Anotações Individual TP1

1 . FROM UNTYPED TO TYPED UNIVERSES

1.1 Organizing Untyped Universes

A primeira parte do artigo mostra um pouco sobre a trajetória de universos não tipados para universos tipados. Universos não tipados são aqueles que apresentam apenas um tipo, e o artigo explica e mostra isso com os exemplos abaixo:

1) bit strings na memória do computador.
2) S-expressions em LISP.
3) λ-expressions no cálculo λ.
4) sets na teoria dos conjuntos.

No final dessa seção, é colocado mostrado que tentar observar um universo não tipado como sendo tipado é algo complicado de ser realizado, já que os conceitos antes criados para explicar distinções de tipos seriam violados e por não levar em consideração conceitos e interações de representações de maior grau.

1.2 Static and Strong Typing

No começo da sessão, é apresentado como sistema tipados resolvem problemas que antes foram citados sobre universos não tipados. Durante a sessão é discutido sobre os benefícios e malefícios de se adotar linguagens com ambos os termos citados abaixo:

- **Static Typing:** Está relacionada em que cada tipo de toda expressão pode ser determinada por um programa de análise estática. É uma propriedade muito útil, mas pode acabar se tornando muito restritiva (já que todas as variáveis e expressões precisam estar associadas a algum tipo).

- **Strong Typing:** Está relacionada em todas as expressões terem tipos consistentes, garantindo assim a execução do compilador sem erros.

Observação: É importante lembrar que toda linguagem estaticamente tipada é fortemente tipada, mas o contrário não se aplica.

1.3 Kinds of Polymorphism

Nesta parte do artigo se mostra a classificação de tipos de polimorfismos presentes. Abaixo está os termos citados:

- **Monomórfica:** Linguagens de programação em que todo valor ou variável pode ser interpretada por um único tipo.
- **Polimórfica:** Linguagens de programação em que algum valor ou variável pode ser interpretada por um único tipo.

- **Polimorfismo paramétrico:** A função funciona de forma uniforme com uma gama de tipos, sendo que esses tipos apresentam uma estrutura em comum.
- **Polimorfismo de inclusão:** Capacidade de modelar subtipos e herança.
- **Polimorfismo de sobrecarga:** O mesmo nome de variável é usado para indicar diferentes funções e o contexto é usado para decidir que função será indicada por uma instância do nome.
- **Polimorfismo de coerção:** Operação semântica é necessária para converter um tipo específico para uma função, que resultaria em erro como em qualquer situação.

1.4 The Evolution of Types in Programming Languages

Nesta parte do artigo é discutir a evolução da tipagem nas linguagens de programação com o passar do tempo.

1.5 Type Expression Sublanguages

Nesta seção é apresentado a importância de definir uma . O intuito disso é definir um set de tipos para a língua, já que os sets de tipos definitivos acabam se tornando difíceis de serem identificados com o enriquecimento da língua.

Sendo um dos pontos que o artigo propõe, é o estudo dos pontos positivos e negativos entre riqueza e tratabilidade para expressões típicas de sub linguagens de linguagens fortemente tipadas.

1.6 Preview of Fun

Nesta seção é apresentada a modelagem criada pelos dois autores sobre uma nova linguagem de programação que se baseia em todos os conceitos estabelecidos nas seções anteriores. **Fun** é uma linguagem baseada no cálculo λ, que se baseia na sua tipagem e capaz de modelar polimorfismo e linguagens orientadas a objetos.

2. THE λ-CALCULUS

2.1 The Untyped λ-Calculus

Nesta seção é mostrado o cálculo λ em notação não tipada, com o objetivo de mostrar a funcionalidades das aplicações de operandos para operandos.

2.2 The Typed λ-Calculus

Nesta seção é mostrado o cálculo λ em notação tipada, sendo que todas as variáveis explicitamente com tipos deveriam ser introduzidas como uma variável de ligação.

2.3 Basic Types, Structured Types, and Recursion

Nesta seção é mostrado no cálculo λ os tipos básicos da linguagem (citando como exemplo: Unit, Bool, Int, Real, String), como também é citado a estruturação de tipos e como funciona a recursão na linguagem.