The PREV'19 programming language

(academic year 2018/19)

Boštjan Slivnik

1 Lexical structure

Programs in the PREV'19 programming language are written in ASCII character set (no additional characters denoting post-alveolar consonants are allowed).

Programs in the PREV'19 programming language consist of the following lexical elements:

- *Literals*:
  - literals of type void: `none`
  - literals of type bool: `true` `false`
  - literals of type char:
    An character with a character code in decimal range $\{32 \ldots 126\}$ (from space to `~`) enclosed in single quotes (`'`).
  - literals of type int:
    A nonempty finite string of digits (`0`...`9`) optionally preceded by a sign (`+` or `-`).
  - literals of pointer types: `null`
  - string literals:
    A possibly empty finite string of characters with character codes in decimal range $\{32 \ldots 126\} \setminus \{34\}$ (from space to `~` but excluding double quote) enclosed in double quotes (`"`).
- *Symbols*:
  `! | ^ & == != <= >= < > + - * / % $ @ = . , : ; [ ] ( ) { }`
- *Keywords*:
  `arr bool char del do else end fun if int new ptr rec then typ var void where while`
- *Identifiers*:
  A nonempty finite string of letters (`A`...`Z` and `a`...`z`), digits (`0`...`9`), and underscores (`_`) that (a) starts with either a letter or an underscore and (b) is not a keyword or a literal.
- *Comments*:
  A string of characters starting with a hash (`#`) and extending to the end of line.
- *White space*:
  Space, horizontal tab (HT), line feed (LF) and carriage return (CR). Line feed alone denotes the end of line within a source file. Horizontal tab is 8 spaces wide.

Lexical elements should be recognised from left to right using the longest match approach.

2 Syntax structure

The concrete syntax of the PREV programming language is defined by context free grammar with the start symbol *source* and productions

(*program*)	*source*	⟶ *decl* {*decl*}
(*type declaration*)	*decl*	⟶ `typ` *identifier* `:` *type* `;`
(*variable declaration*)	*decl*	⟶ `var` *identifier* `:` *type* `;`
(*function declaration*)	*decl*	⟶ `fun` *identifier* `(`[*identifier* `:` *type* {`,` *identifer* `:` *type*}]`)` `:` *type* [`=`*expr*]`;`
(*atomic type*)	*type*	⟶ `void` $\mid$ `bool` $\mid$ `char` $\mid$ `int`
(*array type*)	*type*	⟶ `arr` `[`*expr*`]` *type*
(*record type*)	*type*	⟶ `rec` `(`*identifier* `:` *type* {`,` *identifier* `:` *type*}`)`
(*pointer type*)	*type*	⟶ `ptr` *type*
(*named type*)	*type*	⟶ *identifier*
(*enclosed type*)	*type*	⟶ `(`*type*`)`
(*literal*)	*expr*	⟶ *literal*
(*unary expression*)	*expr*	⟶ *unop expr*
(*binary expression*)	*expr*	⟶ *expr binop expr*
(*variable access*)	*expr*	⟶ *identifier*
(*function call*)	*expr*	⟶ *identifier* `(`[*expr* {`,` *expr*}]`)`
(*element access*)	*expr*	⟶ *expr* `[`*expr*`]`
(*component access*)	*expr*	⟶ *expr* `.` *identifier*
(*memory allocation*)	*expr*	⟶ `new` `(`*type*`)`
(*memory deallocation*)	*expr*	⟶ `del` `(`*expr*`)`
(*compound expression*)	*expr*	⟶ `{` *stmt* {*stmt*} `:` *expr* [`where` *decl* {*decl*}]`}`
(*typecast*)	*expr*	⟶ `(`*expr* `:` *type*`)`
(*enclosed expression*)	*expr*	⟶ `(`*expr*`)`
(*expression*)	*stmt*	⟶ *expr* `;`
(*assignment*)	*stmt*	⟶ *expr* `=` *expr* `;`
(*conditional*)	*stmt*	⟶ `if` *expr* `then` *stmt* {*stmt*} [`else` *stmt* {*stmt*}] `end;`
(*loop*)	*stmt*	⟶ `while` *expr* `do` *stmt* {*stmt*} `end;`

where *literal* denotes any literal, *unop* denotes an unary operator (any of `!`, `+`, `-`, `$` and `@`) and *binop* denotes a binary operator (any of `|`, `^`, `&`, `==`, `!=`, `<=`, `>=`, `<`, `>`, `+`, `-`, `*`, `/` and `%`). In the grammar above, braces typeset as {} enclose sentential forms that can repeated zero or more times, brackets typeset as [] enclose sentential forms that can be present or not while braces and brackets typeset as `{}` and `[]` denote characters that are a part of the program text.

Relational operators are non-associative, all other binary operators are left associative.

The precedence of operators is as follows:

<table>
<tr><td><code>| ^</code></td><td>THE LOWEST PRECEDENCE</td></tr>
<tr><td><code>&</code></td><td></td></tr>
<tr><td><code>== != <= >= < ></code></td><td></td></tr>
<tr><td><code>+ -</code></td><td>(binary <code>+</code> and <code>-</code>)</td></tr>
<tr><td><code>* / %</code></td><td></td></tr>
<tr><td><code>! + - $ @ new del</code> type-cast</td><td>(unary <code>+</code> and <code>-</code>)</td></tr>
<tr><td>element-access component-access</td><td>THE HIGHEST PRECEDENCE</td></tr>
</table>

3 Semantics

3.1 Name binding

Namespaces. There are two kinds of namespaces:

1. Names of types, functions, variables and parameters belong to one single global namespace.
2. Names of record components belong to record-specific namespaces, i.e., each record defines its own namespace containing names of its components.

Scopes. A new scope is created in two ways:

1. A compound expression creates a new scope. The scope starts right after `{` and ends just before `}`.
2. A function declaration creates a new scope. The name of a function, the types of parameters and the type of a result belong to the scope of the function declaration while the names of parameters and the expression denoting the function body (if present) belong to the new inner scope created by the function declaration.

All names declared within a given scope are visible in the entire scope unless hidden by a declaration in the nested scope. A name can be declared within the same scope at most once.

Let $\mathcal{P}$, $\mathcal{I}$, $\mathcal{D}$ and $\mathcal{T}$ be a set of all phrases, a set of all identifiers, a set of all declarations and a set of all types of the PREV'19 programming language, respectively.

The semantic function

$$[\![\cdot]\!]_{\mathrm{ENV}}\colon (\mathcal{P} \cup \mathcal{T}) \to (\mathcal{I} \to \mathcal{D})$$

maps a phrase of PREV'19 to an environment the phrase appears in.

The program itself appears in the empty environment $\mathcal{E}_0\colon \mathcal{I} \to \mathcal{D}$ which is undefined for all identifiers in $\mathcal{I}$, i.e., $\mathcal{E}_0(\mathit{identifier}) = \bot$ for each $\mathit{identifier} \in \mathcal{I}$.

(*source*):

$$\frac{\begin{array}{c}[\![\mathit{decl}_1 \ldots \mathit{decl}_m]\!]_{\mathrm{ENV}} = \mathcal{E}_0 \\ \forall i,j \in \{1 \ldots m\}\colon \mathrm{nm}(\mathit{decl}_i) \neq \mathrm{nm}(\mathit{decl}_j) \\ \mathcal{E}' = \mathcal{E}_0 \triangleright (\mathrm{nm}(\mathit{decl}_1) \mapsto \mathit{decl}_1) \triangleright \ldots \triangleright (\mathrm{nm}(\mathit{decl}_m) \mapsto \mathit{decl}_m)\end{array}}{\forall i \in \{1 \ldots m\}\colon [\![\mathit{decl}_i]\!]_{\mathrm{ENV}} = \mathcal{E}'}$$

(*type and variable declaration*):

$$\frac{[\![\texttt{typ}\ \mathit{identifier}\,\texttt{:}\,\mathit{type}\,\texttt{;}]\!]_{\mathrm{ENV}} = \mathcal{E}}{[\![\mathit{type}]\!]_{\mathrm{ENV}} = \mathcal{E}} \qquad \frac{[\![\texttt{var}\ \mathit{identifier}\,\texttt{:}\,\mathit{type}\,\texttt{;}]\!]_{\mathrm{ENV}} = \mathcal{E}}{[\![\mathit{type}]\!]_{\mathrm{ENV}} = \mathcal{E}}$$

(*function declaration*):

$$\frac{\begin{array}{c}[\![\texttt{fun}\ \mathit{identifer}\,\texttt{(}\mathit{identifer}_1\,\texttt{:}\,\mathit{type}_1\,\texttt{,}\,\ldots\,\texttt{,}\,\mathit{identifer}_n\,\texttt{:}\,\mathit{type}_n\texttt{)}\,\texttt{:}\,\mathit{type}\,\texttt{;}]\!]_{\mathrm{ENV}} = \mathcal{E} \\ \forall i,j \in \{1 \ldots n\}\colon \mathit{identifier}_i \neq \mathit{identifier}_j \\ \mathcal{E}' = \mathcal{E} \triangleright (\mathit{identifier}_1 \mapsto \mathit{identifier}_1\,\texttt{:}\,\mathit{type}_1) \triangleright \ldots \triangleright (\mathit{identifier}_n \mapsto \mathit{identifier}_n\,\texttt{:}\,\mathit{type}_n)\end{array}}{\forall i \in \{1 \ldots n\}\colon [\![\mathit{type}_i]\!]_{\mathrm{ENV}} = \mathcal{E} \quad [\![\mathit{type}]\!]_{\mathrm{ENV}} = \mathcal{E}}$$

$$\frac{\begin{array}{c}[\![\texttt{fun}\ \mathit{identifer}\,\texttt{(}\mathit{identifer}_1\,\texttt{:}\,\mathit{type}_1\,\texttt{,}\,\ldots\,\texttt{,}\,\mathit{identifer}_n\,\texttt{:}\,\mathit{type}_n\texttt{)}\,\texttt{:}\,\mathit{type}\,\texttt{=}\,\mathit{expr}\,\texttt{;}]\!]_{\mathrm{ENV}} = \mathcal{E} \\ \forall i,j \in \{1 \ldots n\}\colon \mathit{identifier}_i \neq \mathit{identifier}_j \\ \mathcal{E}' = \mathcal{E} \triangleright (\mathit{identifier}_1 \mapsto \mathit{identifier}_1\,\texttt{:}\,\mathit{type}_1) \triangleright \ldots \triangleright (\mathit{identifier}_n \mapsto \mathit{identifier}_n\,\texttt{:}\,\mathit{type}_n)\end{array}}{\forall i \in \{1 \ldots n\}\colon [\![\mathit{type}_i]\!]_{\mathrm{ENV}} = \mathcal{E} \quad [\![\mathit{type}]\!]_{\mathrm{ENV}} = \mathcal{E} \quad [\![\mathit{expr}]\!]_{\mathrm{ENV}} = \mathcal{E}'}$$

(*compound statement*):

$$\frac{\begin{array}{c}[\![\texttt{\{}\mathit{stmt}_1 \ldots \mathit{stmt}_n \,\texttt{:}\, \mathit{expr}\ \texttt{where}\ \mathit{decl}_1 \ldots \mathit{decl}_m\texttt{\}}]\!]_{\mathrm{ENV}} = \mathcal{E} \\ \forall i,j \in \{1\ldots m\}\colon \mathrm{nm}(\mathit{decl}_i) \neq \mathrm{nm}(\mathit{decl}_j) \\ \mathcal{E}' = \mathcal{E} \triangleright (\mathrm{nm}(\mathit{decl}_1) \mapsto \mathit{decl}_1) \triangleright \ldots \triangleright (\mathrm{nm}(\mathit{decl}_m) \mapsto \mathit{decl}_m)\end{array}}{\forall i \in \{1\ldots n\}\colon [\![\mathit{stmt}_i]\!]_{\mathrm{ENV}} = \mathcal{E}' \quad \forall i \in \{1\ldots m\}\colon [\![\mathit{decl}_i]\!]_{\mathrm{ENV}} = \mathcal{E}' \quad [\![\mathit{expr}]\!]_{\mathrm{ENV}} = \mathcal{E}'}$$

(*record type*):

$$\frac{[\![\texttt{rec}\ \texttt{(}\mathit{identifier}_1 \,\texttt{:}\, \mathit{type}_1\,\texttt{,}\, \ldots\,\texttt{,}\, \mathit{identifier}_n \,\texttt{:}\, \mathit{type}_n\texttt{)}]\!]_{\mathrm{ENV}} = \mathcal{E}}{\forall i \in \{1\ldots n\}\colon [\![\mathit{type}_i]\!]_{\mathrm{ENV}} = \mathcal{E}}$$

(*component access*):

$$\frac{\begin{array}{c}[\![\mathit{expr}\,\texttt{.}\,\mathit{id}]\!]_{\mathrm{ENV}} = \mathcal{E} \\ [\![\mathit{expr}]\!]_{\mathrm{OFTYPE}} = \tau \quad [\![\tau]\!]_{\mathrm{ENV}} = \mathcal{E}'\end{array}}{[\![\mathit{expr}]\!]_{\mathrm{ENV}} = \mathcal{E} \quad [\![\mathit{id}]\!]_{\mathrm{ENV}} = \mathcal{E}'}$$

For all other phrases, the constituents of a phrase inherit the environment of the phrase itself.

Given a declaration, the auxiliary function nm returns the identifier declared by the declaration. The left associative operator $\triangleright$ is defined as follows:

$$[\mathcal{E} \triangleright (\mathit{identifier} \mapsto \mathit{decl})](\mathit{identifier}') = \begin{cases} \mathit{decl} & \mathit{identifier}' = \mathit{identifier} \\ \mathcal{E}(\mathit{identifier}') & \text{otherwise} \end{cases}$$

The semantic function

$$[\![\cdot]\!]_{\mathrm{BIND}}\colon \mathcal{I} \to \mathcal{D}$$

maps an identifier to its declaration. It is defined as

$$\frac{[\![\mathit{identifier}]\!]_{\mathrm{ENV}} = \mathcal{E}}{[\![\mathit{identifier}]\!]_{\mathrm{BIND}} = \mathcal{E}(\mathit{identifier})}$$

whenever *identifier* appears in (*named type*), (*variable access*), (*component access*) or (*function call*).

3.2 Type system

Set

$$\begin{array}{llr} \mathcal{T}_d = & \{\mathbf{void}, \mathbf{bool}, \mathbf{char}, \mathbf{int}\} & \text{(atomic types)} \\ & \cup\ \{\mathbf{arr}(n \times \tau) \mid n > 0 \wedge \tau \in \mathcal{T}_d\} & \text{(arrays)} \\ & \cup\ \{\mathbf{rec}_{id_1,\ldots,id_n}(\tau_1,\ldots,\tau_n) \mid n > 0 \wedge \tau_1,\ldots,\tau_n \in \mathcal{T}_d\} & \text{(records)} \\ & \cup\ \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} & \text{(pointers)} \end{array}$$

denotes a set of all data types of PREV'19. Set

$$\begin{array}{llr} \mathcal{T} = & \mathcal{T}_d & \text{(data types)} \\ & \cup\ \{(\tau_1,\ldots,\tau_n) \to \tau \mid n \geq 0 \wedge \tau_1,\ldots,\tau_n,\tau \in \mathcal{T}_d\} & \text{(functions)} \end{array}$$

denotes a set of all types of PREV'19.

Two types are equal if they share the same structure.

Semantic functions

$$[\![\cdot]\!]_{\mathrm{ISTYPE}}\colon \mathcal{P} \to \mathcal{T} \quad \text{and} \quad [\![\cdot]\!]_{\mathrm{OFTYPE}}\colon \mathcal{P} \to \mathcal{T}$$

maps phrases of PREV'19 to types. Function $[\![\cdot]\!]_{\mathrm{ISTYPE}}$ denotes the type described by a phrase, function $[\![\cdot]\!]_{\mathrm{OFTYPE}}$ denotes the type of a value described by a phrase.

Type expressions.

$$\frac{}{[\![\texttt{void}]\!]_{\text{ISTYPE}} = \textbf{void}} \quad \frac{}{[\![\texttt{bool}]\!]_{\text{ISTYPE}} = \textbf{bool}} \quad \frac{}{[\![\texttt{char}]\!]_{\text{ISTYPE}} = \textbf{char}} \quad \frac{}{[\![\texttt{int}]\!]_{\text{ISTYPE}} = \textbf{int}} \tag{T1}$$

$$\frac{\begin{array}{c}[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \quad \text{val}(\mathit{int}) = n \\ n > 0 \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}\end{array}}{[\![\texttt{arr[}\mathit{int}\texttt{]}\,\mathit{type}]\!]_{\text{ISTYPE}} = \textbf{arr}(n \times \tau)} \tag{T2}$$

$$\frac{\begin{array}{c}[\![\mathit{type}_1]\!]_{\text{ISTYPE}} = \tau_1 \;\ldots\; [\![\mathit{type}_n]\!]_{\text{ISTYPE}} = \tau_n \\ \forall i \in \{1 \ldots n\} \colon \tau_i \in \mathcal{T}_d \setminus \{\textbf{void}\}\end{array}}{[\![\texttt{rec(}\mathit{id}_1 \texttt{:} \mathit{type}_1\texttt{,} \ldots \texttt{,} \mathit{id}_n \texttt{:} \mathit{type}_n\texttt{)}]\!]_{\text{ISTYPE}} = \textbf{rec}_{\mathit{id}_1,\ldots,\mathit{id}_n}(\tau_1, \ldots, \tau_n)} \tag{T3}$$

$$\frac{[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \quad \tau \in \mathcal{T}_d}{[\![\texttt{ptr}\ \mathit{type}]\!]_{\text{ISTYPE}} = \textbf{ptr}(\tau)} \tag{T4}$$

$$\frac{[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau}{[\![\texttt{(}\mathit{type}\texttt{)}]\!]_{\text{ISTYPE}} = \tau} \tag{T5}$$

Value expressions.

$$\frac{}{[\![\texttt{none}]\!]_{\text{OFTYPE}} = \textbf{void}} \quad \frac{}{[\![\texttt{null}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\textbf{void})} \quad \frac{}{[\![\mathit{string}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\textbf{char})} \tag{V1}$$

$$\frac{}{[\![\mathit{bool}]\!]_{\text{OFTYPE}} = \textbf{bool}} \quad \frac{}{[\![\mathit{char}]\!]_{\text{OFTYPE}} = \textbf{char}} \quad \frac{}{[\![\mathit{int}]\!]_{\text{OFTYPE}} = \textbf{int}} \tag{V2}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{bool}}{[\![\texttt{!}\ \mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{bool}} \quad \frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{int} \quad \mathit{op} \in \{\texttt{+}, \texttt{-}\}}{[\![\mathit{op}\ \mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{int}} \tag{V3}$$

$$\frac{[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \textbf{bool} \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \textbf{bool} \quad \mathit{op} \in \{\texttt{\&}, \texttt{|}, \texttt{\^{}}\}}{[\![\mathit{expr}_1\ \mathit{op}\ \mathit{expr}_2]\!]_{\text{OFTYPE}} = \textbf{bool}} \tag{V4}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \tau \\ \tau \in \{\textbf{char}, \textbf{int}\} \quad \mathit{op} \in \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{\%}\}\end{array}}{[\![\mathit{expr}_1\ \mathit{op}\ \mathit{expr}_2]\!]_{\text{OFTYPE}} = \textbf{int}} \tag{V5}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \tau \\ \tau \in \{\textbf{bool}, \textbf{char}, \textbf{int}\} \cup \{\textbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \quad \mathit{op} \in \{\texttt{==}, \texttt{!=}\}\end{array}}{[\![\mathit{expr}_1\ \mathit{op}\ \mathit{expr}_2]\!]_{\text{OFTYPE}} = \textbf{bool}} \tag{V6}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \tau \\ \tau \in \{\textbf{char}, \textbf{int}\} \cup \{\textbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \quad \mathit{op} \in \{\texttt{<=}, \texttt{>=}, \texttt{<}, \texttt{>}\}\end{array}}{[\![\mathit{expr}_1\ \mathit{op}\ \mathit{expr}_2]\!]_{\text{OFTYPE}} = \textbf{bool}} \tag{V7}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}}{[\![\texttt{\$}\ \mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\tau)} \quad \frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\tau) \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}}{[\![\texttt{@}\ \mathit{expr}]\!]_{\text{OFTYPE}} = \tau} \tag{V8}$$

$$\frac{[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}}{[\![\texttt{new(}\mathit{type}\texttt{)}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\tau)} \quad \frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \textbf{ptr}(\tau) \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}}{[\![\texttt{del(}\mathit{expr}\texttt{)}]\!]_{\text{OFTYPE}} = \textbf{void}} \tag{V9}$$

$$\frac{[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \mathbf{arr}(n \times \tau) \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \mathbf{int}}{[\![\mathit{expr}_1\,\mathtt{[}\,\mathit{expr}_2\,\mathtt{]}]\!]_{\text{OFTYPE}} = \tau} \tag{v10}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{rec}_{id_1,\ldots,id_n}(\tau_1,\ldots,\tau_n) \quad \mathit{identifier} = id_i}{[\![\mathit{expr}\,\mathtt{.}\,\mathit{identifier}]\!]_{\text{OFTYPE}} = \tau_i} \tag{v11}$$

$$\frac{\begin{array}{c}[\![\mathit{identifier}]\!]_{\text{OFTYPE}} = (\tau_1,\ldots,\tau_n) \to \tau \quad [\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau_1 \;\ldots\; [\![\mathit{expr}_n]\!]_{\text{OFTYPE}} = \tau_n \\ \forall i \in \{1\ldots n\}\colon \tau_i \in \{\mathbf{bool},\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \\ \tau \in \{\mathbf{void},\mathbf{bool},\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![\mathit{identifier}\,\mathtt{(}\,\mathit{expr}_1\,\mathtt{,}\,\ldots\,\mathtt{,}\,\mathit{expr}_n\,\mathtt{)}]\!]_{\text{OFTYPE}} = \tau} \tag{v12}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau}{[\![\mathtt{\{}\,\mathit{stmts}\,\mathtt{:}\,\mathit{expr}\;\mathtt{where}\;\mathit{decls}\,\mathtt{\}}]\!]_{\text{OFTYPE}} = \tau} \tag{v13}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau_1 \quad [\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau_2 \\ \tau_1,\tau_2 \in \{\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![\mathtt{(}\,\mathit{expr}\,\mathtt{:}\,\mathit{type}\,\mathtt{)}]\!]_{\text{OFTYPE}} = \tau_2} \tag{v14}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau}{[\![\mathtt{(}\,\mathit{expr}\,\mathtt{)}]\!]_{\text{OFTYPE}} = \tau} \tag{v15}$$

Statements.

$$\frac{}{[\![\mathit{expr}\,\mathtt{;}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s1}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \tau \\ \tau \in \{\mathbf{bool},\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![\mathit{expr}_1\mathtt{=}\mathit{expr}_2\,\mathtt{;}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s2}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}]\!]_{\text{OFTYPE}} = \mathbf{void}}{[\![\mathtt{if}\;\mathit{expr}\;\mathtt{then}\;\mathit{stmts}\;\mathtt{end;}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s3}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}_1]\!]_{\text{OFTYPE}} = \mathbf{void} \quad [\![\mathit{stmts}_2]\!]_{\text{OFTYPE}} = \mathbf{void}}{[\![\mathtt{if}\;\mathit{expr}\;\mathtt{then}\;\mathit{stmts}_1\;\mathtt{else}\;\mathit{stmts}_2\;\mathtt{end;}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s4}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}]\!]_{\text{OFTYPE}} = \mathbf{void}}{[\![\mathtt{while}\;\mathit{expr}\;\mathtt{do}\;\mathit{stmts}\;\mathtt{end;}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s5}$$

Declarations.

$$\frac{[\![\mathit{identifier}]\!]_{\text{BIND}} = \mathtt{typ}\;\mathit{identifier}\,\mathtt{:}\,\mathit{type} \quad [\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau}{[\![\mathit{identifier}]\!]_{\text{ISTYPE}} = \tau} \tag{D1}$$

$$\frac{\begin{array}{c}[\![\mathit{identifier}]\!]_{\text{BIND}} = \mathtt{var}\;\mathit{identifier}\,\mathtt{:}\,\mathit{type} \quad [\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \\ \tau \in \mathcal{T}_d \setminus \{\mathbf{void}\}\end{array}}{[\![\mathit{identifier}]\!]_{\text{OFTYPE}} = \tau} \tag{D2}$$

$$\frac{\begin{array}{c}[\![identifier]\!]_{\text{BIND}} = \texttt{fun}\ identifier\,\texttt{(}identifer_1\,\texttt{:}\,type_1\,\texttt{,}\ \ldots\ \texttt{,}\,identifer_n\,\texttt{:}\,type_n\texttt{)}\,\texttt{:}\,type \\ [\![type_1]\!]_{\text{ISTYPE}} = \tau_1\ \ \ldots\ \ [\![type_n]\!]_{\text{ISTYPE}} = \tau_n \quad [\![type]\!]_{\text{ISTYPE}} = \tau \\ \forall i \in \{1\ldots n\}\colon \tau_i \in \{\mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \\ \tau \in \{\mathbf{void}, \mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![identifier]\!]_{\text{OFTYPE}} = (\tau_1, \ldots, \tau_n) \to \tau} \tag{D3}$$

3.3 Lvalues

The semantic function

$$[\![\cdot]\!]_{\text{ISADDR}}\colon \mathcal{P} \to \{\mathbf{true}, \mathbf{false}\}$$

denotes which phrases represent lvalues.

$$\frac{[\![identifier]\!]_{\text{BIND}} = \text{variable declaration}}{[\![identifier]\!]_{\text{ISADDR}} = \mathbf{true}} \qquad \frac{[\![identifier]\!]_{\text{BIND}} = \text{parameter declaration}}{[\![identifier]\!]_{\text{ISADDR}} = \mathbf{true}}$$

$$\frac{[\![expr]\!]_{\text{OFTYPE}} = \mathbf{ptr}(\tau)}{[\![\texttt{@}\,expr]\!]_{\text{ISADDR}} = \mathbf{true}} \qquad \frac{[\![expr]\!]_{\text{ISADDR}} = \mathbf{true}}{[\![expr\,\texttt{[}expr'\texttt{]}]\!]_{\text{ISADDR}} = \mathbf{true}} \qquad \frac{[\![expr]\!]_{\text{ISADDR}} = \mathbf{true}}{[\![expr\,\texttt{.}\,identifier\texttt{]}]\!]_{\text{ISADDR}} = \mathbf{true}}$$

In all other cases the value of $[\![\cdot]\!]_{\text{ISADDR}}$ equals **false**.

3.4 Operational semantics

Operational semantics is described by semantic functions

$$\begin{aligned}[\![\cdot]\!]_{\text{ADDR}} &: \mathcal{P} \times \mathcal{M} \to \mathcal{I} \times \mathcal{M} \\ [\![\cdot]\!]_{\text{EXPR}} &: \mathcal{P} \times \mathcal{M} \to \mathcal{I} \times \mathcal{M} \\ [\![\cdot]\!]_{\text{STMT}} &: \mathcal{P} \times \mathcal{M} \to \mathcal{M}\end{aligned}$$

where P denotes the set of phrases of PREV'19, I denotes the set of 64-bit integers, and M denotes possible states of the memory. Unary operators and binary operators perform 64-bit signed operations (except for type **char** where operations are performed on the lower 8 bits only).

Auxilary function addr returns either an absolute address for a static variable or a string constant or an offset for a local variable, parameter or record component. Auxilary function sizeof returns the size of an array. Auxilary function val returns the value of an integer constant or an ASCII code of a char constant.

Addresses.

$$\frac{}{[\![string]\!]^{\text{M}}_{\text{ADDR}} = \langle \text{addr}(string), \text{M}\rangle} \tag{A1}$$

$$\frac{\text{addr}(identifier) = a}{[\![identifier]\!]^{\text{M}}_{\text{ADDR}} = \langle a, \text{M}\rangle} \tag{A2}$$

$$\frac{[\![expr_1]\!]^{\text{M}}_{\text{ADDR}} = \langle n_1, \text{M}'\rangle \quad [\![expr_2]\!]^{\text{M}'}_{\text{EXPR}} = \langle n_2, \text{M}''\rangle \quad [\![expr_1]\!]_{\text{OFTYPE}} = \mathbf{arr}(n \times \tau)}{[\![expr_1\,\texttt{[}expr_2\texttt{]}]\!]^{\text{M}}_{\text{ADDR}} = \langle n_1 + n_2 * \text{sizeof}(\tau), \text{M}''\rangle} \tag{A3}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1, \mathrm{M}' \rangle}{[\![\mathit{expr}\,.\,\mathit{identifier}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1 + \mathrm{addr}(\mathit{identifier}), \mathrm{M}' \rangle} \tag{A4}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{\$}\ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n, \mathrm{M}' \rangle} \quad \frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{@}\ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n, \mathrm{M}' \rangle} \tag{A5}$$

Expressions.

$$\frac{}{[\![\texttt{none}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle 0, \mathrm{M} \rangle} \quad \frac{}{[\![\texttt{null}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle 0, \mathrm{M} \rangle} \tag{EX1}$$

$$\frac{}{[\![\texttt{true}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle 1, \mathrm{M} \rangle} \quad \frac{}{[\![\texttt{true}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle 0, \mathrm{M} \rangle} \tag{EX2}$$

$$\frac{}{[\![\mathit{char}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{val}(\mathit{char}), \mathrm{M} \rangle} \quad \frac{}{[\![\mathit{int}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{val}(\mathit{int}), \mathrm{M} \rangle} \tag{EX3}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle \quad \mathrm{op} \in \{\texttt{!}, \texttt{+}, \texttt{-}\}}{[\![\mathrm{op}\ \ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{op}\ n \rangle} \tag{EX4}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n_1, \mathrm{M}' \rangle \quad [\![\mathit{expr}_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle n_2, \mathrm{M}'' \rangle \\ \mathrm{op} \in \{\texttt{|}, \texttt{\^{}}, \texttt{\&}, \texttt{==}, \texttt{!=}, \texttt{<=}, \texttt{>=}, \texttt{<}, \texttt{>}, \texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{\%}\}\end{array}}{[\![\mathit{expr}_1\ \ \mathrm{op}\ \ \mathit{expr}_2]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n_1\ \mathrm{op}\ n_2, \mathrm{M}'' \rangle} \tag{EX5}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{\$}\ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle} \quad \frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{@}\ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{M}'[n], \mathrm{M}' \rangle} \tag{EX6}$$

$$\frac{[\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \tau}{[\![\texttt{new(}\mathit{expr}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{new}(\mathrm{sizeof}(\tau)), \mathrm{M} \rangle} \quad \frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{del(}\mathit{expr}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{del}(n), \mathrm{M}' \rangle} \tag{EX7}$$

$$\frac{\mathrm{addr}(\mathit{identifier}) = a}{[\![\mathit{identifier}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle M[a], \mathrm{M} \rangle} \tag{EX8}$$

$$\frac{[\![\mathit{expr}_1]\!]^{\mathrm{M}_0}_{\mathrm{EXPR}} = \langle n_1, \mathrm{M}_1 \rangle \ \ldots\ [\![\mathit{expr}_m]\!]^{\mathrm{M}_{m-1}}_{\mathrm{EXPR}} = \langle n_1, \mathrm{M}_m \rangle}{[\![\mathit{identifier}\texttt{(}\mathit{expr}_1\texttt{,}\ \ldots\ \texttt{,}\ \mathit{expr}_m\texttt{)}]\!]^{\mathrm{M}_0}_{\mathrm{EXPR}} = \langle \mathit{identifier}(), \mathrm{M}_m \rangle} \tag{EX9}$$

$$\frac{[\![\mathit{stmts}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \mathrm{M}' \quad [\![\mathit{expr}]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'' \rangle}{[\![\texttt{\{}\mathit{stmts}\texttt{:}\mathit{expr}\ \texttt{where}\ \mathit{decls}\texttt{\}}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'' \rangle} \tag{EX10}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle \quad [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} \neq \mathbf{char}}{[\![\texttt{(}\mathit{expr}\texttt{:}\mathit{type}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle} \tag{EX11}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle \quad [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \mathbf{char}}{[\![\texttt{(}\mathit{expr}\texttt{:}\mathit{type}\texttt{)}\ \ \mathit{rpar}^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n \,\%\, 256, \mathrm{M}' \rangle} \tag{EX12}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle}{[\![\texttt{(}\mathit{expr}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle} \tag{EX13}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n, \mathrm{M}' \rangle}{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle M'[n], \mathrm{M}' \rangle} \tag{EX14}$$

Statements.

$$\frac{[\![expr]\!]^{M}_{\mathrm{EXPR}} = \langle n, \mathrm{M}' \rangle}{[\![expr\texttt{;}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \mathrm{M}' = \mathrm{M}} \tag{ST1}$$

$$\frac{\begin{array}{c}[\![expr_1]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1, \mathrm{M}' \rangle \quad [\![expr_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle n_2, \mathrm{M}'' \rangle \\ \mathrm{M}''' = \mathrm{M}'' \text{ except for } \mathrm{M}'''[n1] = n2\end{array}}{[\![expr_1\texttt{=}expr_2\texttt{;}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \mathrm{M}'''} \tag{ST2}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{true}, \mathrm{M}' \rangle \quad [\![stmts]\!]^{\mathrm{M}'}_{\mathrm{STMT}} = \mathrm{M}''}{[\![\texttt{if } expr \texttt{ then } stmts \texttt{ end;}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \mathrm{M}''} \tag{ST3}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{false}, \mathrm{M}' \rangle}{[\![\texttt{if } expr \texttt{ then } stmts \texttt{ end;}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \mathrm{M}'} \tag{ST4}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{true}, \mathrm{M}' \rangle \quad [\![stmts_1]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \mathrm{M}''}{[\![\texttt{if } expr \texttt{ then } stmts_1 \texttt{ else } stmts_2 \texttt{ end;}]\!]_{\mathrm{OFTYPE}} = \mathrm{M}''} \tag{ST5}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{false}, \mathrm{M}' \rangle \quad [\![stmts_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \mathrm{M}''}{[\![\texttt{if } expr \texttt{ then } stmts_1 \texttt{ else } stmts_2 \texttt{ end;}]\!]_{\mathrm{OFTYPE}} = \mathrm{M}''} \tag{ST6}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{true}, \mathrm{M}' \rangle \quad [\![stmts]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \mathrm{M}''}{[\![\texttt{while } expr \texttt{ do } stmts \texttt{ end;}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = [\![\texttt{while } expr \texttt{ do } stmts \texttt{ end;}]\!]^{\mathrm{M}''}_{\mathrm{EXPR}}} \tag{ST7}$$

$$\frac{[\![expr]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathbf{false}, \mathrm{M}' \rangle}{[\![\texttt{while } expr \texttt{ do } stmts \texttt{ end;}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \mathrm{M}'} \tag{ST8}$$

$$\frac{[\![stmt_1]\!]^{\mathrm{M}_0}_{\mathrm{STMT}} = \mathrm{M}_1 \ \ldots \ [\![stmt_m]\!]^{M_{m-1}}_{\mathrm{STMT}} = \mathrm{M}_m}{} \tag{ST9}$$