em cima de funções.

Eis mais algumas características do paradigma

- Dados imutáveis e são evitados estados.
- Não existe efeitos colaterais
- "Funções puras"(sem efeitos colaterais)
- Cálculo Lambda

4 Características mais marcantes da linguagem

- Como dito anteriormente, a linguagem só faz utilização de funções e funções dentro de funções. Por isso Haskell é descrito como puramente funcional.
- Haskell possui uma sintaxe simples, elegante e concisa. Como resultado, programas em Haskell possuem poucas linhas.
- Além disso, a linguagem usa avaliação preguiçosa(Lazy evaluation), que é uma técnica para atrasar a computação até um ponto em que o resultado da computação é considerado necessário.
- Tipagem estática: Verificação dos tipos usados em dados e variáveis para garantir que sempre está sendo usado um tipo que é esperado em todas as situações.
- função de ordem superior: Função que tem como argumento uma outra função, ou que produz uma função como resultado.

5 Linguagem similares ou confrontantes

- Prolog
- LISP
- Scheme
- ML
- Miranda
- Elixir

6 Exemplo(s) de programa(s)

7 Estudo de Caso, sobre o(s) tema(s)

8 Considerações finais

Apêndices

APÊNDICE A – Teste