Category Theory Arrows e Monads em Java

Mozart L. Siqueira
Ciências da Computação
Centro Universitário La Salle - Unilasalle
Email: mozarts@unilasalle.edu.br

Pablo M. Parada
Ciências da Computação
Centro Universitário La Salle - Unilasalle
Email: pablo.paradabol@gmail.com

***Resumo*—<escrever>**
***Index Terms*—<escrever>**

I. INTRODUÇÃO

<escrever>

II. SOBRE O PARADIGMA FUNCIONAL

Influenciado principalmente pelo desenvolvimento do *lambuda calculus* [1], compondo o grupo da programação declarativa, o paradigma funcional utiliza-se da idéia de expressar computações através de funções combinadas em expressões. Neste, funções expressam o que deverá ser computado, ao invés de como sera computado [2]. Programas são construídos através da composição, tal que funções triviais (ou *building blocks*) são combinadas dando origem a novas funções que descrevem computações mais complexas.

Building blocks não devem fazer uso de variáveis que dependam de estado, isso significa que a computação deve ser pura e sem efeitos indesejados (ou *side-effects*). Também destaca-se o princípio de imutabilidade, onde o valor é de uma variável é determinado em sua criação, não permitindo novas atribuições posteriormente.

É possível afirmar que ao expressar um programa em uma linguagem funcional, obtem-se uma maneira concisa de solucionar problemas, dado que este constitui-se de operações e objetos atômicos e regras gerais para sua composição [3]. Estas qualidades são apreciadas nos tempos atuais, onde há necessidade de tratar os problemas oriundos do não-determinismo. Assim, o paradigma funcional mostra-se capaz, inclusive de influenciar outras linguagens como *Java* [4].

A. Anonymous Inner Classes e Lambda Expressions

A cada release cycle, Java incrementa seu ambiente de desenvolvimento fornecendo novos recursos para seus desenvolvedores. Em sua oitava distribuição, a linguagem adicionou as funcionalidades necessárias para habilitar o uso de funções de primeira classe (ou *lambda expressions*).

AICs fornecem instruções básicas para a criação de representações concretas de interfaces e classes abstratas. Quando instanciadas, devem conter a implementação definida pelo contrato de sua interface. Estas podem também referenciar o objeto corrente utilizando a palavra reservada *this*, possibilitando a invocação de métodos e a mutação de variáveis.

REFERÊNCIAS

[1] P. Hudak, "Conception, evolution, and application of functional programming languages," *ACM Computing Surveys (CSUR)*, vol. 21, no. 3, pp. 359–411, 1989.
[2] K. Louden *et al.*, *Programming languages: principles and practices*. Cengage Learning, 2011.
[3] G. Michaelson, *An introduction to functional programming through lambda calculus*. Courier Corporation, 2011.
[4] B. Goetz, R. Forax, D. Lea and B. Lee, "State of the lambda," Sept 2013, White Paper. [Online]. Available: http://cr.openjdk.java.net/ briangoetz/lambda/lambda-state-final.html