Anotação Individual

1. FROM UNTYPED TO TYPED UNIVERSES

Nessa seção, o autor nos faz questionar e refletir sobre o mundo da tipagem de dados. Ele explica a não tipagem através de exemplos que nos mostra que não tipagem é ser de apenas um tipo. No sexto paragrafo ele nos diz que tipos surgem quando temos a necessidade de categorizar objetos de acordo com seu uso e comportamento. Classificar esses objetos nos traz um sistema de tipos que nasce naturalmente.

1.2 STATIC AND STRONG TYPING

O autor nos define a tipagem estática e a tipagem forte.

A tipagem estática permite que inconsistência de tipos sejam descobertos em tempo de compilação e garante que os programas executados tenham tipos consistentes. Sendo assim, a execução do programa irá sempre ter funções de tipo consistente, tornando a tipagem forte. Se uma linguagem é estaticamente tipada então ela também é forte, mas a premissa contrária não é necessariamente verdadeira.

1.3 KINDS OF POLYMORPHISM

Definindo o que seria polimorfismo, o autor explica como se aplicaria o conceito tanto para funções quanto para tipos. Uma função aceitaria mais de um tipo de parâmetro. Um tipo polimórfico seria aquele cujas operações seriam aplicáveis a operandos de mais de um tipo.

Tipos de polimorfismo:

-Universal

-> paramétrico: uma função pode funcionar com vários tipos de dados diferentes, uma estrutura.

-> inclusão: basicamente, um tipo que seria herdado.

-Não Universal

-> sobrecarga: seria o caso da linguagem Java, temos uma funcionalidade, uma maneira de executar uma função para cada tipo de dado que seja requisitado.

-> coerção: podemos realizar sobre um determinado tipo, um tratamento que não é dele de natureza, mas devemos convertê-lo para que isso seja necessário.

1.4 The Evolution of Types in Programming Languages

O passar do tempo nos faz procurar soluções para problemas. Isso significa que a tipagem de dados se torna necessário para que problemas sejam resolvidos e possamos evoluir. O autor conta um pouco sobre a história da tipagem ao longo do tempo.

1.5 Type Expression Sublanguages

Sublíngua de expressão de tipos, é algo que o autor discute sobre como deve ser criada, funcional, simples e prática. Ela deve ser suficientemente rica para suportar tipos de todos os valores com que se deseja computar, mas suficientemente tratável para permitir decisões e checagem de tipos eficiente. Um dos propósitos desse artigo é examinar trocas entre riqueza e tratabilidade para SET's de linguagem de tipos fortes.

2 THE λ-CALCULUS

Nessa seção, o autor explica detalhadamente como funciona o cálculo lambda para linguagens sem tipagem, com tipagem e com tipagem básica. Sendo o cálculo lambda um sistema que estuda funções recursivas computáveis, levando em conta o tipo de variáveis ligadas a elas.