On Understanding Types, Data Abstraction and Polymorphism

Rafael Castro G. Silva

rafaelcgs10@gmail.com

Departamento de Ciência da Computação
Centro de Ciências e Tecnológicas
Universidade do Estado de Santa Catarina

4 de Setembro de 2017

Introdução

O artigo "On Understanding Types, Data Abstraction and Polymorphism" apresenta uma classificação de sistemas de tipos. E como monomorfismo, polimorfismo (paramétrico e subtipagem) e dados abstratos são classificados.

Classificação de Sistemas de Tipos

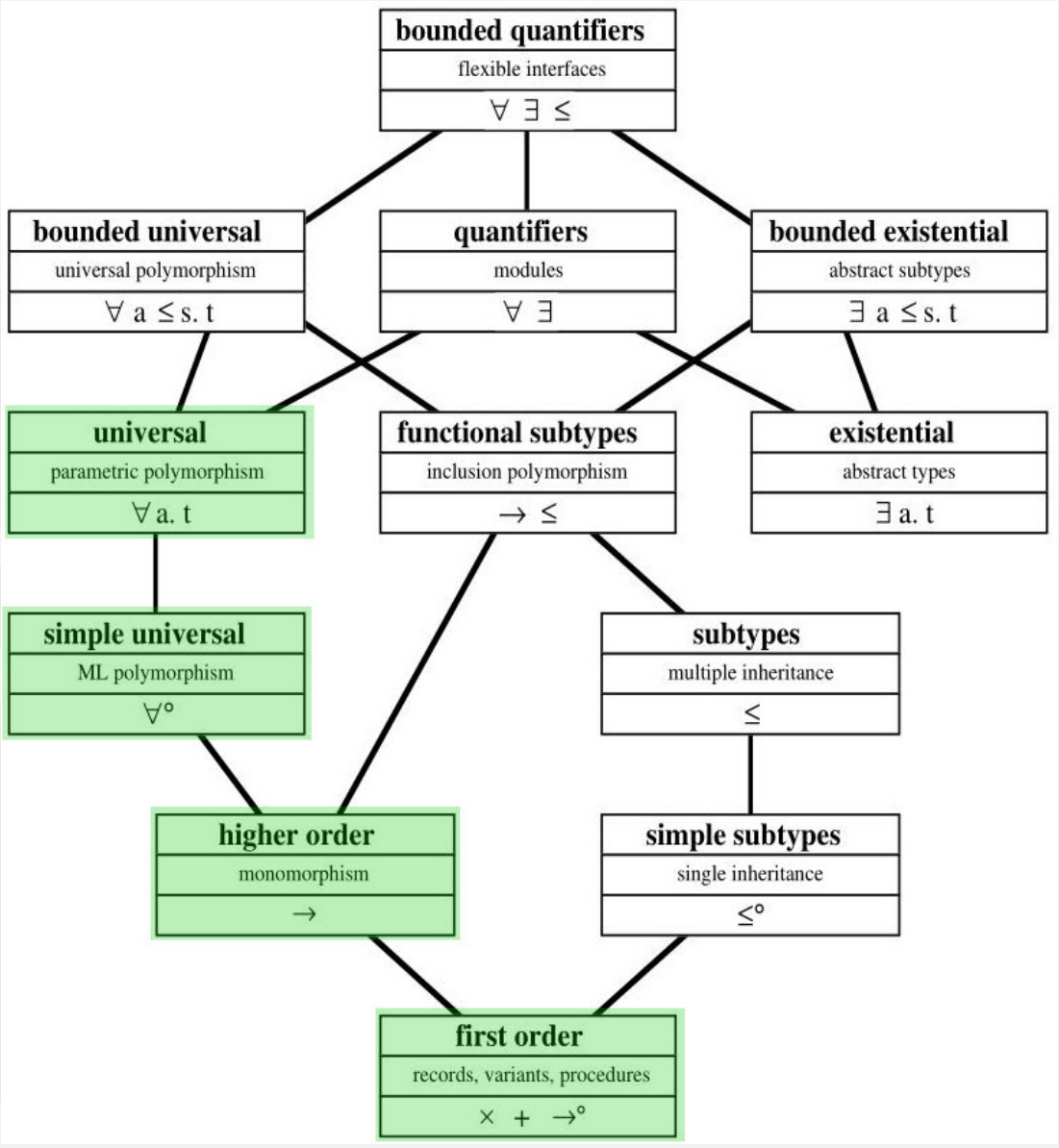

Figura: Classificação Sistemas de Tipos. Retirado de [1].

Primeira Ordem

$$FunType := ConstType \rightarrow FunType$$
$$ConstType := Int, Float, String...$$

- Funções não são dados.
- Linguagens: Java < 8, Python < 2, Ruby...

Ordem Superior

$$Type := Type \rightarrow Type | ConstType$$
$$ConstType := Int, Float, String...$$

- Funções são dados.
- Linguagens: Algol 68, Fortran, C, Pascal, Java $\geq$ 8, Python $\geq$ 2, Haskell...

Polimorfismo ML (Simple Universal)

$$
\begin{aligned}
PolyType &:= \forall VarType.PolyType | Type \\
Type &:= Type \rightarrow Type | ConstType | VarType \\
ConstType &:= Int, Float, String... \\
VarType &:= a, b, c...
\end{aligned}
$$

- Funções são polimórficas: uma função assume vários tipos.
- Funções de ordem superior tratam argumentos de maneira monomórfica.
- Linguagens: ML, OCaml, Haskell...

Segunda Ordem (Universal)

$$Poly := \forall VarType.Type | Type \rightarrow Type | ConstType | VarType$$
$$ConstType := Int, Float, String...$$
$$VarType := a, b, c...$$

- Funções são polimórficas: uma função assume vários tipos.
- Funções de ordem superior tratam argumentos de maneira polimórficas.
- Linguagens: Haskell...

Considerações finais

- Polimorfismo (subtipagem e paramétrico) e dados abstrados podem ser formalizados pela Teoria dos Tipos.
- A classificação das linguagens é boa do ponto de vista teórico, mas não é ideal para classificar as linguagens reais.
- Linguagens reais, geralmente, não são implementadas usando tais formalizações.