

Grupo de Pesquisa em Linguagens de Programação, Verificação e Engenharia de Sistemas

Elton Máximo
Glauber Cabral
Leonardo Reis
Rodrigo Ribeiro

Departamento de Computação e Sistemas (DECSI)

18 de Junho, 2015

Projetos

- 1 Desenvolvimento de Software Correto por Construção
- 2 Modularização e Extensibilidade de Linguagens
- 3 Elton'work
- 4 Correção e Avaliação Automáticas em Sistemas MOOCs

Sumário

- 1 Desenvolvimento de Software Correto por Construção
- 2 Modularização e Extensibilidade de Linguagens
- 3 Elton'work
- 4 Correção e Avaliação Automáticas em Sistemas MOOCs

Testes e Correção de Software



*“Testing can only show the presence,
not the absence of bugs.”*

Verificação Formal

(Assignment Axiom)

$$\frac{}{\{Q[E/id]\} \text{ id} = E; \{Q\}}$$

(Conditional Rule)

$$\frac{\{P \wedge E\} S_1 \{Q\} \quad \{P \wedge \neg E\} S_2 \{Q\}}{\{P\} \text{ if } (E) \{S_1\} \text{ else } \{S_2\} \{Q\}}$$

(Sequencing Rule)

$$\frac{\{P\} S_1 \{R\} \quad \{R\} S_2 \{Q\}}{\{P\} S_1 S_2 \{Q\}}$$

(Pre-strengthening, Post-weakening)

$$\frac{P \Rightarrow P' \quad \{P'\} S \{Q'\} \quad Q' \Rightarrow Q}{\{P\} S \{Q\}}$$

Proof Tableaux

$\{P_1\}$	$c_1;$	$\{P_0\}$
$\{P_2\}$	$c_2;$	$\{Q_1\}$
	$.$	$\{Q_2\}$



Teoria de Tipos

$$\frac{x : \sigma \in \Gamma \quad \tau \sqsubseteq \sigma}{\Gamma \vdash x : \tau} \text{ (TVar)}$$

$$\frac{\Gamma \vdash e : \tau' \rightarrow \tau \quad \Gamma \vdash e' : \tau'}{\Gamma \vdash ee' : \tau} \text{ (TApp)}$$

Testing can prove the absence of bugs, if we reduce program's weirdness.

Pesquisa: Aplicações de Teoria de Tipos

- Software correto por construção.
 - ▶ Especificações expressas como tipos. Verificação de correção feita pelo compilador.
 - ▶ Trabalhos realizados: Intepretadores, algoritmos e estruturas de dados.
 - ▶ Em andamento: Sistemas de tipos para verificação de propriedades de sistemas embarcados.
- Formalização
 - ▶ Uso de assistentes de provas para demonstração de propriedades de formalismos como sistemas de tipos.
 - ▶ Construção de provas de terminação de algoritmos sem efeitos colaterais em linguagens funcionais.

Sumário

- 1 Desenvolvimento de Software Correto por Construção
- 2 Modularização e Extensibilidade de Linguagens**
- 3 Elton'work
- 4 Correção e Avaliação Automáticas em Sistemas MOOCs

Era da Produtividade



- Foco na eficiência do programador
- DSLs como uma alternativa para melhorar a eficiência do programador
- Linguagens extensíveis como mecanismo para implementar e usar DSLs



O que são Linguagens Extensíveis?

- Linguagens extensíveis são linguagens que permitem estender a própria sintaxe concreta

```
1 package syntactic;
2 public sugar Pair {
3     context-free syntax
4     "(" JavaType "," JavaType ")"
5         → JavaType
6     "(" JavaExpr "," JavaExpr ")"
7         → JavaExpr
8     ...
9 }
```

SugarJ defining syntax

```
1 import syntactic.Pair;
2 public class Test {
3     private (String, Integer) p
4         = ("12", 34);
5 }
```

Using Pair syntax

O que são Linguagens Extensíveis?

- Linguagens extensíveis são linguagens que permitem estender a própria sintaxe concreta

```
1 package syntactic;
2 public sugar Pair {
3     context-free syntax
4     "(" JavaType "," JavaType ")"
5         → JavaType
6     "(" JavaExpr "," JavaExpr ")"
7         → JavaExpr
8     ...
9 }
```

SugarJ defining syntax

```
1 import syntactic.Pair;
2 public class Test {
3     private (String,Integer) p
4         = ("12", 34);
5 }
```

Using Pair syntax

Como Essas Características Dinâmicas Afetam o Parsing?

- Necessidade de modificar o parser de forma dinâmica, durante a análise da entrada

```
1 package syntactic;
2 public sugar Pair {
3     context-free syntax
4     ("JavaType ", " JavaType") "
5         → JavaType
6     ("JavaExpr ", " JavaExpr") "
7         → JavaExpr
8     ...
9 }
```

SugarJ defining syntax

```
1 import syntactic.Pair;
2 public class Test {
3     private (String, Integer) p
4         = ("12", 34);
5 }
```

Using Pair syntax

As Teorias de Parsing Suportam Modificação Dinâmica?

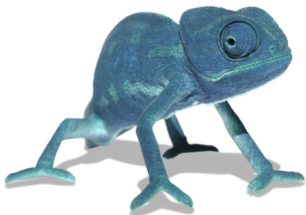
- Principais avanços recentes na área não tratam de modificações dinâmicas
 - ▶ PEG, LL(*), Adaptive LL(*), SGLR, YAKKER
- Trabalhos que lidam com modificação dinâmica das regras têm eficiência questionável ou não apresentam algoritmos de parsing
 - ▶ Adaptable Grammar de Christiansen; RAG; Parsing Reflective Grammars;
 - ▶ AMG; Dynamic Grammars; Evolving Grammars



Adaptable Parsing Expression Grammars



- Extensão de Parsing Expression Grammar;
- Modelo que permite modificações no conjunto de regras dinamicamente.



A Pesquisa

- Desenvolvimento de um gerador automático de analisador sintático baseado em APEG:
 - ▶ Implementação eficiente;
 - ▶ Tratamento de erros;
 - ▶ Construção automática de AST e metaprogramação;
 - ▶ provas de propriedades;
- Análise (métricas) de uso de DSLs em sistemas;
- Formalismos e mecanismos para especificação modular de linguagens
 - ▶ o que é modularização no contexto de especificação de linguagens?
 - ▶ especificação de sintaxe e semântica;
 - ▶ implementação de DSLs como bibliotecas.

Sumário

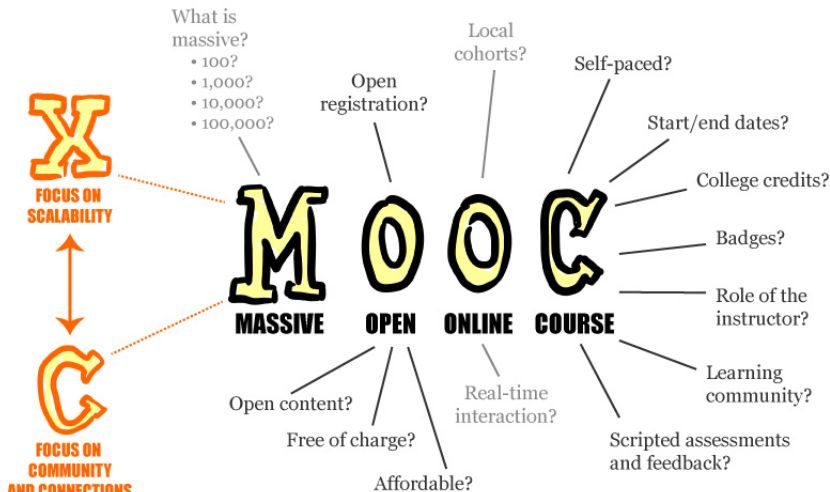
- 1 Desenvolvimento de Software Correto por Construção
- 2 Modularização e Extensibilidade de Linguagens
- 3 Elton'work**
- 4 Correção e Avaliação Automáticas em Sistemas MOOCs

Espaço do Elton

Sumário

- 1 Desenvolvimento de Software Correto por Construção
- 2 Modularização e Extensibilidade de Linguagens
- 3 Elton'work
- 4 Correção e Avaliação Automáticas em Sistemas MOOCs

MOOCs



Correção Automáticas de Programas em MOOCs

Correção Automática

- Suporte a linguagens funcionais: *Haskell*, *Scala*, ...
- Geração de valores de teste automaticamente
- Sistemas atuais: testes caixa preta
- É preciso fornecer melhor retorno dos erros aos alunos!



Avaliação Automáticas de Programas em MOOCs

Avaliação Automática

- Retorno indicativo dos erros no código
- Sugestões de alterações para corrigir o código
- Medir progresso do estudante

