
Centro de Informática
UFPE
Universidade Federal de Pernambuco
CemPilha
Paradigmas de Linguagem de Programação
Discentes:
Jóison Oliveira Pereira
Maria Fernanda Sales
Ricardo Pereira

Overview

CEMPILHA

- **Expansão da LF1:**

  Adição da estrutura Pilha e seus métodos:

  Push, Pop, Top, EmptyStack,Inverter, SumAll.

BNF

Valor ::= ValorConcreto

ValorConcreto ::= ValorInteiro

|ValorBooleano

| ValorString

| ValorPilha

| ValorEmptyStack

BNF

```
ExpUnaria ::= "-" Expressao
            | "not" Expressao
            | "length" Expressao
            | "pop" Expressao
            | "top" Expressao
            | "sumAll" Expressao
            | "inverter" Expressao
```

Centro de Informática
UFPE

BNF

```
ExpBinaria ::= Expressao "+" Expressao
             | Expressao "-" Expressao
             | Expressao "and" Expressao
             | Expressao "or" Expressao
             | Expressao "==" Expressao
             | Expressao "++" Expressao
             | Expressao "push" "(" Expressao ")"
```

Sistemas de Tipos

Regras de Tipagem Implementadas

Centro de Informática UFPE

Regras:

1. ***push** exige compatibilidade entre o valor e o tipo interno da pilha*
2. ***emptyStack** é polimórfica → tipo definido no primeiro push*
3. ***sumAll** permitido apenas para pilhas de inteiros*
4. *Erros semânticos detectados em tempo de verificação*

```java
@Override
protected boolean checaTipoElementoTerminal(AmbienteCompilacao ambiente)
        throws VariavelNaoDeclaradaException, VariavelJaDeclaradaException {

    Tipo tipo = getExp().getTipo(ambiente);

    if (!(tipo instanceof TipoPilha)) {
        System.err.println("Erro de tipo: 'sumAll' esperado sobre uma pilha. Encontrado: "
                + tipo.getNome());
        return false;
    }

    TipoPilha tipoPilha = (TipoPilha) tipo;
    if (!tipoPilha.getSubTipo().equals(TipoPrimitivo.INTEIRO)) {
        System.err.println("Erro de tipo: 'sumAll' funciona apenas com pilha de inteiros. Encontrado: "
                + tipoPilha.getSubTipo().getNome());
        return false;
    }

    return true;
}
```

Avaliação e Imutabilidade

Avaliar

Centro de Informática UFPE

Estratégia de Avaliação:

1. ***Nenhuma** operação modifica a pilha existente*
2. *Cada operação retorna uma **nova** instância de pilha*
3. *Implementação baseada em cópia/clonagem controlada*
4. *Preserva o estado das variáveis, conforme o paradigma funcional*

```java
@Override
public Valor avaliar(AmbienteExecucao ambiente)
        throws VariavelNaoDeclaradaException, VariavelJaDeclaradaException {

    // Avalia as sub-expressões (herdadas)
    ValorPilha pilha = (ValorPilha) getEsq().avaliar(ambiente);
    Valor elemento = (Valor) getDir().avaliar(ambiente);

    // Retorna a NOVA pilha resultante do push (imutabilidade)
    return pilha.push(elemento); // Assumindo que o método em ValorPilha é 'push'
}
```

Pilhas de Pilhas

```
let var s = emptyStack in
  let var s1 = push(s, "c") in
    let var s2 = push(s1, "R") in
      let var v2 = pop(s1) in
        let var v3 = push(s1, top(s2)) in
          let var v4 = push (v3, "a") in
            let var v5 = inverte (push(v4, "oi")) in
              let var v6 = push (s2, "Topo") in
                let var pilha = emptyStack in
                  let var pilha2 = push (pilha, s1) in
                    let var pilha3 = push (pilha2, v5) in
                      let var pilha4 = push (pilha3, s2) in
                        let var topP = top(pilha4) in
                          pilha4
```

```
[INFO]
[INFO] --- exec:3.6.2:java (default-cli) @ Funcional1 ---
Funcional 1 PLP Parser Version 0.0.1:  Reading from file input . . .
Funcional 1 PLP Parser Version 0.0.1: Funcional1 program parsed successfully.
Funcional 1 PLP Parser Version 0.0.1: running...
Funcional 1 PLP Parser Version 0.0.1: Pilha= [Pilha: ["c"], Pilha: ["oi", "a", "R", "c"], Pilha: ["c", "R"]]
[INFO] ------------------------------------------------------------------------
```

SumAll

```
let var s = emptyStack in
  let var s1 = push(s, 3) in
    let var s2 = push(s1, 5) in
      let var s3 = push(s2, 7) in
        let var total = sumAll(s3) in
          total
```

```
[INFO] --- exec:3.6.2:java (default-cli) @ Funcional1 ---
Funcional 1 PLP Parser Version 0.0.1:  Reading from file input . . .
Funcional 1 PLP Parser Version 0.0.1: Funcional1 program parsed successfully.
Funcional 1 PLP Parser Version 0.0.1: running...
Funcional 1 PLP Parser Version 0.0.1: resultado=15
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Top

```
let var s = emptyStack in
  let var s1 = push(s, 3) in
    let var s2 = push(s1, 5) in
      let var s3 = top(s2) in

        s3
```

```
[INFO] --- exec:3.6.2:java (default-cli) @ Funcional1 ---
Funcional 1 PLP Parser Version 0.0.1:  Reading from file input . . .
Funcional 1 PLP Parser Version 0.0.1: Funcional1 program parsed successfully.
Funcional 1 PLP Parser Version 0.0.1: running...
Funcional 1 PLP Parser Version 0.0.1: resultado=5
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

Checa tipo

```
let var s = emptyStack in
  let var s1 = push(s, 3) in
    let var s2 = push(s1, 5) in
      let var s3 = push(s2, "b") in
        let var total = sumAll(s3) in
          total
```

```
[INFO] --- exec:3.6.2:java (default-cli) @ Funcional1 ---
Funcional 1 PLP Parser Version 0.0.1:  Reading from file input . .
Erro de Tipo: 'push' em pilha(INTEIRO) com elemento STRING
Funcional 1 PLP Parser Version 0.0.1: Erro de tipo.
```

Conclusão

Principais Contribuições

- Extensão da LF1 com um novo tipo funcional
- Tratamento correto de polimorfismo na pilha vazia
- Implementação imutável e segura de operações sobre pilhas
- Integração completa com sintaxe, semântica e execução da LF1

Muito obrigado!