MAC-5754
CML – Sintaxe e Semântica

Caio Lopes, Marcelo Silvarolla, Leonardo Blanger

IME-USP

23 de maio de 2018

Sintaxe

Sintaxe - Expressões

```
/* Expressões necessariamente "atômicas": */

primary_expression
        : IDENTIFIER
        | literal
        | '{' '}'
        | '{' expression_list '}'
        | '(' expression ')'
        | array_access
        | IDENTIFIER '(' ')'
        | IDENTIFIER '(' expression_list ')'
        ;

literal
        : INT_LITERAL
        | REAL_LITERAL
        | BOOL_LITERAL
        | CHAR_LITERAL
        | STRING_LITERAL
        ;
```

Sintaxe - Expressões

```
/* Expressões "não-atômicas" */
expression_list
        : expression
        | expression_list ',' expression
        ;

expression
        : logical_or_expression
        | IDENTIFIER '=' expression
        | array_access '=' expression
        ;


array_access
        : IDENTIFIER '[' expression ']'
        | array_access '[' expression ']'
        ;
```

Sintaxe - Expressões

```
logical_or_expression
        : logical_and_expression
        | logical_or_expression OR_OP logical_and_expression
        ;

logical_and_expression
        : relational_expression
        | logical_and_expression AND_OP relational_expression
        ;

relational_expression
        : additive_expression
        | additive_expression '<' additive_expression
        | additive_expression '>' additive_expression
        | additive_expression LE_OP additive_expression
        | additive_expression GE_OP additive_expression
        | additive_expression EQ_OP additive_expression
        | additive_expression NE_OP additive_expression
        ;
```

Sintaxe - Expressões

```
additive_expression
        : multiplicative_expression
        | additive_expression '+' multiplicative_expression
        | additive_expression '-' multiplicative_expression
        ;

multiplicative_expression
        : unary_minus_expression
        | multiplicative_expression '*' unary_minus_expression
        | multiplicative_expression '/' unary_minus_expression
        ;

unary_minus_expression
        : neg_expression
        | '-' neg_expression
        ;

neg_expression
        : primary_expression
        | '!' neg_expression
        ;
```

Sintaxe - Comandos

```
command
        : compound_command
        | expression_command
        | selection_command
        | iteration_command
        | jump_command
        | SKIP ';'
        ;

compound_command
        : '{' declaration_or_command_list '}'
        ;

declaration_or_command_list
        : declaration_or_command
        | declaration_or_command_list declaration_or_command
        ;

declaration_or_command
        : declaration
        | command
        ;
```

Sintaxe - Comandos

```
expression_command
        : ';'
        | expression ';'
        ;

selection_command
        : IF '(' expression ')' compound_command ELSE compound_command
        | IF '(' expression ')' compound_command
        ;

iteration_command
        : WHILE '(' expression ')' command
        ;

jump_command
        : RETURN ';'
        | RETURN expression ';'
        ;
```

Sintaxe - Declarações

```
declaration
        : type_specifier IDENTIFIER ';'
        | type_specifier IDENTIFIER '=' expression ';'
        ;

parameter_declaration_list
        : parameter_declaration
        | parameter_declaration_list ',' parameter_declaration
        ;

parameter_declaration
        : type_specifier IDENTIFIER
        ;
```

Sintaxe - Declarações

```
type_specifier
        : VOID
        | CHAR
        | INT
        | REAL
        | BOOL
        | STRING
        | DATASET
        | MODEL
        | type_specifier '[' ']'
        ;
```

Sintaxe - Definições

```
function_definition
        : type_specifier IDENTIFIER '(' ')' compound_command
        | type_specifier IDENTIFIER '('
        | parameter_declaration_list ')' compound_command
        ;
```

Sintaxe - Programa

```
program
        : declaration_or_function_definition
        | program declaration_or_function_definition
        ;

declaration_or_function_definition
        : function_definition
        | declaration
        ;
```

Semântica

Domínios Semânticos

- $\mathit{Int} := \{\ldots, -2, -1, 0, 1, 2, \ldots\} = \mathbb{Z}$
- $\mathit{Real} := \mathbb{R}$
- $\mathit{Bool} := \{\mathit{true}, \mathit{false}\}$
- $\mathit{Char} := \{0, \ldots, 127\}$, onde os números de 0 a 127 são interpretados conforme o padrão ASCII, e.g., 43 representa '$+$', 49 representa '1', etc. Para mais detalhes, ver tabela no início do relatório.

Domínios Semânticos

Definimos, para cada conjunto X, o conjunto $X[]$ dos vetores de elementos de X pondo $X[] := \bigcup_{n=0}^{+\infty} X^n$
$rotulo(X)$ denota o conjunto $\{rotulo(x) : x \in X\}$ de elementos de X rotulados pela palavra *rotulo*. Por exemplo,

$$int(Int) = \{\ldots, int(-2), int(-1), int(0), int(1), int(2), \ldots\}.$$

Domínios Semânticos

Isto serve para que possamos tomar a união, por exemplo, $int(Int) \cup char(Char)$, que é

$$\{\ldots, int(-2), int(-1), int(0), int(1), int(2), \ldots,$$

$$char(0), char(1), \ldots, char(127)\},$$

e saber de que conjunto cada elemento provém. Se simplesmente tomássemos a união sem rótulos, $Int \cup Char = Int$, não saberíamos, por exemplo, se $5 \in Int \cup Char$ provém do conjunto Int ou do conjunto $Char$, isto é, não saberíamos o tipo do valor 5.

Domínios Semânticos

- $String = Char[]$
  Observe que *String* é simplesmente o conjunto dos vetores de *Char*'s. Usaremos os rótulos *string* e *array* para distingui-los.
- $Dataset = \{0, \ldots, n\} \times \{1, \ldots, m\} \rightarrow int(Int) \cup real(Real) \cup string(String)$
- $Model = Dataset \rightarrow Dataset$
- $Array = Location[]$
- $Input := \bigcup_{n=0}^{+\infty} Dataset^n$
- $Output := \bigcup_{n=0}^{+\infty} Dataset^n$

Domínios Semânticos

- $Function = \bigcup_{n=0}^{+\infty}((Location^n \times Store \times Input \times Output) \times Location \rightarrow Store \times Input \times Output)$
- $StorableValue := int(Int) \cup real(Real) \cup bool(Bool) \cup char(Char) \cup string(String) \cup dataset(Dataset) \cup model(Model) \cup array(Array)$
- $ExpressibleValue := StorableValue$
- $DenotableValue := var(Location) \cup fun(Function)$
- $Location := \mathbb{N} := \{0, 1, 2, \ldots\}$

Domínios Semânticos

- $\mathit{Environment} := \texttt{Identifier} \rightarrow \mathit{DenotableValue} \cup \{\mathit{unbound}\}$[1]
- $\mathit{ReturnFlag} := \{\texttt{returnFlag0}, \texttt{returnFlag1}\}$
  A *flag* indica se houve `return` no comando.
- $\mathit{Store} := \mathit{Location} \rightarrow \mathit{StorableValue} \cup \{\mathit{unused}\} \cup \{\mathit{undefined}\}$[2]
- $\Sigma := \mathit{State} := \mathit{Environment} \times \mathit{Store} \times \mathit{Input} \times \mathit{Output}$

[1] O valor *unbound* indica que o identificador não foi associado a uma posição de memória ou função, isto é, não foi declarado.

[2] O valor *unused* indica que a localização não está sendo utilizada por nenhuma variável. Já o valor *undefined* indica que a localização está sendo utilizada por uma variável que não foi inicializada.

Domínios Semânticos

Obs.: acrescentamos o valor especial *error* em todos os domínios, para indicar erro no programa e supomos que erros são propagados pelas funções semânticas.

Frunções Semânticas

$$\begin{aligned}
E &: \texttt{Expression} \rightarrow (\Sigma \rightarrow \mathit{Store} \times \mathit{Input} \times \mathit{Output} \times \mathit{ExpressibleValue}) \\
C &: \texttt{Command} \rightarrow (\Sigma \rightarrow \mathit{Store} \times \mathit{Input} \times \mathit{Output} \times \mathit{ReturnFlag}) \\
\mathit{Dec} &: \texttt{Declaration} \rightarrow (\Sigma \rightarrow \Sigma) \\
\mathit{Def} &: \texttt{FunctionDefinition} \rightarrow (\Sigma \rightarrow \mathit{Environment}) \\
P &: \texttt{Program} \rightarrow (\mathit{Input} \rightarrow \mathit{Output} \times \mathit{Int}) \\
P_1 &: \texttt{Program} \rightarrow (\Sigma \rightarrow \Sigma) \\
P_2 &: \texttt{Program} \rightarrow (\Sigma \rightarrow \Sigma) \\
P_3 &: \texttt{Program} \rightarrow (\Sigma \rightarrow \Sigma)
\end{aligned}$$

Funções Semânticas Auxiliares

$$
\begin{aligned}
&\mathit{value} : \mathit{Literal} \to \mathit{int}(\mathit{Int}) \cup \mathit{real}(\mathit{Real}) \cup \mathit{bool}(\mathit{Bool}) \cup \mathit{char}(\mathit{Char}) \cup \mathit{string}(\mathit{String})\\
&\mathit{emptyEnv} : \mathit{Environment}\\
&\mathit{extendEnv} : \mathit{Environment} \times \mathit{Identifier} \times \mathit{DenotableValue} \to \mathit{Environment}\\
&\mathit{applyEnv} : \mathit{Environment} \times \mathit{Identifier} \to \mathit{DenotableValue} \cup \{\mathit{unbound}\}\\
&\mathit{emptySto} : \mathit{Store}\\
&\mathit{updateSto} : \mathit{Store} \times \mathit{Location} \times \big(\mathit{StorableValue} \cup \{\mathit{undefined}, \mathit{unused}\}\big) \to \mathit{Store}\\
&\mathit{applySto} : \mathit{Store} \times \mathit{Location} \to \mathit{StorableValue} \cup \{\mathit{undefined}, \mathit{unused}\}\\
&\mathit{allocate} : \mathit{Store} \to \mathit{Store} \times \mathit{Location}\\
&\mathit{deallocate} : \mathit{Store} \times \mathit{Location} \to \mathit{Store}
\end{aligned}
$$

Equações Semânticas - Expressões

$$E\ [\![\texttt{id}]\!]\ (env, sto, in, out) = \ \textbf{if}\ val = undefined\ \textbf{then}\ error\ \textbf{else}\ (sto, in, out, val)$$

$$\qquad \textbf{where}\ val = applySto(sto, loc)$$

$$\qquad \textbf{where}\ loc = applyEnv(env, \texttt{id})$$

$$E[\![\texttt{lit}]\!](env, sto, in, out) = (sto, in, out, value(\texttt{lit}))$$

Equações Semânticas - Expressões

$$
\begin{aligned}
&E[[\{\}]]\ (env, sto, in, out) = (sto, in, out, array(\emptyset))\\
&E[[\{\texttt{exp}\}]]\ (env, sto, in, out) = (sto_3, in_1, out_1, array(loc))\\
&\qquad \textbf{where}\ (sto_1, in_1, out_1, val) = E[[\texttt{exp}]](env, sto, in, out)\\
&\qquad \textbf{and}\ (sto_2, loc) = allocate\ sto_1\\
&\qquad \textbf{and}\ sto_3 = updateSto(sto_2, loc, val)\\
&E[[\{\texttt{exp\_list}, \texttt{exp}\}]]\ (env, sto, in, out) = (sto_f, in_f, out_f, array(extendArray(arr, loc)))\\
&\qquad \textbf{where}\ (sto_1, in_1, out_1, array(arr)) = E[[\{\texttt{exp\_list}\}]]\ (env, sto, in, out)\\
&\qquad \textbf{and}\ (sto_2, in_f, out_f, val) = E[[\texttt{exp}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and}\ (sto_3, loc) = allocate\ sto_2\\
&\qquad \textbf{where}\ sto_f = updateSto(sto_3, loc, val)
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{aligned}
&E[[\texttt{exp}_1 + \texttt{exp}_2]](env, sto, in, out) = (sto_f, in_f, out_f, val_f)\\
&\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = int(val_1 + val_2)\\
&\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = real(val_1 + val_2)\\
&\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = real(val_1 + val_2)\\
&\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = real(val_1 + val_2)
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{aligned}
& E[[\texttt{exp}_1 - \texttt{exp}_2]](env, sto, in, out) = (sto_f, in_f, out_f, val_f) \\
& \quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
& \qquad \textbf{and } val_f = int(val_1 - val_2) \\
& \quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
& \qquad \textbf{and } val_f = real(val_1 - val_2) \\
& \quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
& \qquad \textbf{and } val_f = real(val_1 - val_2) \\
& \quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
& \qquad \textbf{and } val_f = real(val_1 - val_2)
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{array}{l}
E[[\mathtt{exp}_1 * \mathtt{exp}_2]](env, sto, in, out) = (sto_f, in_f, out_f, val_f) \\
\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\mathtt{exp\_1}]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\mathtt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
\qquad \textbf{and } val_f = int(val_1 * val_2) \\
\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\mathtt{exp\_1}]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\mathtt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
\qquad \textbf{and } val_f = real(val_1 * val_2) \\
\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\mathtt{exp\_1}]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\mathtt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
\qquad \textbf{and } val_f = real(val_1 * val_2) \\
\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\mathtt{exp\_1}]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\mathtt{exp\_2}]]\ (env, sto_1, in_1, out_1) \\
\qquad \textbf{and } val_f = real(val_1 * val_2)
\end{array}
$$

Equações Semânticas - Expressões

$$
\begin{aligned}
&E[[\mathtt{exp}_1/\mathtt{exp}_2]](env, sto, in, out) = ans\\
&\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and if } val_2 = 0 \textbf{ then } ans = error \textbf{ else } ans = (sto_f, in_f, out_f, int(val_1/val_2))\\
&\quad \textbf{where } (sto_1, in_1, out_1, int(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and if } val_2 = 0 \textbf{ then } ans = error \textbf{ else } ans = (sto_f, in_f, out_f, real(val_1/val_2))\\
&\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, int(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and if } val_2 = 0 \textbf{ then } ans = error \textbf{ else } ans = (sto_f, in_f, out_f, real(val_1/val_2))\\
&\quad \textbf{where } (sto_1, in_1, out_1, real(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, real(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and if } val_2 = 0 \textbf{ then } ans = error \textbf{ else } ans = (sto_f, in_f, out_f, real(val_1/val_2))
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{aligned}
&E[[-\texttt{exp}]](env, sto, in, out) = (sto_f, in_f, out_f, labeledVal)\\
&\quad \textbf{where } (sto_f, in_f, out_f, int(val)) = E[[\texttt{exp}]](env, sto, in, out)\\
&\qquad \textbf{where } labeledVal = int(-val)\\
&\quad \textbf{where } (sto_f, in_f, out_f, real(val)) = E[[\texttt{exp}]](env, sto, in, out)\\
&\qquad \textbf{where } labeledVal = real(-val)
\end{aligned}
$$

Equações Semânticas - Expressões

$$\begin{aligned}
&E[[\texttt{exp}_1\ \texttt{||}\ \texttt{exp}_2]](env, sto, in, out) = (sto_f, in_f, out_f, val_f)\\
&\quad \textbf{where } (sto_1, in_1, out_1, bool(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, bool(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = bool(val_1 \vee val_2)
\end{aligned}$$

$$\begin{aligned}
&E[[\texttt{exp}_1\ \texttt{\&\&}\ \texttt{exp}_2]]\ (env, sto, in, out) = (sto_f, in_f, out_f, val_f)\\
&\quad \textbf{where } (sto_1, in_1, out_1, bool(val_1)) = E[[\texttt{exp\_1}]]\ (env, sto, in, out)\\
&\qquad \textbf{and } (sto_f, in_f, out_f, bool(val_2)) = E[[\texttt{exp\_2}]]\ (env, sto_1, in_1, out_1)\\
&\qquad \textbf{and } val_f = bool(val_1 \wedge val_2)
\end{aligned}$$

Equações Semânticas - Expressões

$$
\begin{aligned}
& E[[\texttt{exp}_1\ \texttt{==}\ \texttt{exp}_2]]\ (env, sto, in, out) = \\
& \quad \textbf{if } labeledVal_1 = labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
& \ \textbf{else } (sto_2, in_2, out_2, bool(false)) \\
& \qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]](env, sto_1, in_1, out_1)
\end{aligned}
$$

$$
\begin{aligned}
& E[[\texttt{exp}_1\ \texttt{!=}\ \texttt{exp}_2]]\ (env, sto, in, out) = \\
& \quad \textbf{if } labeledVal_1 \neq labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
& \ \textbf{else } (sto_2, in_2, out_2, bool(false)) \\
& \qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
& \qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]](env, sto_1, in_1, out_1)
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{aligned}
&E[[\texttt{exp}_1\ \texttt{<}\ \texttt{exp}_2]]\ (env, sto, in, out) = \\
&\quad \textbf{if } labeledVal_1 < labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
&\ \textbf{else } (sto_2, in_2, out_2, bool(false)) \\
&\qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
&\qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]](env, sto_1, in_1, out_1)
\end{aligned}
$$

$$
\begin{aligned}
&E[[\texttt{exp}_1\ \texttt{<=}\ \texttt{exp}_2]]\ (env, sto, in, out) = \\
&\quad \textbf{if } labeledVal_1 \leq labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
&\ \textbf{else } (sto_2, in_2, out_2, bool(false)) \\
&\qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
&\qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]](env, sto_1, in_1, out_1)
\end{aligned}
$$

Equações Semânticas - Expressões

$$
\begin{array}{l}
E[[\texttt{exp}_1 \ \texttt{>} \ \texttt{exp}_2]]\ (env, sto, in, out) = \\
\quad \textbf{if } labeledVal_1 > labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
\textbf{else } (sto_2, in_2, out_2, bool(false)) \\
\qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]]\ (env, sto_1, in_1, out_1)
\end{array}
$$

$$
\begin{array}{l}
E[[\texttt{exp}_1 \ \texttt{>=} \ \texttt{exp}_2]]\ (env, sto, in, out) = \\
\quad \textbf{if } labeledVal_1 \geq labeledVal_2 \textbf{ then } (sto_2, in_2, out_2, bool(true)) \\
\textbf{else } (sto_2, in_2, out_2, bool(false)) \\
\qquad \textbf{where } (sto_1, in_1, out_1, labeledVal_1) = E[[\texttt{exp}_1]]\ (env, sto, in, out) \\
\qquad \textbf{and } (sto_2, in_2, out_2, labeledVal_2) = E[[\texttt{exp}_2]](env, sto_1, in_1, out_1)
\end{array}
$$

Equações Semânticas - Expressões

$$E[[\texttt{!exp}]]\ (env, sto, in, out) = (sto_f, in_f, out_f, bool(\neg val))$$
$$\quad \textbf{where}\ (sto_f, in_f, out_f, bool(val)) = E[[\texttt{exp}]]\ (env, sto, in, out)$$

$$E[[\texttt{(e)}]]\ (env, sto, in, out) = E[[\texttt{e}]]\ (env, sto, in, out)$$

Equações Semânticas - Expressões

$$
\begin{array}{l}
E[\![\texttt{id\_or\_arr\_access}]\!]\ (env, sto, in, out) = (sto_f, in_f, out_f, val_f) \\
\quad \textbf{where}\ (sto_f, in_f, out_f, val_f) = applyArray(\texttt{id\_or\_arr\_access}, sto, in, out) \\
\quad \textbf{and}\ applyArray(\texttt{id}, sto, in, out) = (sto, in, out, applySto(sto, loc)) \\
\qquad \textbf{where}\ loc = applyEnv(env, \texttt{id}) \\
\quad \textbf{and}\ applyArray(\texttt{id\_or\_arr\_access[exp]}, sto, in, out) = (sto_2, in_2, out_2, arr_{pos}) \\
\qquad \textbf{where}\ (sto_1, in_1, out_1, arr) = applyArray(\texttt{id\_or\_arr\_access}, sto, in, out) \\
\qquad \textbf{and}\ (sto_2, in_2, out_2, pos) = E[\![\texttt{exp}]\!]\ (env, sto_1, in_1, out_1)
\end{array}
$$

Equações Semânticas - Expressões

$$E[[\texttt{id\_or\_arr\_access} = \texttt{exp}]]\ (env, sto, in, out) = (sto_2, in_2, out_2, expVal)$$

$$\textbf{where}\quad (sto_1, in_1, out_1, expVal) = E[[\texttt{exp}]]\ (env, sto, in, out)$$

$$\textbf{where}\quad (sto_2, in_2, out_2) = updateArray((env, sto_1, in_1, out_1), \texttt{id\_or\_arr\_access}, expVal)$$

where

$$updateArray : \Sigma \times (\texttt{Identifier} \cup \texttt{ArrayAccess}) \times ExpressibleValue \rightarrow Store \times Input \times Output$$

$$\textbf{where}\quad updateArray((env, sto, in, out), \texttt{id}, val) = (updateSto(sto, loc, val), in, out)$$

$$\textbf{where}\quad loc = applyEnv(env, \texttt{id})$$

$$\textbf{and}\ updateArray((env, sto, in, out), \texttt{id\_or\_arr\_access}[\texttt{exp}], val) = (sto_f, in_f, out_f)$$

$$\textbf{where}\ (sto_1, in_1, out_1, array(arr)) = E[[\texttt{id\_or\_arr\_access}]]\ (env, sto, in, out)$$

$$\textbf{where}\ (sto_2, in_f, out_f, int(pos)) = E\ [[\texttt{exp}]]\ (env, sto_1, in_1, out_1)$$

$$\textbf{where}\ sto_f = updateSto(sto_2, arr_{pos}, val)$$

Equações Semânticas - Expressões

$$
\begin{array}{l}
E[[\texttt{id()}]]\ (env, sto, in, out) = (sto_f, in_f, out_f, val_f) \\
\quad \textbf{where}\ \ fun(f) = applyEnv(env, \texttt{id}) \\
\quad \textbf{and}\ \ (sto_f, in_f, out_f) = f((\emptyset, sto', in, out), loc) \\
\qquad \textbf{where}\ \ (sto', loc) = allocate\ sto \\
\quad \textbf{and}\ \ val_f = applySto(sto_f, loc)
\end{array}
$$

Equações Semânticas - Expressões

$E[[\texttt{id(exp\_list)}]]\ (env, sto, in, out) = (sto_f, in_f, out_f, val_f)$

$\quad$ **where** $fun(f) = applyEnv(env, \texttt{id})$

$\quad$ **and** $(sto_f, in_f, out_f) = f(locations(\texttt{exp\_list}, sto', in, out), loc)$

$\qquad$ **where** $(sto', loc) = allocate\ sto$

$\qquad$ **and**

$locations : ExpressionList \times Store \times Input \times Output \rightarrow Location[] \times Store \times Input \times Output$

$\qquad locations(\texttt{exp}, sto, in, out) = ((loc), sto_3, in_1, out_1)$

$\qquad\quad$ **where** $(sto_1, in_1, out_1, val) = E[[\texttt{exp}]]\ (env, sto, in, out)$

$\qquad\quad$ **and** $(sto_2, loc) = allocate\ sto$

$\qquad\quad$ **and** $sto_3 = updateSto(sto_2, loc, val)$

$\qquad$ **and** $locations(\texttt{exp\_list ","exp}, sto, in, out) = (concat(init, last), sto_2, in_2, out_2)$

$\qquad\quad$ **where** $(init, sto_1, in_1, out_1) = locations(\texttt{exp\_list}, sto, in, out)$

$\qquad\quad$ **and** $(last, sto_2, in_2, out_2) = locations(\texttt{exp}, sto_1, in_1, out_1)$

Equações Semânticas - Comandos

$$
\begin{array}{l}
C[[\{\texttt{dec\_cmd\_list}\}]]\ (env, sto, in, out) = (sto_f, in_f, out_f, retFlag) \\
\quad \textbf{where}\ (env_f, sto_f, in_f, out_f, retFlag) = sequence(\texttt{dec\_cmd\_list})\ (env, sto, in, out) \\
\quad \textbf{and}\ sequence(\texttt{dec})\ (env, sto, in, out) = (Dec[[\texttt{dec}]]\ (env, sto, in, out), returnFlag0) \\
\quad \textbf{and}\ sequence(\texttt{cmd})\ (env, sto, in, out) = (env, C[[\texttt{cmd}]]\ (env, sto, in, out)) \\
\quad \textbf{and}\ sequence(\texttt{dec\_cmd\_list dec\_cmd})\ (env, sto, in, out) = \textbf{if}\ retFlagRec = returnFlag1 \\
\qquad \textbf{then}\ (env_1, sto_1, in_1, out_1, retFlagRec) \\
\qquad \textbf{else}\ (env_2, sto_2, in_2, out_2, retFlagRec') \\
\qquad\ \textbf{where}\ (env_1, sto_1, in_1, out_1, retFlagRec) = sequence(\texttt{dec\_cmd\_list})\ (env, sto, in, out) \\
\qquad\ \textbf{and}\ (env_2, sto_2, in_2, out_2, retFlagRec') = sequence(\texttt{dec\_cmd})\ (env_1, sto_1, in_1, out_1)
\end{array}
$$

Equações Semânticas - Comandos

$$C[[\texttt{exp;}]]\ (env, sto, in, out) = (env, sto_f, in_f, out_f, returnFlag0)$$
$$\textbf{where}\ (sto_f, in_f, out_f, val) = E[[\texttt{exp}]]\ (env, sto, in, out)$$

Equações Semânticas - Comandos

$$
\begin{aligned}
&C[[\texttt{IF(exp) comp\_cmd}]]\ (env, sto, in, out) = \textbf{if } b\\
&\quad \textbf{then } C[[\texttt{comp\_cmd}]]\ (env, sto_e, in_e, out_e)\\
&\quad \textbf{else } (sto_e, in_e, out_e, returnFlag0)\\
&\qquad \textbf{where } (sto_e, in_e, out_e, bool(b)) = E[[\texttt{exp}]]\ (env, sto, in, out)
\end{aligned}
$$

$$
\begin{aligned}
&C[[\texttt{IF(exp) comp\_cmd}_1\ \texttt{ELSE comp\_cmd}_2]]\ (env, sto, in, out) = \textbf{if } b\\
&\quad \textbf{then } C[[\texttt{comp\_cmd}_1]]\ (env, sto_e, in_e, out_e)\\
&\quad \textbf{else } C[[\texttt{comp\_cmd}_2]]\ (env, sto_e, in_e, out_e)\\
&\qquad \textbf{where } (sto_e, in_e, out_e, bool(b)) = E[[\texttt{exp}]]\ (env, sto, in, out)
\end{aligned}
$$

Equações Semânticas - Comandos

$$
\begin{array}{l}
C[[\texttt{while (exp) cmd}]]\ (env, sto, in, out) = loop(sto, in, out, returnFlag0) \\
\quad \textbf{where } loop(sto, in, out, retFlag) = \textbf{ if } retFlag = returnFlag1 \\
\qquad \textbf{then } (sto, in, out, returnFlag1) \\
\qquad \textbf{else if } b \\
\qquad\quad \textbf{then } loop(C[[\texttt{cmd}]]\ (env, sto_e, in_e, out_e)) \\
\qquad\quad \textbf{else } (sto_e, in_e, out_e, returnFlag0) \\
\qquad \textbf{where } (sto_e, in_e, out_e, bool(b)) = E[[\texttt{exp}]]\ (env, sto, in, out)
\end{array}
$$

Equações Semânticas - Comandos

$$C[[\texttt{RETURN;}]]\ (env, sto, in, out) = \ (sto, in, out, returnFlag1)$$

$$\begin{aligned}
&C[[\texttt{RETURN exp;}]]\ (env, sto, in, out) = \ (sto', in', out', returnFlag1)\\
&\quad \textbf{where}\ (sto_e, in', out', val) = E[[\texttt{exp}]]\ (env, sto, in, out)\\
&\quad \textbf{and}\ sto' = updateSto(sto_e, applyEnv(env, \texttt{return}), val)
\end{aligned}$$

$$C[[\texttt{SKIP;}]]\ (env, sto, in, out) = (sto, in, out, returnFlag0)$$

Equações Semânticas - Declarações

$$
\begin{aligned}
&\mathit{Dec}[[\texttt{type\_spec id;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = (\mathit{env}_f, \mathit{sto}_f, \mathit{in}, \mathit{out})\\
&\quad \textbf{where } \mathit{env}_f = \mathit{extendEnv}(\mathit{env}, \texttt{id}, \mathit{var}(\mathit{loc}))\\
&\quad \textbf{and } (\mathit{sto}_f, \mathit{loc}) = \mathit{allocate}\ \mathit{sto}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Dec}[[\texttt{type\_spec id = exp;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = (\mathit{env}_f, \mathit{sto}_f, \mathit{in}_f, \mathit{out}_f)\\
&\quad \textbf{where } (\mathit{env}_f, \mathit{sto}_1, \mathit{in}_1, \mathit{out}_1) = \mathit{Dec}[[\texttt{type\_spec id;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out})\\
&\quad \textbf{and } (\mathit{sto}_f, \mathit{in}_f, \mathit{out}_f, \mathit{val}) = E[[\texttt{id = exp}]](\mathit{env}_1, \mathit{sto}_1, \mathit{in}_1, \mathit{out}_1)
\end{aligned}
$$

Equações Semânticas - Definições

$$
\begin{aligned}
& \mathit{Def}[[\texttt{type\_spec id() comp\_cmd}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = \mathit{env}_1 \\
& \quad \textbf{where } \mathit{env}_1 = \mathit{extendEnv}(\mathit{env}, \texttt{id}, \mathit{fun}(f)) \\
& \quad \textbf{and } f(\emptyset, \mathit{sto}, \mathit{in}, \mathit{out}) = C[[\texttt{comp\_cmd}]]\ (\mathit{env}_1, \mathit{sto}, \mathit{in}, \mathit{out})
\end{aligned}
$$

Equações Semânticas - Definições

$$
\begin{array}{l}
\mathit{Def}[[\texttt{type\_spec id(param\_dec\_list) comp\_cmd}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = \mathit{env}_1 \\
\quad \textbf{where } \mathit{env}_1 = \mathit{extendEnv}(\mathit{env}, \texttt{id}, \mathit{fun}(f)) \\
\quad \textbf{and } f(\mathit{array}, \mathit{sto}, \mathit{in}, \mathit{out}) = C[[\texttt{comp\_cmd}]]\ (\mathit{env}', \mathit{sto}, \mathit{in}, \mathit{out}) \\
\quad\quad \textbf{and } \mathit{env}' = \mathit{modifyEnv}(\mathit{env}1, \texttt{param\_dec\_list}, \mathit{array}) \\
\quad\quad \textbf{where } \mathit{modifyEnv}(\mathit{env}, \texttt{type\_spec id}, \mathit{array}) = \mathit{extendEnv}(\mathit{env}, \texttt{id}, \mathit{array}_0) \\
\quad\quad \textbf{and } \mathit{modifyEnv}(\mathit{env}, \texttt{param\_dec\_list, type\_spec id}, \mathit{array}) = \mathit{extendEnv}(\mathit{env}_2, \texttt{id}, \mathit{last}) \\
\quad\quad\quad \textbf{where } \mathit{env}_2 = \mathit{modifyEnv}(\mathit{env}, \texttt{param\_dec\_list}, \mathit{init}) \\
\quad\quad\quad\quad \textbf{where } \mathit{init} = \mathit{init}(\mathit{array}) \\
\quad\quad\quad\quad \textbf{and } \mathit{last} = \mathit{last}(\mathit{array})
\end{array}
$$

Equações Semânticas - Programa

$$
\begin{aligned}
&P_1[[\texttt{type\_spec id;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = \mathit{Dec}[[\texttt{type\_spec id;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out})\\
&P_1[[\texttt{type\_spec id = exp;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = \mathit{Dec}[[\texttt{type\_spec id;}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out})\\
&P_1[[\texttt{fun\_def}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = (\mathit{env}_1, \mathit{sto}, \mathit{in}, \mathit{out})\\
&\quad \textbf{where } \mathit{env}_1 = \mathit{Def}[[\texttt{fun\_def}]]
\end{aligned}
$$

$$P_1[[\texttt{prog dec\_or\_fun\_def}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}) = P_1[[\texttt{dec\_or\_fun\_def}]]\ (P_1[[\texttt{prog}]]\ (\mathit{env}, \mathit{sto}, \mathit{in}, \mathit{out}))$$

$$P_2 = P_1$$

Equações Semânticas - Programa

$$
\begin{aligned}
P_3[[\texttt{dec}]] &= Dec[[\texttt{dec}]] \\
P_3[[\texttt{fun\_def}]]\ (env, sto, in, out) &= (env, sto, in, out) \\
P_3[[\texttt{prog dec\_or\_fun\_def}]]\ (env, sto, in, out) &= P_3[[\texttt{dec\_or\_fun\_def}]]\ (P_3[[\texttt{prog}]]\ (env, sto, in, out))
\end{aligned}
$$

Equações Semânticas - Programa

$$
\begin{aligned}
&P[[\texttt{prog}]](env, sto, in, out) = (sto_f, in_f, out_f)\\
&\quad \textbf{where } (env_1, sto_1, in_1, out_1) = (P_3[[\texttt{prog}]] \circ P_2[[\texttt{prog}]] \circ P_1[[\texttt{prog}]])\ (env, sto, in, out)\\
&\quad \textbf{and } (sto_f, in_f, out_f, retVal) = E[[\textbf{main()}]]\ (env_f, sto_1, in_1, out_1)
\end{aligned}
$$