Plano de Trabalho de Conclusão de Curso
Visão Categórica do Sistema de Tipos de Haskell

Rafael Castro G. Silva – rafaelcgs10@gmail.com
Karina Roggia – karina.roggia@gmail.com *(orientadora)*

Turma 2016/2 – Joinville/SC

19 de Julho de 2016

Resumo

Sistemas de tipos de linguagens de programação estão cada vez mais sofisticados e, em alguns casos, são modelos de lógica consistentes baseados nos formalismos criados no século passado. Na linguagem Haskell, estão presentes muitas ideias de Teoria de Tipos, Cálculo Lambda e Teoria das Categorias. O objetivo deste trabalho é explorar as teorias mencionadas e relacioná-las com o sistema de tipos de Haskell.

Palavras-chave: *Sistemas de tipos, teoria das categorias, cálculo lambda, haskell, correspondência curry-howard-lambek*

1 Introdução e Justificativa

Sistemas de tipos de linguagens de programação são regras que definem o que é um programa corretamente tipado e o propósito principal é a identificação de erros, como o descasamento do tipo do operador e do operando. Tipos também garantem que um programa tenha sentido operacional, pois o tipo do dado sinaliza como o processador deve tratá-lo. As regras do sistema de tipo são utilizadas por um algoritmo de verificação, o qual verifica se o tipo atribuído a um termo é válido [Sebesta 2012]. Nem todas as linguagens de programação tem sistemas de tipos formalmente especificados, o que pode resultar em inconsistências e pouca confiabilidade na corretude do programa. Linguagens como ML, Miranda e Haskell têm sistema de tipos conhecido como Hindley-Milner (HM), capaz de identificar o tipo de todas as possíveis expressões. A consistência do sistema HM é uma garantia de segurança, pois nem um erro de tipo acontece em tempo de execução. O sistema HM é fruto direto dos avanços da lógica no século passado, como a criação da Teoria dos Tipos e do Cálculo Lambda.

A Teoria dos Tipos foi criada em 1902 por Bertrand Russell e Alfred N. Whitehead com o propósito de ser uma fundação consistente para a matemática, inpirado pelo programa de Hilbert, por meio de uma solução *ad hoc* aos paradoxos da Teoria de Conjuntos [Hindley 1996]. Na Teoria dos Tipos cada termo habita um único tipo, diferente da Teoria dos Conjuntos a qual elementos podem pertencer a diversos conjuntos distintos. A única maneira de dois termos de tipos diferentes estarem num mesmo tipo é por meio da criação de um novo tipo híbrido. O tipo A de um termo

M é denotado por $M : A$ e funções são denotadas por $\rightarrow$. Assim, uma função sucessora *succ* pode ter o tipo $succ : Nat \rightarrow Nat$, onde Nat é o tipo que representa os números naturais na linguagem.

O Cálculo Lambda é um sistema formal apresentado por Alonzo Church em 1932 [Church 1932] com o propósito, assim como a Teoria dos Tipos, de ser uma fundação consistente para a matemática. Entre 1931 e 1934, Church orientou Stephen Kleene e Barkjley Rosser, e juntos fizeram diversas descobertas sobre o Cálculo Lambda. Inconvenientemente, uma delas foi a sua inconsistência. Em 1936, Church por meio de seu sistema formal provou a indecibilidade do problema da parada (Entscheidungsproblem), uma das peças do programa de Hilbert para fundação formal da matemática. Após o fracasso em tentar criar um sistema formal consistente, Church apresentou em 1940 um sistema formal menos ambicioso, o qual reformula a Teoria dos Tipos encima do Cálculo Lambda, conhecido como Cálculo Lambda Simplesmente Tipado.

Cálculo Lambda é uma teoria sobre funções e alguém poderia questionar como ela se relaciona com outras teorias funcionais. Teoria dos Conjuntos pode fornecer uma teoria de funções por meio de relações funcionais, porém mesmo no modelo de Zermelo um conjuto qualquer A é muito pequeno se comparado com a classe V, o universo de todos os conjutos. Assim, uma função $f : A \rightarrow B$ nos diz muito pouco sobre operações sobre todos os conjutos. Poderia-se estender Teria dos Conjuntos por meio de classes, mas isso cria tantos problemas quanto soluciona. Como apontado por Dana Scott em [Scott 1980], o que deseja-se é uma pura teoria de funções conhecida como Teoria das Categorias. Dana notou que quando se está lidando com qualquer Teoria dos Tipos, está se lidando com alguma estrutura categórica.

A Teoria das Categorias foi criada em 1942 por Samuel Eilenberg e Sauders Mac Lane como maneira de entender o processo de preservação de estruturas matemáticas na Topologia Algébrica. Em suma, Teoria das Categorias lida com objetos, morfismos e a composição de morfismos. Objetos são representados por pontos ou letras e morfismos são representados por setas $\rightarrow$, as quais representam qualquer tipo de transformação entre os objetos. Há diversas vantagens na abordagem categórica de teorias matemáticas (conjuntos, grupos, lógicas, tipos, etc), como dualidade, herança de resultados, notação gráfica e a expressividade das construções as quais permitem expressar estruturas complexas de forma simples [Menezes and Haeusler 2001].

A mais obvia relação entre Teoria das Categorias e Teoria dos Tipos (ou sistemas de tipos) é a categoria onde tipos são objetos e funções são morfismos. Elementos categóricos estão presentes em sistemas de tipos, como as monadas, as quais foram introduzidos no Haskell por Philip Wadler e Simon Peyton-Jones em Maio de 1996 como uma maneira de evitar efeitos colaterais em I/O [Hudak et al. 2007]. Com base nessa simples relação é possível utilizar o alto nível de abstração da Teoria das Categorias para descobrir propriedades universais e provas de equivalência entre funções, como feito por [Tate et al. 2012] na generalização de transformação de programas com base numa formalização categórica da prova. É notável que se uma linguagem de programação é construída com base numa teoria bem estabelecida, como a Teoria das Categorias, teoremas e propriedades são ganhos gratuitamente.

O Isomorfismo de Curry-Howard estabelece que proposições são tipos e provas construtivas (do intuicionismo de Brouwer) são programas. A equivalência foi descoberta em 1969 por William Alvin

Howard com base nos resultados prévios de Haskell Curry e relaciona diretamente o Cálculo Lambda Simplesmente Tipado com a Dedução Natural de Gentzen [Howard 1980]. Joachim Lambek descobriu que tal correspondência também está presente dentro das Categorias Cartesianas Fechadas, o que resulta na tripla correspondência Curry-Howard-Lambek [Lambek and Scott 1986].

Uma utilidade do Isomorfismo de Curry-Howard é o uso do sistema de tipos na prova de teoremas. [Sheard 2005] mostrou que com três funcionalidades adicionais no sistema de tipos de Haskell é possível provar propriedades de uma aplicação de transmissão de dados. Além disso, Haskell conta com diversos elementos de Teoria das Categorias como funtores e mônadas. Portanto, o sistema de tipos do Haskell é uma realização da correspondência Curry-Howard-Lambek. Poderia-se questionar se existem outros elementos de Categorias no sistema de tipos do Haskell e se novos poderiam ser inseridos.

2 Objetivos

Objetivo Geral: Estabelecer os conceitos categoriais do sistema de tipos da linguagem de programação Haskell.

Objetivos Específicos:

- Estudar os principais conceitos da Teoria dos Tipos.
- Compreender os principais conceitos e resultados de Teoria das Categorias.
- Estabelecer conceitos paralelos entre as duas teorias acima indicadas.
- Identificar estruturas destas duas Teorias dentro da linguagem de programação Haskell.
- Definir e apresentar exemplos concretos do uso destas estruturas.

3 Metodologia

Devido à natureza teórica do tema e trabalho proposto, o desenvolvimento se dará através de estudos bibliográficos e elaboração de comparações entre conceitos e exemplos concretos dentro da linguagem Haskell. Dentre as atividades inicialmente propostas, destacam-se:

- Estudos bibliográficos:
    1. Cálculo Lambda Livre de Tipos
    2. Teoria dos Tipos
    3. Cálculo Lambda Simplesmente Tipado
    4. Sistema de Tipos de Haskell
    5. Teoria das Categorias: categorias, objetos, morfismos, funtores, limites, transformações naturais, mônadas e categorias cartesianas fechadas.
    6. Correspondência Curry-Howard-Lambek

7. Semântica Categórica do Cálculo Lambda Tipado
8. Categoria do Sistema de Tipos

- Elaboração de:

9. Comparativos e analogias entre conceitos estudados
10. Exemplos concretos para Haskell
11. Texto final

4 Cronograma proposto

Etapas	Agosto		Setembro				Outubro				Novembro				Dezembro			
	1	2	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	■	■																
2			■	■	■													
3					■	■	■											
4								■	■	■	■							
5											■	■	■	■	■	■		
6																	■	■
9												■	■	■	■	■		
11		■			■				■	■	■	■	■	■				

Etapas	Fevereiro				Março				Abril				Maio				Junho	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
6	■	■	■															
7				■	■	■	■											
8								■	■	■	■							
9					■	■	■	■	■	■	■							
10												■	■	■	■	■		
11		■	■				■	■	■	■	■	■	■	■	■	■	■	■

5 Forma de Acompanhamento/Orientação

A orientação será realizada em reuniões semanais, presenciais e/ou via *chat*, com até uma hora de duração. Também serão utilizados recursos como correio eletrônico para, caso necessário, orientação ao longo da semana.

A orientadora, através dos e-mails ou anotações realizadas durante as reuniões, realizará um controle de acompanhamento (data das reuniões, tarefas realizadas pelo aluno, tarefas que devem ser realizadas pelo aluno durante a semana e observações adicionais como o cumprimento ou não dos prazos estabelecidos para as tarefas).

A orientadora solicitará tarefas a serem realizadas durante a semana. Caso o aluno não cumpra alguma tarefa e/ou cumpra após o prazo estipulado (de forma que o orientador não possua tempo hábil para realizar as correções necessárias), o orientador poderá solicitar seu desligamento da orientação. O desligamento também poderá ser solicitado caso o aluno não compareça às reuniões agendadas.

Todos os materiais gerados durante o trabalho deverão ser entregues à orientadora da seguinte forma:

- Documentos textuais e de apresentação em formato editável;

- Pesquisa do estado da arte (referências bibliográficas tais como na fundamentação teórica e trabalhos relacionados) em formatos padrões;
- Base de informações, dados e programas no formato da aplicação (via diretório no Dropbox ou Google Drive, ou em repositório no Github);
- Quando solicitados previamente, alguns documentos em papel.

6 Referências

[Church 1932] Church, A. (1932). A set of postulates for the foundation of logic part i. *Annals of Mathematics*, 33(2):346–366. http://www.jstor.org/stable/1968702Electronic Edition.

[Hindley 1996] Hindley, J. R. (1996). *Basic simple type theory*. Cambridge tracts in theoretical computer science. Cambridge University Press, New York.

[Howard 1980] Howard, W. A. (1980). The formulas-as-types notion of construction. In Seldin, J. P. and Hindley, J. R., editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press.

[Hudak et al. 2007] Hudak, P., Hughes, J., Peyton Jones, S., and Wadler, P. (2007). A history of haskell: Being lazy with class. In *Proceedings of the Third ACM SIGPLAN Conference on History of Programming Languages*, HOPL III, pages 12–1–12–55, New York, NY, USA. ACM.

[Lambek and Scott 1986] Lambek, J. and Scott, P. J. (1986). *Introduction to Higher Order Categorical Logic*. Cambridge University Press, New York, NY, USA.

[Menezes and Haeusler 2001] Menezes, P. and Haeusler, E. H. (2001). *Teoria das Categorias para Ciência da Computação*, volume 12 of *Livros Didáticos*. Editora Sagra Luzzatto, first edition.

[Scott 1980] Scott, D. S. (1980). Relating theories of the lambda calculus. *To HB Curry: Essays on combinatory logic, lambda calculus and formalism*, pages 403–450.

[Sebesta 2012] Sebesta, R. W. (2012). *Concepts of Programming Languages*. Pearson, 10th edition.

[Sheard 2005] Sheard, T. (2005). Putting curry-howard to work. In *Proceedings of the 2005 ACM SIGPLAN Workshop on Haskell*, Haskell '05, pages 74–85, New York, NY, USA. ACM.

[Tate et al. 2012] Tate, R., Stepp, M., and Lerner, S. (2012). Generating compiler optimization from proofs.

Karina Girardi Roggia ***Rafael Castro G. Silva***