

# **Universidade da Beira Interior**

## **Departamento de Informática**



### **Compilador da Linguagem Arith para MIPS**

Elaborado por:

Tiago Roxo, nº 37032

Joana Costa, nº 37606

**Professor Doutor Simão Melo de Sousa**

19 de novembro de 2018

# Conteúdo

<b>Conteúdo</b>	<b>i</b>
<b>1 Sintaxe Abstrata</b>	<b>1</b>
<b>2 Análise semântica</b>	<b>4</b>
2.1 Análise por Big-steps . . . . .	4
2.2 Análise por Small-steps . . . . .	7
2.2.1 Expressões . . . . .	7
2.2.2 Instruções . . . . .	7
2.3 Regras de tipagem . . . . .	11
<b>3 Aplicação de regras</b>	<b>16</b>
3.1 Análise por Big-steps . . . . .	16
3.2 Análise por Small-steps . . . . .	17
3.3 Exemplo de tipagem . . . . .	19
<b>4 Funcionalidades implementadas</b>	<b>21</b>
<b>5 Manual de utilizador</b>	<b>23</b>

# Sintaxe Abstrata

$e ::=$

c	constante
ident	identificador
( e )	expressão entre parênteses
$e_1$ op $e_2$	operador binário
$\neg$ e	negação
fun ident $\rightarrow$ e	função
$e_1$ $e_2$	aplicação
let ident = $e_1$ in e	ligações locais
arrayInt e	inicialização de vetor de inteiros
ident[e]	acesso a valor de index de vetor
inputInt e	leitura de inteiro
inputFloat e	leitura de flutuante
inputString e	leitura de string

$c ::=$

int
bool
string
float

op ::=

+	soma de inteiros
−	subtração de inteiros
*	multiplicação de inteiros
/	divisão de inteiros
+.	soma de flutuante
−.	subtração de flutuante
*.	multiplicação de flutuante
/.	divisão de flutuante
<	menor
>	maior
==	igual
<=	menor ou igual
>=	maior ou igual
<>	diferença
&&	e lógico
	ou lógico

opu ::=

++	incremento de uma unidade
--	decremento de uma unidade

opact ::=

+=	atualização e reatribuição de inteiros
-=	atualização e reatribuição de inteiros
/=	atualização e reatribuição de inteiros
*=	atualização e reatribuição de inteiros

$t ::=$

$\text{let ident} = e$	declaração e atribuição
$\text{ident} \leftarrow e$	atribuição
$\text{ident opu}$	operador unário
$\text{ident opact } e$	operador e atualizador
$\text{if } e \text{ then } t$	execução condicional
$\text{if } e \text{ then } t_1 \text{ else } t_2$	execução condicional
$t; t$	sequência
$( t )$	instrução
$\text{while } e_1 \{ t \}$	ciclo while
$\text{for } (e_1:e_2:e_3) \{ t \}$	ciclo for
$\text{ident}[e_1] \leftarrow e_2$	alteração de valor em index de vetor
$\text{outputInt } e$	apresentação de inteiro
$\text{outputFloat } e$	apresentação de flutuante
$\text{outputString } e$	apresentação de string
$\text{next}$	sem ação

# Análise semântica

## 2.1 Análise por Big-steps

Nas análises semânticas seguintes procura-se uma relação de expressão  $e$  para um valor  $v$ . Os valores  $v$  considerados foram:

$v ::=$

n	inteiro
pos	inteiro positivo
neg	inteiro negativo
b	booleano
s	string
f	flutuante
()	unit
c	valor das constantes consideradas
op	operador binário
opu	operador unário
opact	operador e atualizador
fun ident $\rightarrow$ e	função
n *	apontador para endereço de memória de inteiro

As provas seguintes serão obtidas via regras de inferência, com base nos axiomas definidos

$$\frac{}{c \xrightarrow{v} c} \quad \frac{}{ident \xrightarrow{v} ident} \quad \frac{}{op \xrightarrow{v} op}$$

$$\frac{}{opu \xrightarrow{v} opu} \quad \frac{}{opact \xrightarrow{v} opact}$$

$$\frac{}{fun(ident \rightarrow e) \xrightarrow{v} fun(ident \rightarrow e)}$$

$$\frac{e_1 \xrightarrow{v} n \quad e_2 \xrightarrow{v} pos \quad b = n > 0 \quad b \xrightarrow{v} true}{e_1 \xrightarrow{v} e_2}$$

$$\frac{e_1 \xrightarrow{v} n \quad e_2 \xrightarrow{v} neg \quad b = n < 0 \quad b \xrightarrow{v} true}{e_1 \xrightarrow{v} e_2}$$

$$\frac{e_1 \xrightarrow{v} (fun\ x \rightarrow e) \quad e_2 \xrightarrow{v} v_2 \quad e[x \leftarrow v_2] \xrightarrow{v} v}{e_1\ e_2 \xrightarrow{v} v}$$

$$\frac{e_1 \xrightarrow{v} true}{\neg e_1 \xrightarrow{v} false} \quad \frac{e_1 \xrightarrow{v} false}{\neg e_1 \xrightarrow{v} true}$$

$$\frac{op \xrightarrow{v} + \quad e_1 \xrightarrow{v} n_1 \quad e_2 \xrightarrow{v} n_2 \quad n = n_1 + n_2}{e_1\ op\ e_2 \xrightarrow{v} n} \quad \frac{op \xrightarrow{v} - \quad e_1 \xrightarrow{v} n_1 \quad e_2 \xrightarrow{v} n_2 \quad n = n_1 - n_2}{e_1\ op\ e_2 \xrightarrow{v} n}$$

$$\frac{op \xrightarrow{v} * \quad e_1 \xrightarrow{v} n_1 \quad e_2 \xrightarrow{v} n_2 \quad n = n_1 * n_2}{e_1 \text{ op } e_2 \xrightarrow{v} n} \quad \frac{op \xrightarrow{v} / \quad e_1 \xrightarrow{v} n_1 \quad e_2 \xrightarrow{v} n_2 \quad n = n_1 / n_2}{e_1 \text{ op } e_2 \xrightarrow{v} n}$$

$$\frac{op \xrightarrow{v} +. \quad e_1 \xrightarrow{v} f_1 \quad e_2 \xrightarrow{v} f_2 \quad f = f_1 + f_2}{e_1 \text{ op } e_2 \xrightarrow{v} f} \quad \frac{op \xrightarrow{v} -. \quad e_1 \xrightarrow{v} f_1 \quad e_2 \xrightarrow{v} f_2 \quad f = f_1 - f_2}{e_1 \text{ op } e_2 \xrightarrow{v} f}$$

$$\frac{op \xrightarrow{v} *. \quad e_1 \xrightarrow{v} f_1 \quad e_2 \xrightarrow{v} f_2 \quad f = f_1 * f_2}{e_1 \text{ op } e_2 \xrightarrow{v} f} \quad \frac{op \xrightarrow{v} /. \quad e_1 \xrightarrow{v} f_1 \quad e_2 \xrightarrow{v} f_2 \quad f = f_1 / f_2}{e_1 \text{ op } e_2 \xrightarrow{v} f}$$

$$\frac{op \xrightarrow{v} > \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 > c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b} \quad \frac{op \xrightarrow{v} >= \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 >= c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b}$$

$$\frac{op \xrightarrow{v} < \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 < c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b} \quad \frac{op \xrightarrow{v} <= \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 <= c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b}$$

$$\frac{op \xrightarrow{v} == \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 == c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b} \quad \frac{op \xrightarrow{v} <> \quad e_1 \xrightarrow{v} c_1 \quad e_2 \xrightarrow{v} c_2 \quad b = c_1 <> c_2}{e_1 \text{ op } e_2 \xrightarrow{v} b}$$

$$\frac{op \xrightarrow{v} \&\& \quad e_1 \xrightarrow{v} b_1 \quad e_2 \xrightarrow{v} b_2 \quad b = b_1 \&\& b_2}{e_1 \text{ op } e_2 \xrightarrow{v} b} \quad \frac{op \xrightarrow{v} || \quad e_1 \xrightarrow{v} b_1 \quad e_2 \xrightarrow{v} b_2 \quad b = b_1 || b_2}{e_1 \text{ op } e_2 \xrightarrow{v} b}$$



$$\frac{e_1 \xrightarrow{v} v_1 \quad e_2[ident \leftarrow v_1] \xrightarrow{v} v}{let\ ident = e_1\ in\ e_2 \xrightarrow{v} v}$$

## 2.2 Análise por Small-steps

### 2.2.1 Expressões

$$\frac{}{E, c \rightarrow E, c} \quad \frac{}{E, (e) \rightarrow E, e}$$

$$\frac{E, ident \rightarrow ident \quad E, e \xrightarrow{v} n}{E, let\ ident = arrayInt\ e \rightarrow E_1\{ident \mapsto n^*\}, next}$$

$$\frac{E, ident_1 \rightarrow ident_1 \quad E, ident_2 \rightarrow ident_2 \quad E, e \xrightarrow{v} n}{E\{ident_2 \mapsto n^*\}, let\ ident_1 = ident_2[e] \rightarrow E_1\{ident_1 \mapsto ident_2[e]\}, next}$$

### 2.2.2 Instruções

$$\frac{}{E, next; t \rightarrow E, t} \quad \frac{}{E, (t) \rightarrow E, t}$$

$$\frac{E, t_1 \xrightarrow{v} E_1, t'_1}{E, t_1; t_2 \rightarrow E_1, t'_1; t_2}$$

$$\frac{E, e \xrightarrow{v} v}{E, let\ ident = e \rightarrow E_1\{ident \mapsto v\}, next}$$

$$\frac{E\{ident \mapsto v_1\}, e \xrightarrow{v} v_2}{E\{ident \mapsto v_1\}, ident \leftarrow e \rightarrow E_1\{ident \mapsto v_2\}, next}$$

$$\frac{E, e \xrightarrow{v} true}{E, if\ e\ then\ t_1\ else\ t_2 \rightarrow E, t_1} \quad \frac{E, e \xrightarrow{v} false}{E, if\ e\ then\ t_1\ else\ t_2 \rightarrow E, t_2}$$

$$\frac{E, e \xrightarrow{v} true}{E, if\ e\ then\ t_1; t_2 \rightarrow E_1, t_1} \quad \frac{E, e \xrightarrow{v} false}{E, if\ e\ then\ t_1; t_2 \rightarrow E_1, t_2}$$

$$\frac{E, e_1 \xrightarrow{v} true}{E, while\ e_1\ \{t\} \rightarrow E_1, t; while\ e_1\ \{t\}} \quad \frac{E, e_1 \xrightarrow{v} false}{E, while\ e_1\ \{t\} \rightarrow E_1, next}$$

$$\frac{E, e_1 \xrightarrow{v} ident \quad E, opact \rightarrow + = \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_1 \leftarrow n_1 + n_2}{E\{e_1 \mapsto n_1\}, e_1\ opact\ e_2 \xrightarrow{v} E_1\{e_1 \mapsto n_1 + n_2\}, next}$$

$$\frac{E, e_1 \xrightarrow{v} ident \quad E, opact \rightarrow - = \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_1 \leftarrow n_1 - n_2}{E\{e_1 \mapsto n_1\}, e_1\ opact\ e_2 \xrightarrow{v} E_1\{e_1 \mapsto n_1 - n_2\}, next}$$

$$\frac{E, e_1 \xrightarrow{v} ident \quad E, opact \rightarrow * = \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_1 \leftarrow n_1 * n_2}{E\{e_1 \mapsto n_1\}, e_1\ opact\ e_2 \xrightarrow{v} E_1\{e_1 \mapsto n_1 * n_2\}, next}$$

$$\frac{E, e_1 \xrightarrow{v} ident \quad E, opact \rightarrow / = \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_1 \leftarrow n_1 / n_2}{E\{e_1 \mapsto n_1\}, e_1\ opact\ e_2 \xrightarrow{v} E_1\{e_1 \mapsto n_1 / n_2\}, next}$$

$$\frac{E, opu \rightarrow ++ \quad E, ident \rightarrow ident \quad E\{ident \mapsto n\}, n_1 = n + 1}{E, ident opu \rightarrow E\{ident \mapsto n_1\}, next}$$

$$\frac{E, opu \rightarrow -- \quad E, ident \rightarrow ident \quad E\{ident \mapsto n\}, n_1 = n - 1}{E, ident opu \rightarrow E\{ident \mapsto n_1\}, next}$$

$$\frac{E\{ident \mapsto n\}, e_1 \xrightarrow{v} ident \leftarrow n_1 \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_3 \xrightarrow{v} pos \quad E, n_1 \geq n_2 \xrightarrow{v} true}{E, for(e1 : e2 : e3) \{t_1\}; t_2 \rightarrow E_1\{ident \mapsto n_1 + n_3\}, t_2}$$

$$\frac{E\{ident \mapsto n\}, e_1 \xrightarrow{v} ident \leftarrow n_1 \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_3 \xrightarrow{v} pos \quad E, n_1 \geq n_2 \xrightarrow{v} false}{E, for(e1 : e2 : e3) \{t_1\}; t_2 \rightarrow \Delta, t_1; for(ident \leftarrow n1 + n3 : n2 : n3) \{t_1\}; t_2}$$

Onde  $\Delta$ ,

$$E_1\{ident \mapsto n_1 + n_3\}$$

$$\frac{E\{ident \mapsto n\}, e_1 \xrightarrow{v} ident \leftarrow n_1 \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_3 \xrightarrow{v} neg \quad E, n_1 \leq n_2 \xrightarrow{v} true}{E, for(e1 : e2 : e3) \{t_1\}; t_2 \rightarrow E_1\{ident \mapsto n_1 + n_3\}, t_2}$$

$$\frac{E\{ident \mapsto n\}, e_1 \xrightarrow{v} ident \leftarrow n_1 \quad E, e_2 \xrightarrow{v} n_2 \quad E, e_3 \xrightarrow{v} neg \quad E, n_1 \leq n_2 \xrightarrow{v} false}{E, for(e1 : e2 : e3) \{t_1\}; t_2 \rightarrow \Delta, t_1; for(ident \leftarrow n1 + n3 : n2 : n3) \{t_1\}; t_2}$$

Onde  $\Delta$ ,

$$E_1\{ident \mapsto n_1 + n_3\}$$

$$\frac{E, ident \rightarrow ident \quad E, e_1 \xrightarrow{v} n_1 \quad E, e_2 \xrightarrow{v} n_2}{E\{ident \mapsto n^*\}, ident[e_1] \leftarrow e_2 \rightarrow E_1\{ident[e_1] \mapsto n_2\}, next}$$

$$\frac{E, outputInt\ e \xrightarrow{v} () \quad E, e \xrightarrow{v} n}{E, outputInt\ e \rightarrow E, next} \quad \frac{E, outputFloat\ e \xrightarrow{v} () \quad E, e \xrightarrow{v} f}{E, outputFloat\ e \rightarrow E, next}$$

$$\frac{E, outputString\ e \xrightarrow{v} () \quad E, e \xrightarrow{v} s}{E, outputString\ e \rightarrow E, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputInt\ e \xrightarrow{v} n}{E, let\ ident = inputInt\ e \rightarrow E_1\{ident \mapsto n\}, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputFloat\ e \xrightarrow{v} f}{E, let\ ident = inputFloat\ e \rightarrow E_1\{ident \mapsto f\}, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputString\ e \xrightarrow{v} s}{E, let\ ident = inputString\ e \rightarrow E_1\{ident \mapsto s\}, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputInt\ e \xrightarrow{v} n_1}{E\{ident \mapsto n\}, ident \leftarrow inputInt\ e \rightarrow E_1\{ident \mapsto n_1\}, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputFloat\ e \xrightarrow{v} f_1}{E\{ident \mapsto f\}, ident \leftarrow inputFloat\ e \rightarrow E_1\{ident \mapsto f_1\}, next}$$

$$\frac{E, e \xrightarrow{v} () \quad E, inputString\ e \xrightarrow{v} s_1}{E\{ident \mapsto s\}, ident \leftarrow inputString\ e \rightarrow E_1\{ident \mapsto s_1\}, next}$$

## 2.3 Regras de tipagem

$\tau ::=$

int   bool   string	tipos de base
true   false   neg   pos	tipos particulares de tipos base
unit	tipo unit
$\tau$ *	tipo apontador
$\tau \rightarrow \tau$	tipo de uma função
$\alpha$	variáveis de tipo

$$\frac{\tau \leq \Gamma(\text{ident})}{\Gamma \vdash \text{ident} : \tau}$$

$$\frac{}{\Gamma \vdash n : \text{int}} \quad \frac{}{\Gamma \vdash b : \text{bool}} \quad \frac{}{\Gamma \vdash c : \text{char}}$$

$$\frac{}{\Gamma \vdash s : \text{string}} \quad \frac{}{\Gamma \vdash f : \text{float}}$$

$$\frac{\Gamma \vdash e_1 : \text{int} \quad b = n > 0 \quad b \xrightarrow{v} \text{true} \quad \Gamma \vdash e_1 : \text{int} \vdash e_2 \xrightarrow{v} \text{pos}}{\Gamma \vdash e_1 : \text{pos}}$$

$$\frac{\Gamma \vdash e_1 : \text{int} \quad b = n < 0 \quad b \xrightarrow{v} \text{true} \quad \Gamma \vdash e_1 : \text{int} \vdash e_2 \xrightarrow{v} \text{neg}}{\Gamma \vdash e_1 : \text{neg}}$$

$$\frac{\Gamma \vdash e_1 : \tau}{\Gamma \vdash \text{let ident} = e_1 : \tau}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma + ident : \tau \vdash ident \leftarrow e : \tau}$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 + e_2 : int} \quad \frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 - e_2 : int}$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 * e_2 : int} \quad \frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash e_1 / e_2 : int}$$

$$\frac{\Gamma \vdash e_1 : float \quad \Gamma \vdash e_2 : float}{\Gamma \vdash e_1 + . e_2 : float} \quad \frac{\Gamma \vdash e_1 : float \quad \Gamma \vdash e_2 : float}{\Gamma \vdash e_1 - . e_2 : float}$$

$$\frac{\Gamma \vdash e_1 : float \quad \Gamma \vdash e_2 : float}{\Gamma \vdash e_1 * . e_2 : float} \quad \frac{\Gamma \vdash e_1 : float \quad \Gamma \vdash e_2 : float}{\Gamma \vdash e_1 /. e_2 : float}$$

$$\frac{\Gamma \vdash e_1 : bool \quad \Gamma \vdash e_2 : bool}{\Gamma \vdash e_1 \&\& e_2 : bool} \quad \frac{\Gamma \vdash e_1 : bool \quad \Gamma \vdash e_2 : bool}{\Gamma \vdash e_1 || e_2 : bool}$$

$$\frac{\Gamma + ident : \tau_1 \vdash e : \tau_2}{\Gamma \vdash fun ident \rightarrow e : \tau_1 \rightarrow \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma + ident : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash let ident = e_1 in e_2 : \tau_2}$$

$$\frac{}{\Gamma \vdash \alpha > \alpha : bool} \quad \frac{}{\Gamma \vdash \alpha < \alpha : bool} \quad \frac{}{\Gamma \vdash \alpha >= \alpha : bool}$$

$$\frac{}{\Gamma \vdash \alpha <= \alpha : bool} \quad \frac{}{\Gamma \vdash \alpha == \alpha : bool} \quad \frac{}{\Gamma \vdash \alpha <> \alpha : bool}$$

$$\frac{\Gamma \vdash e : true \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{if } e \text{ then } t_1 \text{ else } t_2 : \tau_1} \quad \frac{\Gamma \vdash e : false \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{if } e \text{ then } t_1 \text{ else } t_2 : \tau_2}$$

$$\frac{\Gamma \vdash e : true \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{if } e \text{ then } t_1; t_2 : \tau_1} \quad \frac{\Gamma \vdash e : false \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{if } e \text{ then } t_1; t_2 : \tau_2}$$

$$\frac{\Gamma \vdash e_1 : true \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{while } e_1 \{t_1\}; t_2 : \tau_1} \quad \frac{\Gamma \vdash e_1 : false \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{while } e_1 \{t_1\}; t_2 : \tau_2}$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int \quad \Gamma \vdash e_3 : pos \quad \Delta \vdash e_1 >= e_2 : true \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash \text{for}(e_1 : e_2 : e_3) \{t_1\}; t_2 : \tau_1}$$

Onde  $\Delta$ ,

$$\Gamma \vdash e_1 : int, e_2 : int$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int \quad \Gamma \vdash e_3 : pos \quad \Delta \vdash e_1 \geq e_2 : false \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash for(e1 : e2 : e3) \{t_1\}; t_2 : \tau_2}$$

Onde  $\Delta$ ,

$$\Gamma \vdash e_1 : int, e_2 : int$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int \quad \Gamma \vdash e_3 : neg \quad \Delta \vdash e_1 \leq e_2 : true \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash for(e1 : e2 : e3) \{t_1\}; t_2 : \tau_1}$$

Onde  $\Delta$ ,

$$\Gamma \vdash e_1 : int, e_2 : int$$

$$\frac{\Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int \quad \Gamma \vdash e_3 : neg \quad \Delta \vdash e_1 \leq e_2 : false \quad \Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash for(e1 : e2 : e3) \{t_1\}; t_2 : \tau_2}$$

Onde  $\Delta$ ,

$$\Gamma \vdash e_1 : int, e_2 : int$$

$$\frac{\Gamma \vdash arrayInt : int \rightarrow \tau^* \quad \Gamma \vdash e : int}{\Gamma \vdash arrayInt \ e : \tau^*}$$

$$\frac{\Gamma \vdash ident : \tau^* \quad \Gamma \vdash e_1 : int \quad \Gamma \vdash e_2 : int}{\Gamma \vdash ident[e_1] \leftarrow e_2 : unit}$$



$$\frac{\Gamma \vdash \text{ident} : \tau^* \quad \Gamma \vdash e_1 : \text{int}}{\Gamma \vdash \text{ident}[e_1] : \text{int}}$$

$$\frac{\Gamma \vdash \text{outputInt} : \text{int} \rightarrow \text{unit} \quad \Gamma \vdash e : \text{int}}{\Gamma \vdash \text{outputInt } e : \text{unit}}$$

$$\frac{\Gamma \vdash \text{outputFloat} : \text{float} \rightarrow \text{unit} \quad \Gamma \vdash e : \text{float}}{\Gamma \vdash \text{outputFloat } e : \text{unit}}$$

$$\frac{\Gamma \vdash \text{outputString} : \text{string} \rightarrow \text{unit} \quad \Gamma \vdash e : \text{string}}{\Gamma \vdash \text{outputString } e : \text{unit}}$$

$$\frac{\Gamma \vdash e : \text{unit} \quad \Gamma \vdash \text{inputInt} : \text{unit} \rightarrow \text{int}}{\Gamma \vdash \text{let ident} = \text{inputInt } e : \text{int}}$$

$$\frac{\Gamma \vdash e : \text{unit} \quad \Gamma \vdash \text{inputFloat} : \text{unit} \rightarrow \text{float}}{\Gamma \vdash \text{let ident} = \text{inputFloat } e : \text{float}}$$

$$\frac{\Gamma \vdash e : \text{unit} \quad \Gamma \vdash \text{inputString} : \text{unit} \rightarrow \text{string}}{\Gamma \vdash \text{let ident} = \text{inputString } e : \text{string}}$$

# Aplicação de regras

## 3.1 Análise por Big-steps

$$\frac{\frac{\frac{op \xrightarrow{v} - \quad \frac{7 \rightarrow 7}{\quad} \quad \frac{1 \rightarrow 1}{\quad}}{7 - 1 \xrightarrow{v} 6} \quad \frac{\frac{\frac{\frac{fun\ y \rightarrow y - (-8) \xrightarrow{v} fun\ y \rightarrow y - (-8)}{\quad} \quad \frac{fun\ y \rightarrow y - (-8)}{\quad}}{fun\ y \rightarrow y - (-8)} \quad \frac{6 \rightarrow 6}{\quad} \quad \frac{\frac{op \xrightarrow{v} - \quad \frac{6 \rightarrow 6}{\quad} \quad \frac{-8 \rightarrow -8}{\quad}}{6 - (-8) \rightarrow 14}}{\frac{(fun\ y \rightarrow y - (-8))\ 6 \xrightarrow{v} 14}}{let\ x = 7 - 1\ in\ (fun\ y \rightarrow y - (-8))\ x \xrightarrow{v} 14}}$$

### 3.2 Análise por Small-steps

Exemplo 1:

$$\begin{array}{c}
\frac{\frac{\frac{}{op \rightarrow -} \quad \frac{\frac{}{5 \rightarrow 5}}{\quad} \quad \frac{\frac{}{4 \rightarrow 4}}{\quad}}{(5-4) \rightarrow 1} \quad \frac{}{2 \rightarrow 2}}{op \rightarrow +} \\
\frac{\frac{\frac{}{(fun\ y \rightarrow y - 4 + 2)} \quad \frac{}{5 \rightarrow 5}}{(5-4)+2 \rightarrow 3}}{2 \rightarrow 2} \quad \frac{}{(fun\ y \rightarrow y - 4 + 2)\ 5 \xrightarrow{v} 3} \\
\frac{\frac{}{op \xrightarrow{v} >} \quad \frac{}{5 \rightarrow 5} \quad \frac{}{10 \rightarrow 10}}{E, 5 > 10 \xrightarrow{v} false} \quad \frac{\frac{}{E, let\ z = 2\ in\ (fun\ y \rightarrow y - 4 + z)\ 5 \xrightarrow{v} 3}}{E, let\ x = (let\ z = 2\ in\ ((fun\ y \rightarrow y - 4 + z)\ 5)) \xrightarrow{v} E\{x \mapsto 3\}; next} \\
\frac{}{E, let\ x = if\ 5 > 10\ then\ 4\ else\ let\ z = 2\ in\ (fun\ y \rightarrow y - 4 + z)\ 5 \xrightarrow{v} E\{x \mapsto 3\}; next}
\end{array}$$

Exemplo 2:

$$\begin{array}{c}
 \Delta \quad \frac{\frac{op \xrightarrow{v} >= \quad \overline{2 \rightarrow 2} \quad \overline{3 \rightarrow 3}}{2 >= 3 \xrightarrow{v} false} \quad \frac{\overline{E\{i \mapsto n\}, output\_int\ i \xrightarrow{v} next} \quad \frac{\overline{E, next; t \rightarrow E, t} \quad \Theta}{E, next; for(i \leftarrow 3 : 3 : 1) do output\_int\ i; t \xrightarrow{v} E, t}}{E, output\_int\ i; for\ (i \leftarrow 3 : 3 : 1) do output\_int\ i; t \xrightarrow{v} E, t}}{E, for(i \leftarrow 2 : 3 : 1) do output\_int\ i; t \xrightarrow{v} E, t}
 \end{array}$$

Onde  $\Delta$ ,

$$\frac{\frac{\overline{2 \rightarrow 2}}{E\{i \mapsto n\}, i \leftarrow 2} \quad \overline{3 \rightarrow 3} \quad \frac{\overline{1 \rightarrow 1} \quad \frac{op \rightarrow > \quad \overline{1 \rightarrow 1} \quad \overline{0 \rightarrow 0}}{1 > 0 \rightarrow true}}{E, 1 \rightarrow 1}$$

Onde  $\Theta$ ,

$$\frac{\frac{\overline{3 \rightarrow 3}}{E\{i \mapsto n\}, i \leftarrow 3} \quad \overline{3 \rightarrow 3} \quad \frac{op \rightarrow > \quad \overline{1 \rightarrow 1} \quad \overline{0 \rightarrow 0}}{1 > 0 \rightarrow true} \quad \frac{op \xrightarrow{v} >= \quad \overline{3 \rightarrow 3} \quad \overline{3 \rightarrow 3}}{3 >= 3 \xrightarrow{v} true} \quad \overline{E, next; t \rightarrow E, t}}{E, for\ (i \leftarrow 3 : 3 : 1) do output\_int\ i; t \rightarrow E, t}$$

### 3.3 Exemplo de tipagem

Exemplo 1:

$$\frac{\overline{\emptyset \vdash 5 : int} \quad \overline{\emptyset \vdash 10 : int}}{\{5 : int, 10 : int\} \vdash 5 > 10 : bool} \quad \frac{\overline{\emptyset \vdash 2 : int} \quad \frac{\overline{\{z : int\} \vdash y - 4 + z : int} \quad \overline{\emptyset \vdash 5 : int}}{\{z : int\} \vdash (\text{fun } y \rightarrow y - 4 + z) : int \rightarrow int}}{\emptyset \vdash let z = 2 in ((\text{fun } y \rightarrow y - 4 + z) 5 : int)} \\ \overline{\emptyset \vdash let x = if 5 > 10 then 4 else let z = 2 in (\text{fun } y \rightarrow y - 4 + z) 5 : int}$$

Onde  $\Delta$ ,

$$\{y : int, z : int, 4 : int\} \vdash (y - 4) + z : int$$

Exemplo 2:

$$\begin{array}{c}
 \frac{}{\emptyset \vdash 2 : int} \quad \frac{\frac{\frac{}{\{z : Gen(int, \emptyset)\} \vdash z : int} \quad \frac{\frac{int \leq \alpha}{\{y : \alpha, z : int\} \vdash y : int}}{\{y : \alpha\} \vdash y : \alpha}}{\emptyset \vdash fun\ y \rightarrow y : \alpha \rightarrow \alpha}}{\frac{\{z : Gen(int, \emptyset)\} \vdash (fun\ y \rightarrow y)\ z : int}{\emptyset \vdash let\ z = 2\ in\ (fun\ y \rightarrow y)\ z : int}}
 \end{array}$$

# Funcionalidades implementadas

As funcionalidades implementadas são apresentadas na seguinte listagem:

- **Declaração e atribuição:** As variáveis utilizadas no nosso projeto são declaradas por recurso à instrução "let ident = e", que associa o valor "e" ao "ident", usando uma Hash-table. A atribuição exige que a variável tenha sido previamente declarada e o novo valor a atribuir terá de ser concordante com o tipo da variável. O nosso projeto permite a declaração e atribuição de variáveis do tipo string, float, int e bool. A expressão respetiva a ligações locais é também disponibilizada neste projeto.
- **Operações:** O nosso projeto disponibiliza operações aritméticas entre int e float reque-rendo para tal o uso de operadores distintos ("+" para int, "+" para float). A tipagem existente exige que as operações aritméticas nunca sejam possíveis entre int e float. Os operadores lógicos (&& e ||) existentes são passíveis de ser usados entre variáveis do tipo bool. O operador negação é aplicável a estas variáveis, invertendo o seu valor. A abordagem "preguiçosa" foi a usada na avaliação de expressões com estes operadores. A comparação entre valores é possível, usando os operadores disponibilizados, assumindo que ambos têm o mesmo tipo. O operador unário de incrementação e decrementação é apenas aplicável a variáveis do tipo int, com o requisito extra de exigir a sua declaração prévia. Os operadores aplicáveis à atualização de valores seguem o modelo "ident += e", onde "ident" terá de ter sido previamente declarado e "e" deverá ter o mesmo tipo que "ident"; estes operadores são apenas aplicáveis a variáveis do tipo int.
- **Execução condicional:** A execução de instruções condicionais do tipo "if e then t1 else t2" exige que a expressão "e" seja bool. A migração para as instruções 1 e 2, em MIPS, é conseguida por recurso a *labels* mediante o valor resultante de "e"; o término de qualquer uma das instruções culmina na migração para uma outra *label* chamada "condContinuar".
- **Sequência:** A execução sequencial de instruções é distinguível por uso de ";" entre estas. Uma instrução pode ser delimitada por parênteses, caso o utilizador assim deseje; a mesma abordagem é usada para as expressões.
- **Ciclos:** O ciclo *for* é disponibilizado no nosso projeto tendo a capacidade de ser crescente ou decrescente. A nomenclatura usada para este exige uma atribuição e duas expressões do tipo int. Caso o valor da expressão usada na atribuição e a segunda expressão

do *for* respeitem a tipagem, avaliam-se os valores introduzidos, verificando se o primeiro é menor que o segundo. Deste modo define-se se o ciclo é crescente ou decrescente e procede-se à escrita do código correspondente. Um exemplo do uso de *for*

$$for(x \leftarrow 2 : 0 : -1)\{t\},$$

onde  $x \leftarrow 2$  se refere à atribuição, o que exige a declaração prévia da variável (não ilustrado no exemplo), 0 corresponde ao valor término do ciclo e -1 ao valor usado para atualizar  $x$ ; este exemplo é um *for* do tipo decrescente.

Outro ciclo disponibilizado é o *while*, que necessita de uma expressão e do seu corpo (instruções), exigindo que a expressão tenha o tipo `bool`; estas instruções são executadas enquanto a expressão for verdade.

- **Funções:** As funções utilizadas têm um modelo análogo às "fun" de OCaml, o que permite que variáveis possam ser do tipo "fun". A utilização de funções é feita por recurso a uma "aplicação" que contém dois parâmetros, onde o primeiro terá de ser a função e o segundo a expressão que lhe serve de argumento. O encadeamento sucessivo de funções não é admissível sendo apenas aplicável a `int`.
- **Apresentação de valores:** O projeto disponibiliza a apresentação de valores do tipo `int`, `float` e `string`, sendo usado para cada um uma instrução dedicada. A apresentação de valores exige que o tipo da expressão a apresentar seja concordante com a instrução de apresentação; não é possível o "outputInt" apresentar um valor do tipo `float`.
- **Leitura de valores:** De forma análoga à apresentação de valores, cada tipo terá a expressão respetiva. Um exemplo do seu uso é

$$let\ x = inputInt,$$

onde `inputInt` irá ler um valor e  $x$  terá o tipo `int`.

- **Inicialização e manipulação de vetores:** É disponibilizada a inicialização de vetores que armazenam valores do tipo `int`. Esta instrução necessita de receber uma expressão, correspondente ao número de elementos do vetor; é necessário que a expressão seja do tipo `int`. De seguida é alocado o espaço necessário para armazenar todos os elementos, respeitando o tipo de array declarado.

É possível atribuir um valor a uma posição específica do vetor, sendo para tal necessário que o identificador seja do tipo `array`, que o valor da posição seja do tipo `int` e que o valor a introduzir seja do mesmo tipo que os elementos do vetor.

É permitida a obtenção de um valor de uma posição específica do vetor. Esta funcionalidade necessita que identificador seja do tipo `array` e que o valor da posição seja do tipo `int`.



# Manual de utilizador

A utilização da nossa aplicação é acessível utilizando o comando "make" que tratará de invocar o Makefile, compilando os ficheiros necessários e executando o "test.exp", utilizando "Mars.jar". Caso o utilizador queira realizar diferentes testes apenas terá de alterar o ficheiro "test.exp", executar "make" e obter o código correspondente no ficheiro "test.s". Seguem-se exemplos de aplicação de funcionalidades:

- **Declaração de variáveis:**

$$\textit{let } x = 2 + 3,$$

onde x é um identificador e 2+3 um exemplo de expressão.

- **Atribuição de valor a variáveis:**

$$x \leftarrow 2 + 3,$$

onde x é um identificador, com declaração prévia, e 2+3 um exemplo de expressão.

- **Operação com expressões:**

$$\textit{let } x = 2; \textit{let } y = 2 + x,$$

onde x é um identificador e y um identificador, com atribuição de valor a partir de operador aritmético + entre o identificador x e valor 2, ambas do tipo int.

- **Execução condicional:**

$$\textit{if}(2 > 3) \textit{ then outputInt } 2 \textit{ else outputFloat } 3.0,$$

onde 2 > 3 representa uma expressão de valor bool e outputInt a instrução de escrita de int e outputFloat a instrução de escrita de float. Visto 2 > 3 ter valor false a instrução correspondente ao else será a executada.

- **Sequência:**

$$\textit{let } x = 2; \textit{let } y = 2 + x,$$

onde todas as considerações tomadas em "Operação com expressões" também se aplica aqui. A separação de instruções é feita por recurso a ";", o separador de instruções usado neste projeto.

- **Ciclos:**

Um exemplo de *for*:

$$\text{let } x = 3; \text{ for}(x \leftarrow 0 : 2 : 1) \{ \text{outputInt } x \}$$

onde  $x \leftarrow 0$  representa a atribuição de 0 a  $x$ , 2 a expressão terminadora de ciclo e 1 o valor a adicionar a  $x$ , a cada iteração. A declaração de  $x$ , exibida acima, representa a obrigatoriedade de declaração prévia antes da atribuição. Um exemplo de *while*:

$$\text{let } x = 0; \text{ while}(x > 3) \{ \text{outputInt } x; \ x ++; \}$$

onde  $x > 3$  representa a expressão bool delimitadora do número de iterações a ocorrer e `outputInt x` e `x++` as instruções a executar a cada iteração.

- **Funções:**

$$\text{let } z = \text{fun } t \rightarrow t + 2; \text{ let } y = z \ 2,$$

onde  $z$  foi declarado como sendo do tipo `fun` usando a expressão `fun t → t + 2`. A variável  $y$  terá o valor 4 por declaração usando uma aplicação (`z 2`). A aplicação terá uma abordagem similar à de OCaml, onde a variável  $z$ , uma função, recebe como parâmetro 2 e executa a operação `2 + 2`.

- **Apresentação e leitura de valores:**

$$\text{let } x = \text{inputInt}; \text{ outputInt } x,$$

onde  $x$  será uma variável de tipo `int`, visto ter-lhe sido atribuído um valor proveniente de `inputInt`, responsável por receber um valor do utilizador. A exibição do valor de  $x$  é feito por recurso a `outputInt`, responsável por exibir valores de tipo `int`.

- **Inicialização e manipulação de vetores:**

$$\text{let } x = \text{arrayInt } 3; \ x[2] \leftarrow 2; \text{ outputInt } x[2],$$

onde  $x$  será uma variável de tipo `array` de tamanho 3. O elemento da posição 2 de  $x$  será alterado, utilizando uma nomenclatura idêntica à atribuição. A última instrução exibida representa o acesso a um elemento do array, exemplificado no `outputInt`.