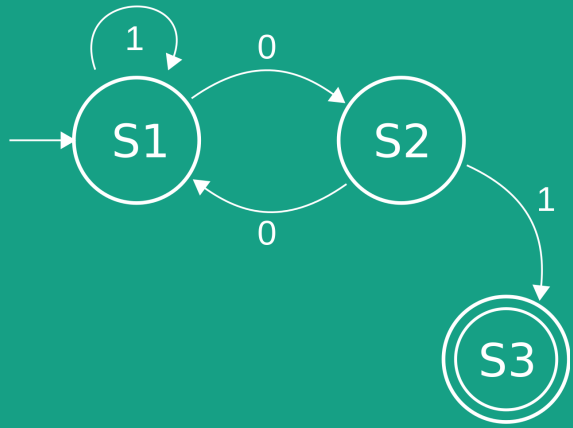
Tupi

Descrição e execução de EFSMs

Filipe Arruda
Raisa Brito
Rodrigo Folha

Finite-State Machines

- **Vantagens**
  - Estados possuem comportamentos bem definidos
  - É fácil modelar situações em novos estados (State Pattern)
  - Conjunto de transições é finito
- **Limitações**
  - Quantidade de estados
  - Turing Universal?
  - Condições de transições rígidas



Extended Finite-State Machines

M=(*Q, Σ1, Σ2, I, V, Λ*)

Q -> Conjunto de estados finitos, simples ou compostos

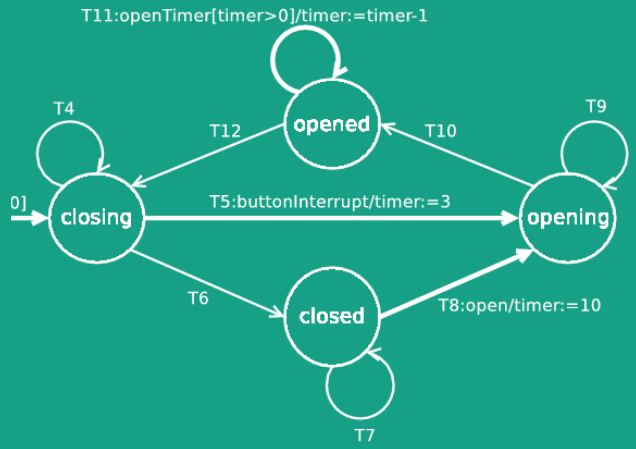
Σ1 -> Conjunto finito de eventos

Σ2 -> Conjunto de ações

I -> Conjunto de estados iniciais $\subset$ Q

***V* -> Conjunto de variáveis globais**

Λ -> Conjunto de transições



Trabalhos Relacionados

1. Expressões regulares. Código para várias linguagens (C, D, Java, Go, Ruby, C#)
2. Transforma FSM em arquivos Java(State Pattern).
3. Ferramenta de modelagem. Componentes em C++

1. Ragel State Machine Compiler
2. *The State Machine Compiler*
3. Rational Rose® RealTime

Objetivos

Linguagem & Visualização

Linguagem

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

- ➔ Mémoria;
- ➔ Estados;
- ➔ Transições;
- ➔ Evento;
- ➔ Guardas;
- ➔ Ações.

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

```
use tupi.list as List

machine Stack

    memory
        List list

    states
        empty, notempty

    guards
        init = true,
        hasMore1Element = list.size > 1,
        pushAllowed = true,
                        hasNoMore1Element    =    !
hasMore1Element
    actions
        initialize
            trigger {START} on list

        addElement [int x]
            trigger {add} on list with [x]
            peek = list.last

        deleteElement
            trigger {delete} on list
            peek = list.last
```

```
events
    {START}
        [*] -> empty                    | init
            initialize [limit]

    {PUSH} [int x]
        [*empty] -> notempty            | pushAllowed
            addElement [x]

    {POP}
        [notempty] -> empty             | hasNoMore1Element
            deleteElement

        [notempty] -> notempty          | hasMore1Element
            deleteElement
```

Linguagem

Sintaxe

```
machine SizeLimitedStack extends Stack
  memory
    int limit

  states
    full

  guards
    pushAllowed = list.size < limit,
    newpush = !pushAllowed
  actions
    initialize [int x]
      super.initialize
      limit = x

  events
    {PUSH} [int x]
      [*] -> full | newpush
        addElement [x]

    {POP}
      [full] -> empty | hasNoMore1Element
        deleteElement
      [full] -> notempty | hasMore1Element
        deleteElement
```

Linguagem

Herança

Exemplo

- ➔ Dado um evento PUSH
- ➔ Se pushAllowed == true
- ➔ Tanto *empty* quanto *notempty* casam padrão.
  - ◆ *estado atual empty* -> transita para *notempty*
  - ◆ *estado atual notempty* -> transita para *notempty*

```
{PUSH} [int x]
        [*empty] -> notempty    | pushAllowed
           addElement [x]
```

Linguagem

Pattern matching

Visualização

Graphviz (.dot)

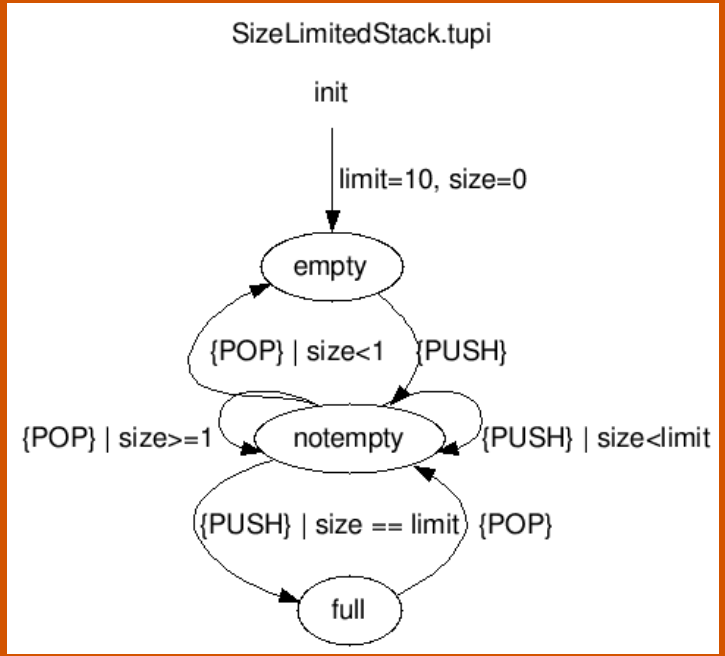
```
digraph g {
	{
	   init[shape=plaintext]
	}

	init -> empty [label=" limit=10, size=0"]
	   empty -> notempty [label= "{PUSH}"];

	notempty -> full [label="{PUSH} | size == limit"];
	notempty -> notempty [tailport=ne label="{PUSH} | size<limit"];
	notempty -> notempty [tailport=nw label=" {POP} | size>=1"];
	notempty -> empty [tailport=nw label=" {POP} | size<1  "];

	full -> notempty[label="  {POP}"];

}
```



Implementação

Frameworks, EBNF & Progresso atual

Framework Xtext

- ➔ Open Source
- ➔ Linguagens de programação e DSLs
- ➔ Parser & Model Generator
- ➔ Suporte por IDE

- ➔ Coloração de sintaxe
- ➔ “Autocomplete”

EBNF

BNF: Comando ::= ComConcreto | ComConcreto Comando

EBNF: Comando::= ComConcreto+

Operadores: ?, *, +

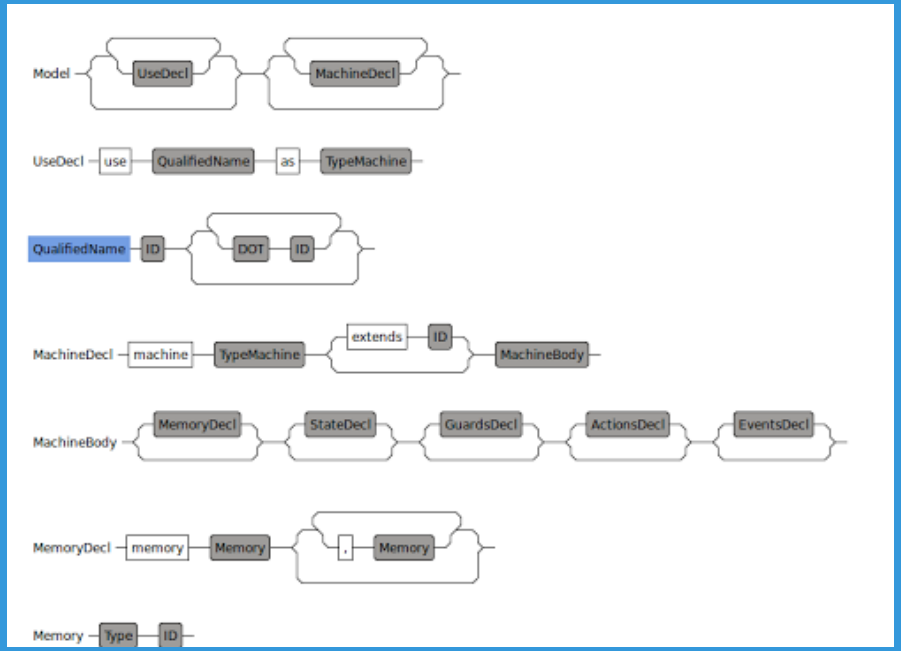
EBNF & XText

Usa ANTLR [algoritmo top-down LL(*)] que não permite recursão à esquerda.

```
Expression :
        Expression '+' Expression |
        Expression '-' Expression |
        INT;
```

```
Addition :
        Multiplication ('+' Multiplication)*;
Multiplication:
        NumberLiteral ('*' NumberLiteral)*;
NumberLiteral:
        INT;
```

Tupi EBNF


Model
UseDecl
MachineDecl
UseDecl
use
QualifiedName
as
TypeMachine
QualifiedName
ID
DOT
ID
MachineDecl
machine
TypeMachine
extends
ID
MachineBody
MachineBody
MemoryDecl
StateDecl
GuardsDecl
ActionsDecl
EventsDecl
MemoryDecl
memory
Memory
,
Memory
Memory
Type
ID

Demonstração