The PREV'23 Programming Language

1 Lexical structure

Programs in the PREV'23 programming language are written in ASCII character set (no additional characters denoting post-alveolar consonants are allowed).

Programs in the PREV'23 programming language consist of the following lexical elements:

- *Constants*:
  - constant of type void: `none`
  - constants of type boolean: `true` `false`
  - constants of type integer:
    A nonempty finite string of digits (`0`...`9`), not 0 padded but optionally preceded by a sign (`+` or `-`).
  - constants of type char:
    A character with ASCII code in range {32...126} enclosed in single quotes (`'`); a single quote in the constant (except at the beginning and at the end) must be preceded with backslash (`\`).
  - string constants:
    A (possibly empty) string of character with ASCI codes in range {32...126} enclosed in double quotes (`"`); each double quote in the constant (except at the beginning and at the end) must be preceded with a backslash (`\`).
  - constants of pointer types: `nil`
- *Symbols*:
  `( ) { } [ ] . , : ; & | ! == != < > <= >= * / % + - ^ =`
- *Keywords*:
  `bool char del do else fun if in int let new then typ var void 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 constant.
- *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 are recognised from left to right using the longest match approach.

2 Syntax structure

The concrete syntax of the PREV'23 programming language is defined by context free grammar with the start symbol *declarations* and the following productions:

declarations
$\longrightarrow$ (*type-declarations* | *function-declarations* | *variable-declarations*)
{ (*type-declarations* | *function-declarations* | *variable-declarations*) }

type-declarations
⟶ `typ` identifier `=` *type* { , identifier `=` *type* } `;`

function-declarations
⟶ `fun` identifier `(` [identifier `:` *type* { , identifier `:` *type* }] `)` `:` *type* [`=` *statement*]
{ , identifier `(` [identifier `:` *type* { , identifier `:` *type* }] `)` `:` *type* [`=` *statement*] } `;`

variable-declarations
⟶ `var` identifier `:` *type* { , identifier `:` *type* } `;`

type
⟶ `void` | `char` | `int` | `bool` | identifier
⟶ `[` *expression* `]` *type*
⟶ `^` *type*
⟶ `{` identifier `:` *type* { , identifier `:` *type* } `}`
⟶ `(` *type* `)`

expression
⟶ constant
⟶ identifier [`(` [*expression* { , *expression* }] `)`]
⟶ *expression* (`[` *expression* `]` | `^` | `.` identifier)
⟶ unary-operator *expression*
⟶ *expression* binary-operator *expression*
⟶ `(` *expression* [`:` *type*] `)`
⟶ `new` `(` *type* `)` | `del` `(` *expression* `)`

statement
⟶ *expression* [`=` *expression*]
⟶ `if` *expression* `then` *statement* [`else` *statement*]
⟶ `while` *expression* `do` *statement*
⟶ `let` *declarations* `in` *statement*
⟶ `{` *statement* { `;` *statement* } `}`

Unary operators are `!`, `+`, `-` and `^`.
Binary operators are `|`, `&`, `==`, `!=`, `<`, `>`, `<=`, `>=`, `*`, `/`, `%`, `+` and `-`.

The precedence of the operators is as follows:

postfix operators	`[] ^ .`	THE HIGHEST PRECEDENCE
prefix operators	`! + - ^`	
multiplicative operators	`* / %`	
additive operators	`+ -`	
relational operators	`== != < > <= >=`	
conjunctive operator	`&`	
disjunctive operator	`|`	THE LOWEST PRECEDENCE

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

In the grammar above, braces typeset as { } enclose sentential forms that can be repeated zero or more times, brackets typeset as [ ] enclose sentential forms that can be present or not while braces typeset as `{}` and brackets typeset as `[]` denote symbols that are a part of the program text.

3 Semantic structure

Let function $[\![\cdot]\!]_{\text{BIND}}$ bind a name to its declaration according to the rules of namespaces and scopes described below. Hence, the value of function $[\![\cdot]\!]_{\text{BIND}}$ depends on the context of its argument.

3.1 Name binding

Namespaces. There are two kinds of a 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. Each global `typ`, `fun` and `var` declaration group creates a new scope extending to the end of file. All names declared in the same declaration group belong to the same scope.
2. Each `typ`, `fun` and `var` declaration group within a `let` *declarations* `in` *statement* creates a new scope extending to the end of *statement*. All names declared in the same declaration group belong to the same scope.
3. Function declaration

   identifier `(` [identifier `:` *type* { `,` identifier `:` *type* }] `)` `:` *type* [`=` *statement*]

   creates a new scope. The name of a function, the types of parameters and the type of a result belong to the outer scope while the names of parameters and the *statement* (if present) denoting the function body belong to the 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.

3.2 Type system

The set

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

denotes the set of all data types of PREV'23. The set

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

denotes the set of all types of PREV'23.

Structural equality of types: Types τ_1 and τ_2 are equal if (a) $\tau_1 = \tau_2$ or (b) if they are type synonyms (introduced by chains of type declarations) of types τ_1' and τ_2' where $\tau_1' = \tau_2'$.

Semantic functions

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

map syntactic phrases of PREV'23 to types. Function $[\![\cdot]\!]_{\text{ISTYPE}}$ denotes the type described by a phrase, function $[\![\cdot]\!]_{\text{OFTYPE}}$ denotes the type of a value described by a phrase.

The following assumptions are made in the rules below:

- Function val maps lexemes to data of the specified type.
- $\tau \in \mathcal{T}_d$ unless specified otherwise.

Type expressions.

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

$$\frac{[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \quad \mathrm{val}(\mathit{int}) = n \quad 0 < n \leq 2^{63}-1 \quad \tau \in \mathcal{T}_d \setminus \{\textbf{void}\}}{[\![\texttt{[}\mathit{int}\texttt{]}\,\mathit{type}]\!]_{\text{ISTYPE}} = \textbf{arr}(n \times \tau)} \tag{T2}$$

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

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

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

Value expressions.

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

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

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

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

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

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

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

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{\^{}}\ \mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{ptr}(\tau)} \quad \frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{ptr}(\tau)}{[\![\mathit{expr}\ \texttt{\^{}}]\!]_{\text{OFTYPE}} = \tau} \tag{v8}$$

$$\frac{[\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau \quad \tau \in \mathcal{T}_d}{[\![\texttt{new(}\mathit{type}\texttt{)}]\!]_{\text{OFTYPE}} = \mathbf{ptr}(\tau)} \quad \frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{ptr}(\tau)}{[\![\texttt{del(}\mathit{expr}\texttt{)}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{v9}$$

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

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

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

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau_1 \quad [\![\mathit{type}]\!]_{\text{ISTYPE}} = \tau_2 \quad \tau_1,\tau_2 \in \{\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}}{[\![\texttt{(}\mathit{expr}\,\texttt{:}\,\mathit{type}\texttt{)}]\!]_{\text{OFTYPE}} = \tau_2} \tag{v13}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{(}\mathit{expr}\texttt{)}]\!]_{\text{OFTYPE}} = \tau} \tag{v14}$$

Statements.

$$\frac{[\![\mathit{expr}_1]\!]_{\text{OFTYPE}} = \tau \quad [\![\mathit{expr}_2]\!]_{\text{OFTYPE}} = \tau \quad \tau \in \{\mathbf{bool},\mathbf{char},\mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}}{[\![\mathit{expr}_1\,\texttt{=}\,\mathit{expr}_2]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s1}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{stmts}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s2}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}_1]\!]_{\text{OFTYPE}} = \tau_1 \quad [\![\mathit{stmts}_2]\!]_{\text{OFTYPE}} = \tau_2}{[\![\texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{stmts}_1\ \texttt{else}\ \mathit{stmts}_2]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s3}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \mathbf{bool} \quad [\![\mathit{stmts}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{while}\ \mathit{expr}\ \texttt{do}\ \mathit{stmts}]\!]_{\text{OFTYPE}} = \mathbf{void}} \tag{s4}$$

$$\frac{[\![\mathit{expr}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{(}\mathit{expr}\texttt{)}]\!]_{\text{OFTYPE}} = \tau} \quad \frac{[\![\mathit{stmt}]\!]_{\text{OFTYPE}} = \tau}{[\![\texttt{let}\ \mathit{decls}\ \texttt{in}\ \mathit{stmt}]\!]_{\text{OFTYPE}} = \tau} \tag{s5}$$

$$\frac{\forall i \in \{1\ldots n\}\colon [\![\mathit{stmt}_i]\!]_{\text{OFTYPE}} = \tau_i}{[\![\texttt{\{}\mathit{stmt}_1\,\texttt{;}\ldots\texttt{;}\,\mathit{stmt}_n\texttt{\}}]\!]_{\text{OFTYPE}} = \tau_n} \tag{s6}$$

Declarations.

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

$$\frac{[\![\mathit{identifier}]\!]_{\mathrm{BIND}} = \texttt{var}\ \mathit{identifier}\,\texttt{:}\,\mathit{type} \quad [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \tau \quad \tau \in \mathcal{T}_d \setminus \{\mathbf{void}\}}{[\![\mathit{identifier}]\!]_{\mathrm{OFTYPE}} = \tau} \tag{D2}$$

$$\frac{\begin{array}{c}[\![\mathit{identifier}]\!]_{\mathrm{BIND}} = \texttt{fun}\ \mathit{identifier}\,\texttt{(}\mathit{identifer}_1\,\texttt{:}\,\mathit{type}_1\texttt{,}\ldots\texttt{,}\mathit{identifer}_n\,\texttt{:}\,\mathit{type}_n\texttt{)}\,\texttt{:}\,\mathit{type} \\ \forall i \in \{1\ldots n\}\colon [\![\mathit{type}_i]\!]_{\mathrm{ISTYPE}} = \tau_i \wedge \tau_i \in \{\mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \\ [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \tau \quad \tau \in \{\mathbf{void}, \mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![\mathit{identifier}]\!]_{\mathrm{OFTYPE}} = (\tau_1, \ldots, \tau_n) \to \tau} \tag{D3}$$

$$\frac{\begin{array}{c}[\![\mathit{identifier}]\!]_{\mathrm{BIND}} = \texttt{fun}\ \mathit{identifier}\,\texttt{(}\mathit{identifer}_1\,\texttt{:}\,\mathit{type}_1\texttt{,}\ldots\texttt{,}\mathit{identifer}_n\,\texttt{:}\,\mathit{type}_n\texttt{)}\,\texttt{:}\,\mathit{type}\,\texttt{=}\,\mathit{stmt} \\ \forall i \in \{1\ldots n\}\colon [\![\mathit{type}_i]\!]_{\mathrm{ISTYPE}} = \tau_i \wedge \tau_i \in \{\mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\} \\ [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \tau \quad [\![\mathit{stmt}]\!]_{\mathrm{OFTYPE}} = \tau \quad \tau \in \{\mathbf{void}, \mathbf{bool}, \mathbf{char}, \mathbf{int}\} \cup \{\mathbf{ptr}(\tau) \mid \tau \in \mathcal{T}_d\}\end{array}}{[\![\mathit{identifier}]\!]_{\mathrm{OFTYPE}} = (\tau_1, \ldots, \tau_n) \to \tau} \tag{D4}$$

3.3 Lvalues

The semantic function

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

denotes which phrases represent lvalues.

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

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

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

3.4 Linkage

A variable or a function has external linkage if

- it is not declared inside a function and
- its declaration does not redeclare a name already declared at any outer scope.

3.5 Operational semantics

Operational semantics is described by semantic functions

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

where P denotes the set of phrases of PREV'23, 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 a type. Auxilary function val returns the value of an integer constant or an ASCII code of a char constant.

Addresses.

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

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

$$\frac{[\![\mathit{expr}_1]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1, \mathrm{M}'\rangle \quad [\![\mathit{expr}_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle n_2, \mathrm{M}''\rangle \quad [\![\mathit{expr}_1]\!]_{\mathrm{OFTYPE}} = \mathbf{arr}(n \times \tau)}{[\![\mathit{expr_1}\,\texttt{[}\mathit{expr_2}\texttt{]}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1 + n_2 * \mathrm{sizeof}(\tau), \mathrm{M}''\rangle} \tag{A3}$$

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

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

Expressions.

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

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

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

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

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

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

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

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle a, \mathrm{M}'\rangle \quad [\![\mathrm{del}(a)]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle \mathrm{undef}, \mathrm{M}''\rangle}{[\![\texttt{del}\ \mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \mathrm{undef}, \mathrm{M}''\rangle} \tag{EX8}$$

$$\frac{\text{addr}(\mathit{identifier}) = a}{[\![\mathit{identifier}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle M[a], \mathrm{M}\rangle} \quad \text{(EX9)}$$

$$\frac{[\![\mathit{expr_1}\,\texttt{[}\mathit{expr_2}\texttt{]}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle a, M'\rangle}{[\![\mathit{expr_1}\,\texttt{[}\mathit{expr_2}\texttt{]}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle M'[a], \mathrm{M}'\rangle} \quad \text{(EX10)}$$

$$\frac{[\![\mathit{expr}\,\texttt{.}\,\mathit{identifier}]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle a, M'\rangle}{[\![\mathit{expr}\,\texttt{.}\,\mathit{identifier}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle M'[a], \mathrm{M}'\rangle} \quad \text{(EX11)}$$

$$\frac{[\![\mathit{expr}_1]\!]^{\mathrm{M}_0}_{\mathrm{EXPR}} = \langle n_1, \mathrm{M}_1\rangle \ \ldots \ [\![\mathit{expr}_m]\!]^{\mathrm{M}_{m-1}}_{\mathrm{EXPR}} = \langle n_m, \mathrm{M}_m\rangle}{[\![\mathit{identifier}\,\texttt{(}\mathit{expr}_1, \ldots, \mathit{expr}_m\texttt{)}]\!]^{\mathrm{M}_0}_{\mathrm{EXPR}} = \langle \mathit{identifier}(n_1, \ldots, n_m), \mathrm{M}_m\rangle} \quad \text{(EX12)}$$

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

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'\rangle \quad [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} \neq \textbf{char}}{[\![\texttt{(}\mathit{expr}\,\texttt{:}\,\mathit{type}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'\rangle} \quad \text{(EX14)}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'\rangle \quad [\![\mathit{type}]\!]_{\mathrm{ISTYPE}} = \textbf{char}}{[\![\texttt{(}\mathit{expr}\,\texttt{:}\,\mathit{type}\texttt{)}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle n \bmod 256, \mathrm{M}'\rangle} \quad \text{(EX15)}$$

Statements.

$$\frac{[\![\mathit{expr}]\!]^{M}_{\mathrm{EXPR}} = \langle n, \mathrm{M}'\rangle}{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \langle n, \mathrm{M}'\rangle} \quad \text{(ST1)}$$

$$\frac{\begin{array}{c}[\![\mathit{expr}_1]\!]^{\mathrm{M}}_{\mathrm{ADDR}} = \langle n_1, \mathrm{M}'\rangle \quad [\![\mathit{expr}_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle n_2, \mathrm{M}''\rangle \\ \forall a\colon \mathrm{M}'''[a] = \begin{cases} n_2 & a = n_1 \\ M''[a] & \text{otherwise}\end{cases}\end{array}}{[\![\mathit{expr}_1\texttt{=}\mathit{expr}_2]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}'''\rangle} \quad \text{(ST2)}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \textbf{true}, \mathrm{M}'\rangle \quad [\![\mathit{stmt}_1]\!]^{\mathrm{M}'}_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}''\rangle}{[\![\texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{stmt}_1\ \texttt{else}\ \mathit{stmt}_2]\!]_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}''\rangle} \quad \text{(ST3)}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \textbf{false}, \mathrm{M}'\rangle \quad [\![\mathit{stmt}_2]\!]^{\mathrm{M}'}_{\mathrm{EXPR}} = \langle \text{undef}, \mathrm{M}''\rangle}{[\![\texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{stmt}_1\ \texttt{else}\ \mathit{stmt}_2]\!]_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}''\rangle} \quad \text{(ST4)}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \textbf{true}, \mathrm{M}'\rangle \quad [\![\mathit{stmt}]\!]^{\mathrm{M}'}_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}''\rangle}{[\![\texttt{while}\ \mathit{expr}\ \texttt{do}\ \mathit{stmt}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = [\![\texttt{while}\ \mathit{expr}\ \texttt{do}\ \mathit{stmt}]\!]^{\mathrm{M}''}_{\mathrm{STMT}}} \quad \text{(ST5)}$$

$$\frac{[\![\mathit{expr}]\!]^{\mathrm{M}}_{\mathrm{EXPR}} = \langle \textbf{false}, \mathrm{M}'\rangle}{[\![\texttt{while}\ \mathit{expr}\ \texttt{do}\ \mathit{stmt}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \langle \text{undef}, \mathrm{M}'\rangle} \quad \text{(ST6)}$$

$$\frac{[\![\mathit{stmt}]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \langle n, \mathrm{M}'\rangle}{[\![\texttt{let}\ \mathit{decls}\ \texttt{in}\ \mathit{stmt}\ ]\!]^{\mathrm{M}}_{\mathrm{STMT}} = \langle n, \mathrm{M}'\rangle} \quad \text{(ST7)}$$

$$\frac{[\![\mathit{stmt}_1]\!]^{\mathrm{M}_0}_{\mathrm{STMT}} = \langle n_1, \mathrm{M}_1\rangle \ \ldots \ [\![\mathit{stmt}_m]\!]^{\mathrm{M}_{m-1}}_{\mathrm{STMT}} = \langle n_m, \mathrm{M}_m\rangle}{[\![\texttt{\{}\mathit{stmt}_1\,\texttt{;}\ldots\texttt{;}\,\mathit{stmt}_m\texttt{\}}]\!]^{\mathrm{M}_0}_{\mathrm{STMT}} = \langle n_m, \mathrm{M}_m\rangle} \quad \text{(ST8)}$$