

# Sistemas de Tipos

Rafael Castro

[rafaelcgs10@gmail.com](mailto:rafaelcgs10@gmail.com)

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

10 de Julho de 2019

# Plano de Aula - Sistema de Tipos

## Objetivos

Explicar o que são e para que servem sistemas de tipos na Ciência da Computação.

## Conteúdos

- Breve visão histórica.
- Usos informais em linguagens de programação.
- Sistemas de tipos formais em linguagens de programação.

## Método

Exposição do conteúdo com exemplos, questionamentos e exercícios.

## Breve visão histórica

# Visão histórica - Russel

## Teoria dos Tipos

- Bertrand Russell
- Teorias de Tipos são uma resposta ao paradoxo de Russel presente na *naive set theory*.
- Uma teoria alternativa a Teoria dos Conjuntos para uma fundação formal da matemática.



# Visão histórica - Church

## Teoria dos Tipos

- Alonzo Church
- Aplicou a teoria de tipos de Russel no modelo de computação chamado Cálculo Lambda, a fim de evitar paradoxos e computações sem fim.
- Essa junção entre Teoria de Tipos e Cálculo Lambda é o fundamento teórico para sistemas de tipos em linguagens de programação.



## Usos informais em linguagens de programação

# Questões preliminares

O que são tipos (em linguagens de programação)?



## Questões preliminares

O que são tipos (em linguagens de programação)?

- Classificações dos dados de um programa: carácter, inteiro, função etc.

## Questões preliminares

O que são tipos (em linguagens de programação)?

- Classificações dos dados de um programa: carácter, inteiro, função etc.

Por que linguagens de programação utilizam tipos?

## Questões preliminares

O que são tipos (em linguagens de programação)?

- Classificações dos dados de um programa: carácter, inteiro, função etc.

Por que linguagens de programação utilizam tipos?

```
void foo() {  
    int n = 1;  
    char m = '2';  
    printf("%d\n", n + m);  
}
```

```
void spam() {  
    int n;  
    scanf("%d", &n);  
    printf("%d\n", n);  
    return 0;  
}
```

## Questões preliminares

O que são tipos (em linguagens de programação)?

- Classificações dos dados de um programa: carácter, inteiro, função etc.

Por que linguagens de programação utilizam tipos?

```
void foo() {  
    int n = 1;  
    char m = '2';  
    printf("%d\n", n + m);  
}
```

```
void spam() {  
    int n;  
    scanf("%d", &n);  
    printf("%d\n", n);  
    return 0;  
}
```

Os tipos fazem parte da construção semântica dos programas.

Existem linguagens de programação sem tipos (*untyped*), como linguagens assembly, BCPL, Tcl, Brainfuck.

# Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.

## Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.
- Estática: atribuições de tipo não mudam em *run time*.
- Dinâmica: atribuições de tipo **podem mudar** em *run time*.

## Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.
- Estática: atribuições de tipo não mudam em *run time*.
- Dinâmica: atribuições de tipo **podem mudar** em *run time*.

Exemplos de linguagens de tipagem estática?

# Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.
- Estática: atribuições de tipo não mudam em *run time*.
- Dinâmica: atribuições de tipo **podem mudar** em *run time*.

Exemplos de linguagens de tipagem estática?

C, C++, Java, Rust, Haskell, OCaml...

# Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.
- Estática: atribuições de tipo não mudam em *run time*.
- Dinâmica: atribuições de tipo **podem mudar** em *run time*.

Exemplos de linguagens de tipagem estática?

C, C++, Java, Rust, Haskell, OCaml...

Exemplos de linguagens de tipagem dinâmica?

# Tipagem estática vs dinâmica

- Na tipagem estática a verificação dos tipos é feita em por análise do código durante a compilação.
- Na tipagem dinâmica a verificação dos tipos é feita somente na execução.  
Se uma parte do código com erro (de tipo) não for executada, o erro ficará oculto.
- Estática: atribuições de tipo não mudam em *run time*.
- Dinâmica: atribuições de tipo **podem mudar** em *run time*.

Exemplos de linguagens de tipagem estática?

C, C++, Java, Rust, Haskell, OCaml...

Exemplos de linguagens de tipagem dinâmica?

Python, PHP, JavaScript, Lisp, Erlang...

# Vantagens e desvantagens: estática e dinâmica

## Estática:

- Vantagens: Reduz os possíveis erros em tempo de execução; Possibilita algumas otimizações de código pelo compilador; Verificação automática; Útil em códigos grandes.
- Desvantagens: Mais uma coisa para aprender sobre a linguagem; Requerem que tipos sejam explicitados (não todas as linguagens); Podem limitar o reuso de código.

## Dinâmica:

- Vantagens: Mais permissível; Mais fácil de aprender; geralmente tem mais interatividade (REPL); geralmente suportam *duck typing*.
- Desvantagens: Testes são muito mais importantes; abre espaço para erros inesperados em execução; geralmente são menos eficientes.

# Tipagem forte e fraca

Esse é um conceito meio nebuloso!

Não é preto ou branco

- Forte: as regras de tipagem são rígidas; não há cast de tipos implícitos; os casts de tipos fazem sentido.
- Fraca: as regras de tipagem são flexíveis; casts automáticos podem ocorrer.

# Tipagem forte e fraca

Esse é um conceito meio nebuloso!

Não é preto ou branco

- Forte: as regras de tipagem são rígidas; não há cast de tipos implícitos; os casts de tipos fazem sentido.
- Fraca: as regras de tipagem são flexíveis; casts automáticos podem ocorrer.

Em Python:

```
"Dia " + str(1) + " de Janeiro"
```

# Tipagem forte e fraca

Esse é um conceito meio nebuloso!

Não é preto ou branco

- Forte: as regras de tipagem são rígidas; não há cast de tipos implícitos; os casts de tipos fazem sentido.
- Fraca: as regras de tipagem são flexíveis; casts automáticos podem ocorrer.

Em Python:

```
"Dia " + str(1) + " de Janeiro"
```

Em JavaScript:

```
[] + [] ; // ""  
[] + 1; // "1"  
1 + {} ; // "1[object object]"  
{ } + [] // 0
```

# Sistemas de tipos formais em linguagens de programação

# Questões preliminares

O que são tipos (na matemática)?



## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

- Teoria dos Tipos é uma teoria matemática sobre construções.
- Teoria dos Conjuntos é uma teoria matemática sobre coleções.

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

- Teoria dos Tipos é uma teoria matemática sobre construções.
- Teoria dos Conjuntos é uma teoria matemática sobre coleções.

O que são sistemas de tipos (formais)?

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

- Teoria dos Tipos é uma teoria matemática sobre construções.
- Teoria dos Conjuntos é uma teoria matemática sobre coleções.

O que são sistemas de tipos (formais)?

Sistemas de tipos são relações que ditam quais expressões/construções podem ser associadas a quais tipos.

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

- Teoria dos Tipos é uma teoria matemática sobre construções.
- Teoria dos Conjuntos é uma teoria matemática sobre coleções.

O que são sistemas de tipos (formais)?

Sistemas de tipos são relações que ditam quais expressões/construções podem ser associadas a quais tipos.

Por que definir sistemas de tipos formalmente?

## Questões preliminares

O que são tipos (na matemática)?

Tipos são propriedades atribuídas a certas construções/objetos.

Ex: 1 tem o tipo natural.

Qual a diferença entre tipos e conjuntos?

- Teoria dos Tipos é uma teoria matemática sobre construções.
- Teoria dos Conjuntos é uma teoria matemática sobre coleções.

O que são sistemas de tipos (formais)?

Sistemas de tipos são relações que ditam quais expressões/construções podem ser associadas a quais tipos.

Por que definir sistemas de tipos formalmente?

Para garantir boas propriedades semânticas de programas.

Ex: Evita que você crie JavaScript.

# Exemplo básico de sistema de tipo formal

A linguagem de programação:

Gramática informal dos programas:

$$e := x \mid \lambda x. e \mid e \ e'$$

Gramática informal dos tipos:

$$\tau := \alpha \mid \tau \rightarrow \tau'$$

Exemplos de programas:

$x$   
 $x\ y$   
 $\lambda x. x$   
 $(\lambda x. x)\ y$   
 $\lambda x. \lambda y. \lambda z. y$

# Exemplo básico de sistema de tipo formal

Relação de tipagem:

$$\frac{}{x : \tau, \Gamma \vdash x : \tau} (\textit{var})$$

$$\frac{\Gamma \vdash e : \tau \rightarrow \tau' \quad \Gamma \vdash e' : \tau}{\Gamma \vdash e\ e' : \tau'} (\textit{abs}) \quad \frac{x : \tau, \Gamma \vdash e : \tau'}{\Gamma \vdash \lambda x. e : \tau \rightarrow \tau'} (\textit{app})$$

# Exemplo básico de sistema de tipo formal

Relação de tipagem:

$$\frac{}{x : \tau, \Gamma \vdash x : \tau} (\text{var})$$

$$\frac{\Gamma \vdash e : \tau \rightarrow \tau' \quad \Gamma \vdash e' : \tau}{\Gamma \vdash e \ e' : \tau'} (\text{abs}) \quad \frac{x : \tau, \Gamma \vdash e : \tau'}{\Gamma \vdash \lambda x. e : \tau \rightarrow \tau'} (\text{app})$$

Exemplos de tipagem:

$$\frac{x : a \vdash x : a}{\vdash \lambda x. x : a \rightarrow a} (\text{abs})$$

$$\frac{\frac{\frac{y : b \vdash y : b}{y : b \vdash \lambda z. y : a \rightarrow b} (\text{abs})}{\vdash \lambda y. \lambda z. y : b \rightarrow a \rightarrow b} (\text{abs})}{\vdash \lambda x. \lambda y. \lambda z. y : c \rightarrow b \rightarrow a \rightarrow b} (\text{abs})$$

# Exercício sobre tipagem

Construa a árvore de tipagem para a expressão

$$(\lambda x.x)(\lambda x.x)$$

# Linguagens de programação baseados em sistemas de tipos formais

A mais importante propriedade semântica que um sistema de tipos pode garantir para uma linguagem de programação é *type soundness*.

Também chamado informalmente de *type safety*.

Significa que erros de tipo não acontecem em programas bem-tipados.

ML é uma linguagem de programação funcional com a propriedade de *type soundness*.

# Resposta do Exercício

$$\frac{\frac{x : a \rightarrow a \vdash x : a \rightarrow a}{\vdash (\lambda x.x) : (a \rightarrow a) \rightarrow (a \rightarrow a)} \text{ (abs)} \quad \frac{x : a \vdash x : a}{\vdash (\lambda x.x) : a \rightarrow a} \text{ (abs)}}{\vdash (\lambda x.x)(\lambda x.x) : a \rightarrow a} \text{ (app)}$$

# Conclusão

Dúvidas?

Nesta aula foi apresentado:

- O que são e para que servem sistemas de tipos em linguagens de programação.
- O que são e para que servem sistemas de tipos *formais* em linguagens de programação.