PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Graduação em Ciência da Computação

Giulia Chiucchi

UMA BREVE INTRODUÇÃO AO HASKELL

Belo Horizonte
2021

SUMÁRIO

1 INTRODUÇÃO

O Cálculo Lambda foi um trabalho apresentado em 1930 por Alonzo Church, ele é um sistema utilizado para estudos de funções recursivas e foi utilizado como fundamentação teórica base para o desenvolvimento do paradigma funcional de programação.

O Haskell é uma linguagem que foi idealizada 1987 por três indivíduos com o intuito de ajudar pesquisadores do paradigma funcional a terem uma linguagem funcional comum e poderem expandir sua linha de pesquisa trocando linhas de código entre si já que até então não havia nenhum padrão mesmo que algumas linguagens desse paradigma já existissem.

Sendo assim cinco reuniões aconteceram para decidir os detalhes e objetivos dessa nova linguagem a ser desenvolvida, na primeira reunião pensaram em utilizar uma linguagem funcional que estava em alta na época, Miranda de David Turner, porém o mesmo não autorizou a sua utilização e o comitê acabou decidindo em desenvolvê-la do zero. Em uma dessas reuniões antes da publicação de sua versão 1.0 decidiram finalmente o nome dessa linguagem, resolveram homenagear o matemático Haskell Curry que possuía trabalhos na área de lógica.

2 HISTÓRIA DA LINGUAGEM

O primeiro relatório da versão 1.0 da linguagem Haskell foi publicado em 1 de Abril de 1990, 125 páginas editadas por Hudak e Wadler, tal documento relatava a motivação da criação da linguagem, a sua natureza e o processo de sua criação.

Esse foi o primeiro passo público dado pelo comitê para a disseminação da linguagem que hoje conhecemos como Haskell, porém a idealização da linguagem teve seu início em 1987 após um encontro entre Peyton Jones, Wadler e Hudak em Yale que deu origem a primeira reunião na FPCA, afim de gerar interesse no desenvolvimento de uma nova linguagem funcional comum. Nessa reunião os participantes entraram em consenso de que seria mais fácil aprimorar uma linguagem já existente, a melhor escolha era a Miranda de David Turner já que era a mais madura que utilizava do conceito de avaliação preguiçosa, entretanto Turner não autorizou sua utilização como ponto de partida e também rejeitou o convite ná época para participar do comitê IFPI W.G. 2.8.

A partir desse momento houveram cinco reuniões presenciais para discutir pontos chaves do desenvolvimento da linguagem antes da publicação do seu primeiro relatório, além de inúmeros e-mails trocados pela lista fplangc@cs.ucl.ac.uk, como o nome Haskell só foi decidido em uma dessas reuniões, eles se chamavam antes de "FPLang Commite".

Após o nascimento da linguagem tivemos alguns marcos importantes: em 1991 e 1992 tivemos publicações dos relatórios das versões 1.1 e 1.2 do Haskell, 92 foi um ano bem agitado pois além da nova versão, no mês de maio Hudak e Fasel publicaram a "Gentle Introduction to Haskell"dando ao Haskell mais credibilidade e notoriedade dentro da comunidade e também foi o ano de criação do GHC (Glorious Glasgow Compiler), compilador mais utilizado pela linguagem até hoje.

Em 1994, a linguagem ganhou ainda mais visibilidade após John Peterson registrar o domínio haskell.org e organizar um servidor e o site em Yale, servidor qual é mantido até hoje por estudantes da universidade. Após isso, os relatórios das versões 1.3, 1.4 e o de 98 foram publicados, este último por sua vez teve uma versão revisada que fora publicada pela Universidade de Cambridge no formato de livro em 2002 sob o acordo de que todo o texto estaria disponível gratuitamente online para que todos pudessem ter acesso, mais um avanço para a comunidade da linguagem e que também gerou o debate de liberdade e propriedade intelectual.

Já no século atual a linguagem foi ainda mais disseminada na comunidade, em 2005 houve o Haskell Workshop sediado na Estônia, evento que foi o ponto de partida para a criação de várias extensões da linguagem, porém com isso foi decidido que a melhor forma de conquistar progresso para evoluir uma linguagem seria em pequenos passos, revisando um pequeno número de mudanças por vez. Com esse pensamento a primeira revisão foi lançada, Haskell 2010, versão que teve a maior quantidade de mudanças comparadas ao Haskell 98. Atualmente, o GHC adere ao padrão estabelecido pelo Haskell 2010, estão

em processo de desenvolvimento do novo padrão Haskell 2020 porém o projeto está paralisado e sem data de lançamento. Em novembro de 2020 foi fundado a Fundação Haskell, recebendo patrocínio de várias empresas importantes.

Algumas mudanças entre suas versões:
1.0 até 1.4 - apresentação da fundamentação base da linguagem, classes de tipo (constructor classes), classes, tipos e funções predefinidas, semânticas de I/O, tipos de dados algébricos foram extendidos.
Haskell 98 - nessa versão foi incentivado a criação de extensões e variações do haskell 98.
Haskell 2010 - as regras de inferência de tipos foram suavizadas, alguns problemas de sintaxe foram corrigidos, a interface de funções externas que permitia ligações com outras linguagens de programação.

3 PARADIGMA A QUE PERTENCE

O Haskell pertence ao paradigma funcional, esse paradigma consiste basicamente na aplicação de funções matemáticas utilizando de expressões ao invés de afirmações como é feito no paradigma imperativo, uma expressão deve produzir um valor enquanto a afirmação é executada para atribuição em uma variável.

O paradigma funcional tem sua fundamentação teórica no cálculo lambda desenvolvido por Alonzo Church na década de 30, esse é um sistema que fora criado para definições, aplicação e recursão de funções. Algumas características desse paradigma são:

- Fácil de entender;
- Breve e conciso;
- Sem core-dumps - linguagens fortemente tipadas eliminam erros bobos no tempo de compilação. Não é possível tratar um inteiro como um ponteiro, ou seguir um ponteiro nulo.;
- Reuso facilitado por conta do polimorfismo - o polimorfismo aumenta a possibilidade de reuso de código;
- Abstrações poderosas - são oferecidas na maioria das linguagens funcionais, a mais poderosa é a função de alta ordem que está presente no haskell. A abstração é importante pois te permite a criação de programas modulares de fácil manutenção.
- Gestão de memória embutida - as linguagens funcionais realizam a alocação de memória e a inicializa implicitamente, sendo recuperada automaticamente pelo garbage collector. Poupando o programador de fazer todo o gerenciamento a mão.
- Dados imutáveis - É possível declarar variáveis de diversos tipos porem uma variável não pode ter o seu valor alterado após ter sido definido uma vez.
- Funções Puras - são funções que produz a mesma saída para os mesmos argumentos e não tem efeitos colaterais, ou seja, não realizam nenhuma modificação além de retornar o valor.
- Recursão - como não é possível utilizar lações de repetição, quando necessário é usado a recursão, função que chama a si mesma.
- Funções de primeira classe - essas funções são tratadas como variáveis de primeira classe, podendo ser passado para outras funções como parâmetro, retornada de funções ou até armazenada em alguma estrutura de dados.

4 CARACTERÍSTICAS DA LINGUAGEM

O Haskell é uma das linguagens que mais aborda o conceito de programação funcional em relação a demais linguagens.

- Lazy evaluation - criar sequências infinitas porém sem estourar a memória. uma técnica para atrasar a computação até um ponto em que esse resultado é considerado necessário; uma expressão só é avaliada quando estiver vinculado a uma variável, e no Haskell somente os argumentos necessários são avaliadas;

- Funções puras - que não sofrem interferência de meios externos e evitam os chamados efeitos colaterais. o design da linguagem é centrado nas funções puras e dados imutáveis, são essas funcionalidades que produz uma codabilidade com erros minimizados;

- Memória segura - Haskell possui gerenciamento de memória automático, reduzindo problemas como buffer overflow, memory leaks, e outros bugs de memória;

- Garbage collection;

- Tipagem estática - todos os valores tem um tipo determinado na hora de compilar, haskell é fortemente tipado, os bugs são identificados rapidamente pelo seu checador de tipos.

- Riqueza de tipos - a linguagem possui suporte a uma ampla gama de tipos como, dados algébricos, polimorfismo paramétrico, e etc;

- Pureza - o design da linguagem é centrado nas funções puras e dados imutáveis, são essas funcionalidades que produz uma codabilidade com erros minimizados;

- Funções de ordem superior - são funções que aceitam outras funções como parâmetros e/ou retornam funções como resultado;

- Funções de primeira classe - funções podem ser usada como valores, atribuídas a uma variável.

5 LINGUAGENS RELACIONADAS

5.1 Standard ML

Linguagem pertencente ao paradigma funcional de propósito geral modular com inferência de tipos e alguns efeitos colaterais. Alguns pontos similares ao Haskell:

- Suporte para tipos de dados algébricos;
- Correspondência de padrões em valores primitivos e personalizados;
- Funções de ordem superior;
- Funções de primeira classe;
- Coletor de lixo;

Alguns diferenciais:

- Strict evaluation;
- Suas estruturas de dados são mutáveis;
- Características impuras;

Exemplo de código:

```
fun fib 0 = 0
    | fib 1 = 1
    | fib n = (fib (n - 1)) + (fib (n - 2))

val x = fib 8
```

5.2 Scala

Linguagem de programação multiparadigma sucessora de Funnel, uma linguagem baseada em join calculus. Alguns pontos similares ao Haskell:

- Correspondência de padrões em parâmetros de funções;
- É baseada no paradigma funcional;
- Imutabilidade de valores;
- Inferência de tipos;

Alguns diferenciais:

- Pode ser compilada na JVM por gerar um .class;
- Orientada a Objetos;

Exemplo de código:

```
def fib1(n: Long): Long = n match {
   case 0 | 1 => n
   case _ => fib1(n - 1) + fib1(n - 2)
}
```

5.3 Clojure

Clojure é um dialeto da linguagem de programação LISP seu propósito geral tem ênfase em programação funcional. Alguns pontos similares ao Haskell:

- Imutabilidade;
- É baseada no paradigma funcional;
- Funções de ordem superior;
- Funções de primeira classe;
- Tipagem estática está sendo introduzida

Alguns diferenciais:

- Usa a JVM;
- É uma linguagem dinâmica;

Exemplo de código:

```
(defn fib
  ([n]
     (fib [0 1] n))
  ([x, n]
     (if (< (count x) n)
       (fib (conj x (+ (last x) (nth x (- (count x) 2)))) n)
       x)))
```

6 CASOS DE USO

O Haskell está cada vez mais ganhando espaço em códigos de inúmeras empresas mundo a fora, alguns casos de uso atuais são:

6.1 Facebook

Dentro do Facebook a linguagem está presente em diversos projetos, porém o que mais chama a atenção é a engine chamada Sigma que filtra o spam de todas publicações feitas, um engenheiro citou o porque escolheram a linguagem foi escolhida:

- Puramente funcional e fortemente tipada - isso garante que as políticas não interajam entre elas, além da tipagem forte eliminar vários bugs antes de colocar essas politicas em produção.
- Performance - o Facebook recebe milhões de postagens diariamente, logo a performance oferecida pela linguagem foi decisivo para que ela fosse escolhida.
- Suporte para desenvolvimento interativo - o código pode ser testado interativamente, podendo ver os resultados na mesma hora.

6.2 Google

A linguagem é usada em alguns projetos internos, para o suporte de infraestrutura e para o seu projeto open-source Ganeti, uma ferramenta de gerenciamento de cluster em máquinas virtuais.

6.3 Nvidia

Empresa que projeta unidades de processamentos gráficos, utiliza o haskell em ferramentas internas.

6.4 Intel

Desenvolveu um compilador de haskell como parte de uma pesquisa em paralelismo multicore em grande escala.

6.5 Microsoft

Utiliza Haskell em seu sistema de serealização de produção, Bond, que é altamente usado em serviços de alta escala. Separadamente, o Microsoft Research é um patrocinador chave do desenvolvimento da linguagem desde 1990.

6.6 AT&T

Empresa do ramo telefônico, é uma operadora estadunidense que utiliza a linguagem em sua divisão de segurança em redes, automatizando alguns processos.

6.7 IMVU

Jogo lançado em 2004 de simulação de avatares 3D, o haskell foi utilizado para a construção de APIs REST pois o seu código que era em PHP não estava suportando as requisições, estava sobrecarregando e com uma latência alta, é o relatador em uma publicação pelo ex-diretor técnico do jogo, Chad Austin. Curiosamente o maior empecilho na implementação foram as pessoas que se recusavam a utilizar uma linguagem funcional.

7 EXEMPLOS DE CÓDIGOS

7.1 Hello World

```
module Main (main) where

main = putStrLn "Hello, World!"
```

Explicação da sintaxe:
Na primeira linha é declarado um modulo chamado Main, e é exportado a função main desse módulo. A função main é o ponto de entrada do programa, o módulo main pode ser qualquer coisa com a função main. O putStrLn imprime a string, deve ser em "(aspas duplas) pois as ' (aspas simples) só pode ser utilizada para caracteres.

7.2 Algoritmo de ordenação quicksort

É nítido ao comparar os códigos o quão Haskell consegue ter um código breve, conciso e auto explicativo em relação a uma linguagem do paradigma imperativo.

7.2.1 Código em C

```
#include <stdio.h>

void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

int partition(int array[], int low, int high) {
  int pivot = array[high];
  int i = (low - 1);

  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {
      i++;
      swap(&array[i], &array[j]);
    }
  }

  swap(&array[i + 1], &array[high]);
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {
    int pi = partition(array, low, high);
    quickSort(array, low, pi - 1);
    quickSort(array, pi + 1, high);
  }
}
```

```
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

int main() {
  int data[] = {8, 7, 2, 1, 0, 9, 6};

  int n = sizeof(data) / sizeof(data[0]);

  printf("Unsorted Array\n");
  printArray(data, n);

  quickSort(data, 0, n - 1);

  printf("Sorted array in ascending order: \n");
  printArray(data, n);
}
```

7.2.2 Código em Haskell

```
qsort []     = []
qsort (x:xs) = qsort small ++ mid ++ qsort large
  where
    small = [y | y<-xs, y<x]
    mid   = [y | y<-xs, y==x] ++ [x]
    large = [y | y<-xs, y>x]
```

Explicação da sintaxe:
Na primeira linha "o resultado de se ordenar uma lista vazia é uma lista vazia". Na segunda linha "para ordenar uma lista cujo primeiro elemento é x e o resto é chamado xs, ordene todos os elementos de xs que são menores que o x (que seria chamado de menor), ordene todos os elementos de xs que são maiores que o x e concatene (++) os resultados, com o x como pivo no meio". O small é a lista de y's que foram retirados da lista de xs, e y é menor do que x. Os símbolos utilizados significam: a setinha "retirar de"e o | "de tal modo".

7.3 Fibonacci

```
fib :: Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```

7.4 Código do 'Adivinha fatorial'

```
module Main where

factorial n = if n == 0 then 1 else n * factorial (n - 1)

main = do putStrLn "Quanto     5! ?"
          x <- readLn
          if x == factorial 5
              then putStrLn "Voc  acertou!"
              else putStrLn "Voc  errou!"
```

7.5 Gerador de números aleatórios

```
import System.Random

main = (randomRIO (1, 100) :: IO Int) >>= print
```

8 CONSIDERAÇÕES FINAIS

O Haskell possui uma comunidade e um suporte incrível que só é assim hoje pois foi cultivada desde o início da linguagem, com reuniões abertas, criação do site logo no início, disponibilização de documentos importantes e úteis, comunidade que hoje é grande e muito importante pois ajuda a disseminar ainda mais essa linguagem do paradigma funcional além de ajudar muitos iniciantes que podem vir a ter alguma dificuldade.

Por ser uma linguagem do paradigma funcional, para pessoas que aprenderam a programar utilizando alguma linguagem do paradigma imperativo pode ser complicado por terem alguns conceitos opostos.

É muito interessante a forma que o seu desenvolvimento foi conduzido e documentado, pois preservou informações que hoje infelizmente não temos de outras linguagens. Além de ter pessoas incríveis envolvidas durante toda essa jornada de desenvolvimento.

É impressionante o quanto hoje essa linguagem está presente na indústria, em empresas pequenas e grandes multinacionais, inclusive em empresas de segmentos bem diversificados.

9 REFERÊNCIAS

OLEYNIKOV, Denis. Haskell. History of a Community-Powered Language. [S. l.], 23 ago. 2019. Disponível em: https://serokell.io/blog/haskell-history. Acesso em: 28 ago. 2021. Nenhuma citação no texto.
BRIEF History of Haskell. [S. l.], 23 ago. 2019. Disponível em: https://www.futurelearn.com/info/courses/functional-programming-haskell/0/steps/27218. Acesso em: 28 ago. 2021. Nenhuma citação no texto.
HUDAK, Paul; WADLER, Philip; JONES, Peyton; HUGHES, John. A history of Haskell: being lazy with class. [S. l.], 23 ago. 2019. Disponível em: https://dl.acm.org/doi/10.1145/1238844.1238856. Acesso em: 28 ago. 2021. Nenhuma citação no texto.
SIMON Peyton Jones - A History of Haskell: being lazy with class. [S. l.: s. n.], 2016. Disponível em: https://www.youtube.com/watch?v=06x8Wf2r2Mc. Acesso em: 30 ago. 2021. Nenhuma citação no texto.
HASKELL Website. [S. l.], 1994. Disponível em: https://wiki.haskell.org/Haskell. Acesso em: 30 ago. 2021. Nenhuma citação no texto.
THOMPSON, Simon. The craft of Functional Programming. [S. l.]: Addison-Wesley, 1999. Nenhuma citação no texto.
HUDAK, Paul; FASEL, Joseph. A Gentle Introduction to Haskell. [S. l.], 1992. Disponível em: https://www.haskell.org/tutorial/. Acesso em: 4 set. 2021. Nenhuma citação no texto.
YANG, Jean. People of Programming Languages - Peyton Jones. [S. l.], 2018. Disponível em: https://www.cs.cmu.edu/ popl-interviews/peytonjones.html. Acesso em: 14 set. 2021. Nenhuma citação no texto.
FUNCTIONAL Programming Haskell - Computerphile. [S. l.: s. n.], 2016. Disponível em: https://www.youtube.com/watch?v=LnX3B9oaKzw. Acesso em: 17 set. 2021. Nenhuma citação no texto.
AUSTIN, Chad. The Story of Haskell at IMVU. [S. l.], 2016. Disponível em: https://chadaustin.me/2016/06/the-story-of-haskell-at-imvu/. Acesso em: 20 set. 2021. Nenhuma citação no texto.
SIMILAR functional languages. [S. l.], [2005-2020]. Disponível em: https://caml.inria.fr/pub/docs/oreilly-book/html/book-ora202.html. Acesso em: 22 set. 2021. Nenhuma citação no texto.
STANDARD ML of New Jersey. [S. l.], [2005-2020]. Disponível em: https://www.smlnj.org/. Acesso em: 24 set. 2021. Nenhuma citação no texto.
CLOJURE. [S. l.], [2005-2020]. Disponível em: https://clojure.org/. Acesso em: 24 set. 2021. Nenhuma citação no texto.
THE SCALA Programming Language. [S. l.], [2005-2020]. Disponível em: https://www.scala-lang.org/. Acesso em: 24 set. 2021. Nenhuma citação no texto.
LEARN Standard ML in Y Minutes. [S. l.], [2005-2020]. Disponível em: https://learnxinyminutes.com/docs/standard-ml/. Acesso em: 24 set. 2021. Nenhuma citação no texto.