

Um Interpretador para **Exp1**

Implementação em Haskell

A Linguagem Exp1

- Inclui apenas valores constantes (literais) e operações sobre valores
- Valores e operações sobre inteiros, booleanos e string são admitidos
- Um programa é uma expressão
- Agrupa valores em tipos e uma verificação de tipos pode ser implementada

Sintaxe Abstrata de Exp1

Programa ::= Expressao

Expressao ::= Valor
 | ExpUnaria
 | ExpBinaria

Valor ::= ValorConcreto

ValorConcreto ::= ValorInteiro
 | ValorBooleano
 | ValorString

Sintaxe Abstrata de Exp1 (cont.)

ExpUnaria ::= "-" Expressao
 | **"not"** Expressao
 | **"length"** Expressao

ExpBinaria ::= Expressao "+" Expressao
 | Expressao "-" Expressao
 | Expressao **"and"** Expressao
 | Expressao **"or"** Expressao
 | Expressao "==" Expressao
 | Expressao "++" Expressao

Programas de Exp1

- Programa 1

4

- Programa 2

4 + 5

Sintaxe Abstrata de Exp1

data Programa = PROG Expressao

data Expressao = LITI Int
 | LITB Bool
 | LITS String

-- ExpUnaria
 | MINUS Expressao
 | NOT Expressao
 | LENGTH Expressao

Sintaxe Abstrata de Exp1 (cont.)

-- ExpBinaria

| SOMA Expressao Expressao

| SUBTRACAO Expressao Expressao

| AND Expressao Expressao

| OR Expressao Expressao

| IGUAL Expressao Expressao

| CONCATENA Expressao Expressao

Exemplos de programas Exp1

$p1 = \text{PROG (LITI 4)}$

$p2 = \text{PROG (SOMA (LITI 4) (LITI 5))}$

Entidades Semânticas

-- Valores

-- data Valor = V1 Int | V2 Bool | V3 String
data Valor = Int | Bool | String

-- showValor :: Valor -> String
-- showValor (V1 i) = show i
-- showValor (V2 b) = show b
-- showValor (V3 s) = show s

Função de Interpretação

`interprete :: Programa -> String`

`interprete (PROG e) = show (eval e)`

`-- Funcao de Avaliacao de Expressoes`

`eval :: Expressao -> Valor`

`eval (LITI i) = i`

`eval (LITB b) = b`

`eval (LITS s) = s`

`eval (NOT e) = not (eval e)`

`eval (SOMA e1 e2) = (eval e1) + (eval e2)`