



Descrição e execução de EFSMs

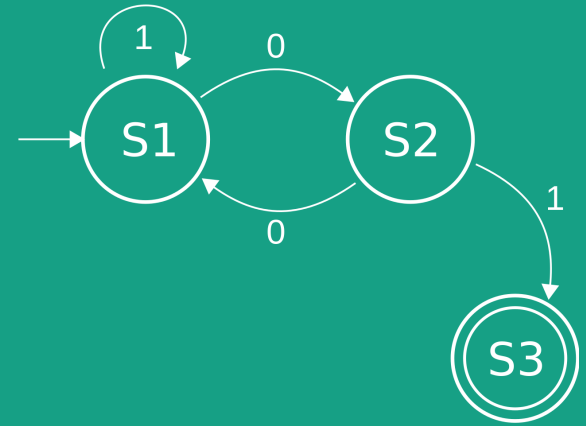
Filipe Arruda
Raisa Brito
Rodrigo Folha

Representar cada propriedade de uma **EFSM** em código:

- Memória;
- Estados;
- Transições;
- Eventos;
- Guardas;
- Ações.

Além da adoção de conceitos e funcionalidades adicionais, tais como Herança e Casamento de Padrões

Tupi



→ Inferir tipos da JVM / Compilação para .java

- Groovy
- Scala
- Java...

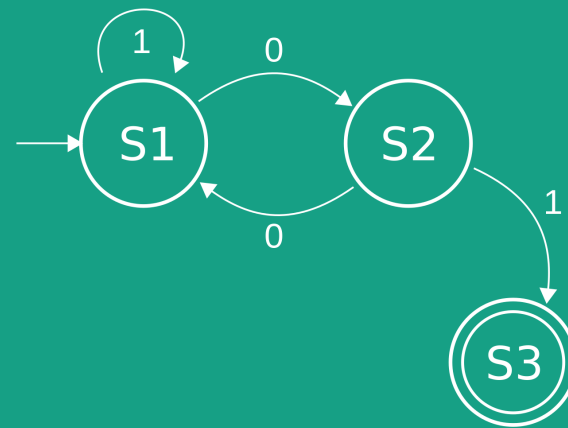
→ Validações / checagens

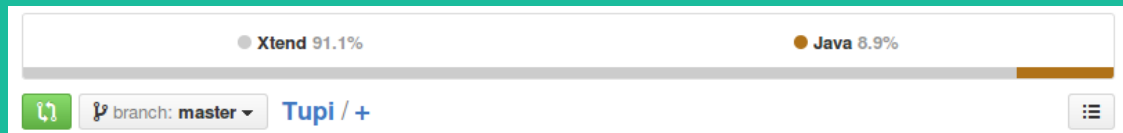
- Na própria IDE, durante codificação

→ Visualização das EFSMs

- Geração de arquivos .dot
- Graphviz

Objetivos





→ Lambdas

```
val compareParamSizes = [ Event a, Event b |  
  a.parameters?.variablesDecl?.size != b.parameters?.variablesDecl?.size  
]
```

→ Inferência de tipos

```
def text(EventsDecl e){  
  'Events'  
}
```

→ Abordagens funcionais

```
transition.originStates.map[x | ''' + x + '''].reduce[a,b| a + ', ' + b]  
events.filter[Event e|e.name.toLowerCase.equals("start")]
```


Xtext

Desenvolvimento

fmca / Tupi


Stargazers

All 1 You know 0



Leonardo Lucena

IFRN

 Follow



Leonardo Lucena

Área de Atuação: Engenharia de Software
Campus: Natal - Central

 leonardo.lucena@ifrn.edu.br 

AVISOS

Disciplinas:
Programação de Computadores
Programação Web
Paradigma e Linguagens de Programação



Compilação

Inferência de tipos na JVM e compilação .java

Machine

Memory

States

Guards

Actions

Events

Class

Attributes

Constructor

Methods

Validações

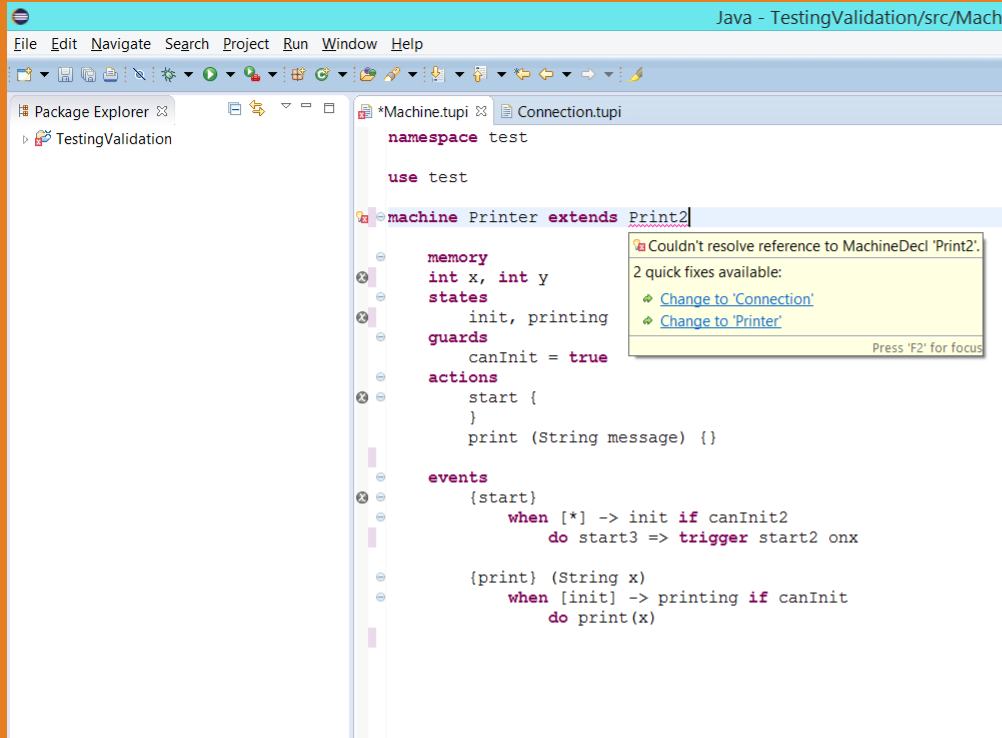
Checagens na IDE

Checagens

- Check Referências Cruzadas;
 - ◆ Machine
 - ◆ **Action**
 - ◆ **Events**
 - ◆ Memory
 - ◆ Guards
 - ◆ Status
 - ◆ State
- Check Declarações duplicadas
 - ◆ Memory
 - ◆ State
 - ◆ Actions
 - ◆ Events

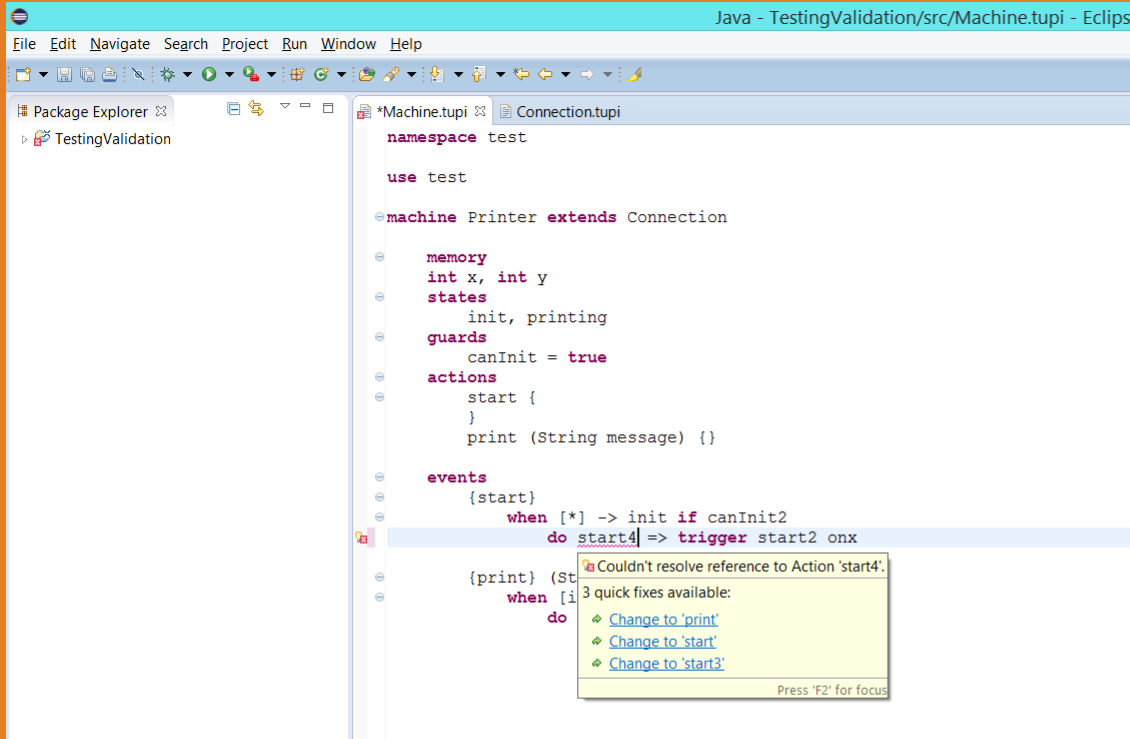
Checagem Referências Cruzadas

EXEMPLO



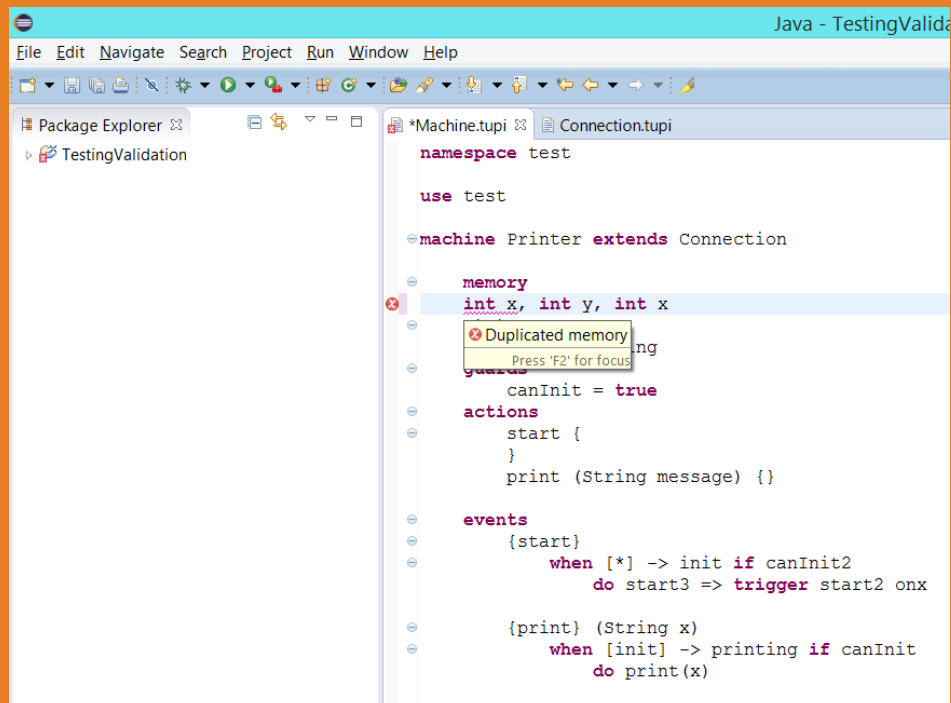
Checagem Referências Cruzadas

EXEMPLO



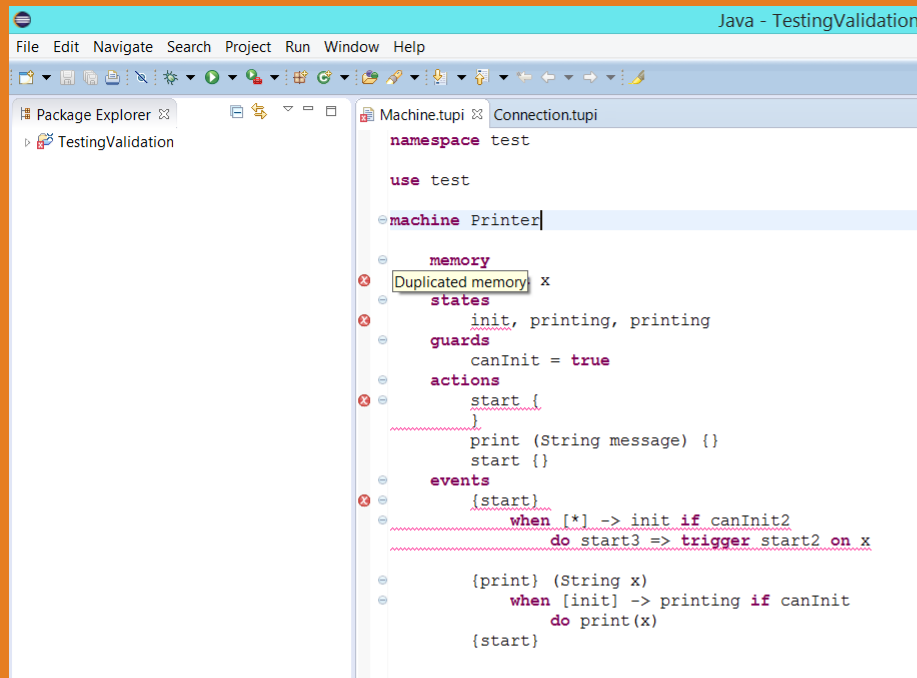
Checagem Duplicação

EXEMPLO



Checagem Duplicação

EXEMPLO



The screenshot shows the TUP (Temporal Unity Programming) environment. The Package Explorer on the left shows a project named 'TestingValidation'. The main editor displays a file named 'Machine.tupi' with the following code:

```
namespace test

use test

machine Printer

    memory
    states
    init, printing, printing
    guards
    canInit = true
    actions
    start {
        print (String message) {}
        start {}
    }
    events
    {start}
    when [*] -> init if canInit2
    do start3 => trigger start2 on x

    {print} (String x)
    when [init] -> printing if canInit
    do print(x)
    {start}
```

A red box highlights the line `memory` in the `Printer` machine definition, with a tooltip indicating a 'Duplicated memory' error. The error is also indicated by a red 'x' icon in the left margin next to the `memory` keyword.

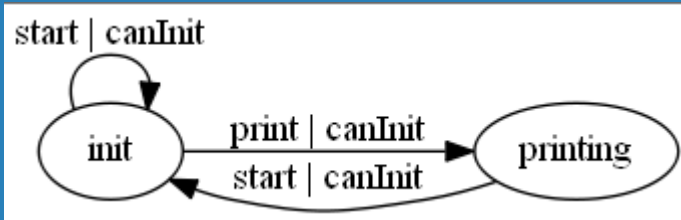
Geração .dot

Visualização das máquinas de estado

- Dígrafos podem representar SFM
- Visualização gráfica
- Fluxo de caminhos pode ser visualizado mais facilmente
- Padrão amplamente utilizado

Geração

```
digraph Printer {  
    rankdir=LR;  
    //states  
    init;  
    printing;  
    //edges  
    init->init [label="start | canInit"];  
    printing->init [label="start | canInit"];  
    init->printing [label="print | canInit"];  
}
```



Trabalhos Futuros

Concorrência, Event pool...

Trabalhos futuros

- Concorrência
- Publish-subscriber pattern
- Event pools
- Edição via UI (plugin)