Plano de Trabalho de Conclusão de Curso
Inferência de tipos para CPS

UDESC – Centro de Ciências Tecnológicas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação
Turma 2024/2 – Joinville/SC

Vinícios Bidin Santos – vinibidin@gmail.com
Cristiano Damiani Vasconcellos – cristiano.vasconcellos@udesc.br *(orientador)*
Paulo Henrique Torrens – paulotorrens@gnu.org *(coorientador)*

Agosto de 2024

Resumo

Este trabalho propõe uma investigação sobre a inferência de tipos para o Estilo de Passagem de Continuação (CPS) - representação intermediária amplamente utilizada em compiladores de linguagens funcionais. A pesquisa se concentra na extensão do algoritmo W, tradicionalmente usado para inferência de tipos no sistema Damas-Milner, para abranger o cálculo de continuações. A proposta inclui a implementação dessa extensão na linguagem Haskell e a validação do algoritmo por meio de programas de teste, assegurando que os tipos inferidos estejam corretos.

Palavras-chave: ***Inferência de Tipos, Estilo de Passagem de Continuação (CPS), Algoritmo W, Damas-Milner, Haskell, Sistema de Tipos***

1 Introdução e Justificativa

A compilação de programas envolve diversas fases, cada uma com funções específicas, como análise léxica, análise sintática, análise semântica, otimizações, e, finalmente, a geração de

código. Uma etapa crítica nesse processo é a otimização, que frequentemente se baseia em representações intermediárias (IRs). Essas representações atuam como ponte entre o código fonte e o código de máquina, permitindo que transformações e otimizações sejam aplicadas de maneira mais eficaz (PLOTKIN, 1975).

As representações intermediárias variam conforme o paradigma da linguagem de programação. Para linguagens imperativas, a Representação em Atribuição Única Estática (*SSA*) é amplamente adotada. Já em linguagens funcionais, a Forma Normal Administrativa (*ANF*) e o Estilo de Passagem de Continuação (*CPS*) se destacam. Este trabalho foca especificamente no CPS, uma IR que oferece vantagens particulares em termos de otimização e simplicidade na geração de código.

Em linguagens de alto nível, a pilha de chamadas é uma abstração fundamental, fornecendo controle sobre onde a execução de uma função deve retornar após sua conclusão. Entretanto, em linguagens de baixo nível, como *assembly*, tal abstração não existe, exigindo que o controle do fluxo de execução seja realizado manualmente por meio de endereços de retorno.

O CPS resolve essa questão ao tornar as continuações explícitas no código. Em vez de confiar na pilha de chamadas para gerenciar retornos, o CPS introduz um parâmetro adicional (KENNEDY, 2007) em cada função, representando a continuação — o que deve ser feito com o resultado da função. Ao invés de simplesmente retornar um valor, a função invoca essa continuação, passando o controle explicitamente à próxima etapa da computação. Isso elimina a necessidade de uma pilha de chamadas, simplificando o modelo de execução e tornando-o mais alinhado com a execução em baixo nível.

A escolha do CPS como representação intermediária não é apenas uma questão de conveniência para a tradução de linguagens de alto nível para código de máquina. Ele também facilita a aplicação de otimizações complexas, como a eliminação de chamadas de cauda e a fusão de funções, além de permitir uma correspondência mais direta com o código gerado em linguagens de montagem (FLANAGAN et al., 1993).

Entretanto, muitas implementações optam por representar CPS sem tipos (MORRISETT et al., 1999), o que, embora simplifique a implementação inicial, pode sacrificar a segurança do código. Um sistema de tipos robusto pode não apenas garantir a correção de certas transformações e otimizações, mas também identificar uma classe inteira de erros antes da execução, proporcionando uma camada adicional de segurança e confiabilidade ao processo de compilação.

2 Objetivos

Objetivo Geral:

Este trabalho visa explorar a inferência de tipos para CPS, propondo uma extensão do algoritmo W - um algoritmo de inferência de tipos para o sistema Damas-Milner - para o cálculo de continuações, definido em (TORRENS; ORCHARD; VASCONCELLOS, 2024) na linguagem de programação Haskell.

Objetivos Específicos:

- Formular uma extensão do algoritmo W para CPS;
- Implementar essa extensão em Haskell;
- Validar a implementação do algoritmo por meio da geração de programas e verificação de que o algoritmo infere corretamente os tipos para eles.

3 Metodologia

A metodologia deste trabalho consistirá em duas principais etapas: pesquisa bibliográfica e implementação. A primeira etapa envolve uma extensa revisão de literatura sobre continuações e seu cálculo, bem como um aprofundamento do sistema Damas-Milner, com o objetivo de proporcionar uma compreensão completa ao autor. A segunda etapa trata da formulação do algoritmo W estendido para o cálculo de continuações, junto com sua implementação.

Para validar a implementação, serão gerados programas de teste, nos quais a inferência de tipos será verificada. A validação consistirá em comparar os tipos inferidos pelo algoritmo com os tipos esperados para esses programas.

4 Cronograma proposto

1. Fundamentação teórica sobre continuações, cálculo de continuações e CPS;
2. Fundamentação teórica sobre o sistema Damas-Milner;
3. Formulação da extensão do algoritmo de inferência de tipos;
4. Implementação do algoritmo em Haskell definido na etapa 3;
5. Validação do resultado.

Etapas	2024/2					2025/1					
	Ago	Set	Out	Nov	Dez	Jan	Fev	Mar	Abr	Maio	Jun
1											
2											
3											
4											
5											

5 Linha e Grupo de Pesquisa

O tema proposto para o trabalho se enquadra nos temas pesquisados pelo Grupo de Pesquisa em Fundamentos da Computação (FUNÇÃO). O tema é pertinente à linha de pesquisa em linguagens funcionais e sistema de tipos.

6 Forma de Acompanhamento/Orientação

O acompanhamento das atividades desenvolvidas será realizada em reuniões semanais entre orientador, coorientador e aluno, presenciais e/ou via chat, com até uma hora de duração e possibilidade de agendar encontros extras, caso necessário. Também serão utilizados recursos como correio eletrônico para, caso necessário, orientação ao longo da semana. Para acompanhamento do progresso, serão entregues aos orientadores artefatos com os resultados obtidos em cada etapa (quando aplicável), para que assim exista uma avaliação contínua com mais ciclos de retorno.

Referências

FLANAGAN, C. et al. The essence of compiling with continuations. In: **Proceedings of the ACM SIGPLAN 1993 Conference on Programming Language Design and Implementation**. New York, NY, USA: Association for Computing Machinery, 1993. (PLDI '93), p. 237–247. ISBN 0897915984. Disponível em: <https://doi.org/10.1145/155090.155113>.

KENNEDY, A. Compiling with continuations, continued. In: **Proceedings of the 12th ACM SIGPLAN International Conference on Functional Programming**. New York, NY, USA: Association for Computing Machinery, 2007. (ICFP '07), p. 177–190. ISBN 9781595938152. Disponível em: <https://doi.org/10.1145/1291151.1291179>.

MORRISETT, G. et al. From system f to typed assembly language. **ACM Trans. Program. Lang. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 21, n. 3, p. 527–568, may 1999. ISSN 0164-0925. Disponível em: <https://doi.org/10.1145/319301.319345>.

PLOTKIN, G. Call-by-name, call-by-value and the λ-calculus. **Theoretical Computer Science**, v. 1, n. 2, p. 125–159, 1975. ISSN 0304-3975. Disponível em: <https://www.sciencedirect.com/science/article/pii/0304397575900171>.

TORRENS, P.; ORCHARD, D.; VASCONCELLOS, C. On the operational theory of the cps-calculus: Towards a theoretical foundation for irs. **Proc. ACM Program. Lang.**, Association for Computing Machinery, New York, NY, USA, v. 8, n. ICFP, aug 2024. Disponível em: <https://doi.org/10.1145/3674630>.

Cristiano Damiani Vasconcellos
Orientador

Vinícios Bidin Santos

Karina Girardi Roggia
(Líder do Grupo de Pesquisa Fundamentos da Computação)