

# 1 Abstract Syntax

This section explains abstract syntax of  $\text{IR}_{\text{ES}}$ .

$$\begin{aligned} n &\in \textit{FloatingPoint} \\ d &\in \textit{Integer} \\ s &\in \textit{String} \\ b &\in \textit{Boolean} \\ r &\in \textit{Reference} \\ x &\in \textit{Identifier} \\ t &\in \textit{Type} \end{aligned}$$

Program  $p ::= i; \dots; i$

Instruction	$i ::= e$	(expression)
	<b>let</b> $x = e$	(let)
	$r := e$	(assign)
	<b>delete</b> $r$	(delete)
	<b>append</b> $e \leftarrow e$	(append)
	<b>prepend</b> $e \rightarrow e$	(prepend)
	<b>return</b> $e$	(return)
	<b>if</b> $e$ $i$ $i$	(if-then-else)
	<b>while</b> $e$ $i$	(while)
	$\{i^*\}$	(sequence)
	<b>assert</b> $e$	(assert)
	<b>print</b> $e$	(print)
	<b>app</b> $x = (e \ e^*)$	(function application)
	<b>access</b> $x = (e \ e)$	(access)
	<b>withcont</b> $x \ (x^*) = i$	(continuation)

  

Reference	$r ::= x$	(identifier)
	$r[e]$	(reference to value of field in heap)

Expression	$e ::=$	$n$	(floating point number)
		$d$	(integer)
		$s$	(string)
		$b$	(boolean)
		$r$	(reference)
		<b>undefined</b>	(undefined)
		<b>null</b>	(null)
		<b>absent</b>	(absent)
		<b>new</b> $e$	(symbol)
		<b>new</b> $\langle e^* \rangle$	(list)
		<b>new</b> $t \{[e \mapsto e]^*\}$	(map)
		<b>pop</b> $e \ e$	(pop)
		<b>typeof</b> $e$	(typeof)
		<b>is-instance-of</b> $e \ s$	(is-instance-of)
		<b>get-elems</b> $e \ s$	(get-elements)
		<b>get-syntax</b> $e$	(get-syntax)
		<b>parse-syntax</b> $e \ e \ e^*$	(parse-syntax)
		<b>convert</b> $e \triangleright e^*$	(convert)
		<b>contains</b> $e \ e$	(contains)
		<b>copy-obj</b> $e$	(copy-object)
		<b>map-keys</b> $e$	(map-keys)
		<b>!!!</b> $s$	(not supported)
		$\odot \ e$	(unary operation)
		$e \oplus \ e$	(binary operation)
		$(x^*) \ [\Rightarrow] \ i$	(continuation)

UnaryOperator	$\odot$	::=	-	(negation)
			!	(boolean not)
			~	(bitwise not)
BinaryOperator	$\oplus$	::=	+	(addition)
			-	(subtraction)
			*	(multiplication)
			**	(power)
			/	(division)
			%	(modulo)
			%	(modulo)
			=	(equals)
			&&	(boolean and)
				(boolean or)
			^^	(boolean xor)
			&	(bitwise and)
				(bitwise or)
			^	(bitwise xor)
			<<	(shift left)
			<	(less-than)
			>>>	(unsigned shift right)
			>>	(shift right)
ConvertOperator	$\triangleright$	::=	str2num	(string to number)
			num2str	(number to string)
			num2int	(number to integer)

## 2 Operational Semantic

This section explains operational semantic of  $\text{IR}_{\text{ES}}$ .

### 2.1 Domain

Semantic domain of  $\text{IR}_{\text{ES}}$ .

State	$\sigma$	$\in$	$\text{Context} \times \text{Context}^* \times \text{Environment} \times \text{Heap}$
Context	$C$	$\in$	$\text{Identifier} \times \text{String} \times \text{Instruction}^* \times \text{Environment}$
Environment	$E$	$\in$	$\text{Identifier} \rightarrow \text{Value}$
Heap	$H$	$\in$	$\text{Address} \rightarrow \text{Object}$
Value	$v$	$\in$	$\text{Value}$
Address	$a$	$\in$	$\text{Address}$
Object	$o$	$\in$	$\text{Object}$

$$\text{State } \sigma ::= (C, C^*, E, H) \quad \text{Context } C ::= (x, s, i^*, E)$$

$$\text{Constant } c ::= n \mid d \mid s \mid b \mid \text{undefined} \mid \text{null} \mid \text{absent}$$

$$\begin{array}{ll} \text{Object } o ::= & \text{symbol } v \quad (\text{symbol}) \\ & \mid t \{ [v \mapsto v]^* \} \quad (\text{map}) \\ & \mid \langle v^* \rangle \quad (\text{list}) \\ & \mid \text{not-supported } s \quad (\text{not supported}) \end{array}$$

$$\begin{array}{ll} \text{Value } v ::= & a \quad (\text{address}) \\ & \mid c \quad (\text{constant}) \\ & \mid \lambda(s, x^*, x, i) \quad (\text{function}) \\ & \mid \kappa(C, C^*, x^*, i) \quad (\text{continuation}) \\ & \mid \text{ASTVal} \quad (\text{AST value}) \\ & \mid \text{ASTMethod } \lambda(s, x^*, x, i) E \quad (\text{AST method}) \end{array}$$

$$\begin{array}{ll} \text{RefValue } rv ::= & x \quad (\text{identifier}) \\ & \mid a[v] \quad (\text{reference to value of map in heap}) \\ & \mid s.v \quad (\text{reference to string field}) \end{array}$$

TODO

ASTValue notation

change ASTMethod notation

## 2.2 Semantic of $\text{IR}_{\text{ES}}$

- program : [[description of program execution]]
- instruction :  $\sigma \vdash i \Rightarrow \sigma$
- expression :  $\sigma \vdash e \Rightarrow v, \sigma$
- expression - escape completion :  $\sigma \vdash_{\text{escape}} e \Rightarrow v, \sigma$
- reference :  $\sigma \vdash r \Rightarrow rv, \sigma$
- reference value :  $\sigma \vdash rv \Rightarrow v, \sigma$
- unary operator :  $\odot v \Rightarrow v$
- binary operator :  $v \oplus v \Rightarrow v$

### 2.2.1 Instruction

$$\begin{array}{c}
\boxed{\sigma \vdash i \Rightarrow \sigma} \quad \frac{\sigma \vdash e \Rightarrow v, \sigma_0}{\sigma \vdash e \Rightarrow \sigma_0} \quad \frac{\sigma \vdash e \Rightarrow v, \sigma_0 \quad \text{define}(\sigma_0, x, v) = \sigma_1}{\sigma \vdash \text{let } x = e \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash r \Rightarrow x, \sigma_0 \quad \sigma_0 \vdash e \Rightarrow v, \sigma_1 \quad \text{updated}(\sigma_1, x, v) = \sigma_2}{\sigma \vdash r := e \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash r \Rightarrow a[v], \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_0, \sigma_1 \quad \text{updated}(\sigma_1, a[v], v_0) = \sigma_2}{\sigma \vdash r := e \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash r \Rightarrow x, \sigma_0 \quad \text{deleted}(\sigma_0, x) = \sigma_1}{\sigma \vdash \text{delete } r \Rightarrow \sigma_1} \quad \frac{\sigma \vdash r \Rightarrow a[v], \sigma_0 \quad \text{deleted}(\sigma_0, a[v]) = \sigma_1}{\sigma \vdash \text{delete } r \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e_0 \Rightarrow v, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_1 \Rightarrow a, \sigma_1 \quad \text{append}(\sigma_0??, a, v) = \sigma_2}{\sigma \vdash \text{append } e_0 \leftarrow e_1 \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash_{\text{escape}} e_0 \Rightarrow v, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_1 \Rightarrow a, \sigma_1 \quad \text{prepend}(\sigma_0??, a, v) = \sigma_2}{\sigma \vdash \text{prepend } e_0 \rightarrow e_1 \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash e \Rightarrow v, \sigma_0 \quad \text{return}(\sigma_0, v) = \sigma_1}{\sigma \vdash \text{return } e \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{true}, \sigma_0 \quad \sigma_0 = (C, C^*, E_G, H) \quad C = (x, s, i^*, E_L) \quad C_0 = (x, s, i_0 +: i^*, E_L) \quad \sigma_1 = (C_0, C^*, E_G, H)}{\sigma \vdash \text{if } e \ i_0 \ i_1 \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{false}, \sigma_0 \quad \sigma_0 = (C, C^*, E_G, H) \quad C = (x, s, i^*, E_L) \quad C_0 = (x, s, i_1 +: i^*, E_L) \quad \sigma_1 = (C_0, C^*, E_G, H)}{\sigma \vdash \text{if } e \ i_0 \ i_1 \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{true}, \sigma_0 \quad \sigma_0 = (C, C^*, E_G, H) \quad C = (x, s, i^*, E_L) \quad C_0 = (x, s, i_b +: i^*, E_L) \quad \sigma_1 = (C_0, C^*, E_G, H)}{\sigma \vdash \text{while } e \ i_b \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{false}, \sigma_0}{\sigma \vdash \text{while } e \ i_b \Rightarrow \sigma_0} \quad \frac{\sigma \vdash e \Rightarrow \text{true}, \sigma_0}{\sigma \vdash \text{assert } e \Rightarrow \sigma_0} \quad \frac{\sigma \vdash e \Rightarrow v, \sigma_0}{\sigma \vdash \text{print } e \Rightarrow \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad C = (x, s, i^*, E_L) \quad C_0 = (x, s, i_s^* ++ i^*, E_L) \quad \sigma_0 = (C_0, C^*, E_G, H)}{\sigma \vdash \{i_s^*\} \Rightarrow \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E, H) \quad \text{define}(\sigma, x_{id}, \kappa(C, C^*, x^*, i)) = \sigma_0}{\sigma \vdash \text{withcont } x_{id} \ (x^*) = i \Rightarrow \sigma_0}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \lambda(s_\lambda, x^*, x_{var}, i_\lambda) \\
\sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \\
\sigma_n = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad C_0 = (x_f, s, i^*, E_L) \\
E_{L_1} = [x \mapsto v]^* \quad x_1 \in \text{Identifier} \quad C_1 = (x_1, s_\lambda, i_\lambda, E_{L_1}) \quad \sigma_{next} = (C_1, C_0 +: C^*, E_G, H) \\
\hline
\sigma \vdash \text{app } x_f = (e_f \ e_0 \ \cdots \ e_n) \Rightarrow \sigma_{next}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \text{ASTMethod } \lambda(s_\lambda, x^*, x_{var}, i_\lambda) \ E_\lambda \\
\sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \\
\sigma_n = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad C_0 = (x_f, s, i^*, E_L) \\
E_{L_1} = E_\lambda + [x \mapsto v]^* \quad x_1 \in \text{Identifier} \quad C_1 = (x_1, s_\lambda, i_\lambda, E_{L_1}) \quad \sigma_{next} = (C_1, C_0 +: C^*, E_G, H) \\
\hline
\sigma \vdash \text{app } x_f = (e_f \ e_0 \ \cdots \ e_n) \Rightarrow \sigma_{next}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \kappa(C, C^*, x^*, i) \quad C = (x_c, s, i_c^*, E_L) \\
\sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \quad \sigma_n = (C_n, C_n^*, E_G, H) \\
E_{L_0} = E_L + [x \mapsto v]^* \quad C_0 = (x_c, s, i, E_{L_0}) \quad \sigma_{next} = (C_0, C^*, E_G, H) \\
\hline
\sigma \vdash \text{app } x_f = (e_f \ e_0 \ \cdots \ e_n) \Rightarrow \sigma_{next}
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow s_p, \sigma_1 \\
\text{assertEquals}(s_p, \text{"length"}) \quad \text{define}(\sigma_1, x, \text{length}(s)) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow s, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow d, \sigma_1 \quad \text{define}(\sigma_1, x, \text{charAt}(s, d)) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = \text{symbol } v_s \quad \text{getHeapProp}(H, a, v_p) = v_0 \\
\text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle \quad \text{getHeapProp}(H, a, v_p) = v_0 \\
\text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = t \ \{[v_k \mapsto v_v]^*\} \quad \text{assertNotEquals}(t, t_{\text{completion}}) \\
\text{getHeapProp}(H, a, v_p) = v_0 \quad \text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = t \ \{[v_k \mapsto v_v]^*\} \quad \text{assertrrtEquals}(t, t_{\text{completion}}) \quad v_p \in \{v_k\} \\
\text{getHeapProp}(H, a, v_p) = v_0 \quad \text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \\
v_p \notin \{v_k\} \quad \text{"Value"} \in \{v_k\} \quad [v_k \mapsto v_v]^*(\text{"Value"}) = a_0 \\
\text{getHeapProp}(H, a_0, v_p) = v_0 \quad \text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$
  

$$\begin{array}{c}
\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_p, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \\
a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \\
v_p \notin \{v_k\} \quad \text{"Value"} \in \{v_k\} \quad [v_k \mapsto v_v]^*(\text{"Value"}) = s \\
\text{getStringProp}(H, s, v_p) = v_0 \quad \text{define}(\sigma_1, x, v_0) = \sigma_2 \\
\hline
\sigma \vdash \text{access } x = (e_b \ e) \Rightarrow \sigma_2
\end{array}$$

TODO

access - ASTVal - Lexical

access - ASTVal

define helper function OR change function to rule explicitly



### 2.2.2 Expression

$$\boxed{\sigma \vdash e \Rightarrow v, \sigma} \quad \sigma \vdash n \Rightarrow n, \sigma \quad \sigma \vdash d \Rightarrow d, \sigma \quad \sigma \vdash s \Rightarrow s, \sigma \quad \sigma \vdash b \Rightarrow b, \sigma$$

$$\sigma \vdash \text{undefined} \Rightarrow \text{undefined}, \sigma \quad \sigma \vdash \text{null} \Rightarrow \text{null}, \sigma \quad \sigma \vdash \text{absent} \Rightarrow \text{absent}, \sigma$$

$$\frac{\sigma \vdash_{\text{escape}} e \Rightarrow v, \sigma_0 \quad \odot v \Rightarrow v_0}{\sigma \vdash \odot e \Rightarrow v_0, \sigma_0} \quad \frac{\sigma \vdash_{\text{escape}} e_0 \Rightarrow v_0, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_1 \Rightarrow v_1, \sigma_1 \quad v_0 \oplus v_1 \Rightarrow v_2}{\sigma \vdash e_0 \oplus e_1 \Rightarrow v_2, \sigma_1}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow v, \sigma_1}{\sigma \vdash r \Rightarrow v, \sigma_1}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 = (C, C^*, E, H) \quad \text{assertDynamicAddr}(a) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \quad [v_k \mapsto v_v]^*(\text{"Value"}) = v}{\sigma \vdash_{\text{escape}} r \Rightarrow v, \sigma_1}$$

$$\frac{\sigma = (C, C^*, E, H)}{\sigma \vdash (x^*) [\Rightarrow] i \Rightarrow \kappa(C, C^*, x^*, i), \sigma} \quad \frac{\sigma \vdash e \Rightarrow v, \sigma_0 \quad \text{getType}(v) = s_{\text{type}}}{\sigma \vdash \text{typeof } e \Rightarrow s_{\text{type}}, \sigma_0}$$

$$\frac{\sigma \vdash_{\text{escape}} e \Rightarrow s, \sigma_0 \quad \text{allocSymbol}(\sigma_0, s) = (a, \sigma_1)}{\sigma \vdash \text{new } e \Rightarrow a, \sigma_1}$$

$$\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{undefined}, \sigma_0 \quad \text{allocSymbol}(\sigma_0, \text{undefined}) = (a, \sigma_1)}{\sigma \vdash \text{new } e \Rightarrow a, \sigma_1}$$

$$\frac{\sigma \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \quad \text{allocList}(\sigma_n, v^*) = (a, \sigma_{\text{next}})}{\sigma \vdash \text{new } [e_0, \dots, e_n] \Rightarrow a, \sigma_{\text{next}}}$$

$$\frac{\begin{array}{c} \text{allocMap}(\sigma, t) = (a, \sigma_t) \\ \sigma_t \vdash_{\text{escape}} e_{k_0} \Rightarrow v_{k_0}, \sigma_{0_k} \quad \sigma_{0_k} \vdash e_{v_0} \Rightarrow v_{v_0}, \sigma_{0_v} \quad \text{updated}(\sigma_{0_v}, v_{k_0}, v_{v_0}) = \sigma_0 \\ \dots \\ \sigma_{n-1} \vdash_{\text{escape}} e_{k_n} \Rightarrow v_{k_n}, \sigma_{n_k} \quad \sigma_{n_k} \vdash e_{v_n} \Rightarrow v_{v_n}, \sigma_{n_v} \quad \text{updated}(\sigma_{n_v}, v_{k_n}, v_{v_n}) = \sigma_n \end{array}}{\sigma \vdash \text{new } t (e_{k_0} \mapsto e_{v_0}, \dots, e_{k_n} \mapsto e_{v_n}) \Rightarrow a, \sigma_n}$$

$$\frac{\sigma \vdash_{\text{escape}} e_0 \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_1 \Rightarrow v_k, \sigma_1 \quad \text{pop}(\sigma_1, a, v_k) = (v, \sigma_2)}{\sigma \vdash \text{pop } e_0 e_1 \Rightarrow v, \sigma_2}$$

$$\frac{\sigma \vdash_{\text{escape}} e \Rightarrow a_0, \sigma_0 \quad \text{copyObj}(\sigma_0, a_0) = (a_1, \sigma_1)}{\sigma \vdash \text{copy-obj } e \Rightarrow a_1, \sigma_1} \quad \frac{\sigma \vdash_{\text{escape}} e \Rightarrow a_0, \sigma_0 \quad \text{keys}(\sigma_0, a_0) = (a_1, \sigma_1)}{\sigma \vdash \text{map-keys } e \Rightarrow a_1, \sigma_1}$$

$$\frac{\sigma \vdash_{\text{escape}} e_0 \Rightarrow a_l, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_1 \Rightarrow v, \sigma_1 \quad \text{contains}(\sigma_1, a_l, v) = b}{\sigma \vdash \text{contains } e_0 e_1 \Rightarrow b, \sigma_1}$$

$$\frac{\sigma \vdash_{\text{escape}} e \Rightarrow s, \sigma_0 \quad \text{convert}(\triangleright, s, e^*) = (v, \sigma_1)}{\sigma \vdash \text{convert } e \triangleright e^* \Rightarrow v, \sigma_1}$$

$$\begin{array}{c}
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow s_0, \sigma_0 \quad \text{equals}(s_0, s) = b}{\sigma \vdash \text{is-instance-of } e \ s \Rightarrow b, \sigma_0} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow ASTVal, \sigma_0 \quad \text{isKindOf}(ASTVal, s) = b}{\sigma \vdash \text{is-instance-of } e \ s \Rightarrow b, \sigma_0} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow ASTVal, \sigma_0 \quad \text{toString}(ASTVal) = s}{\sigma \vdash \text{get-syntax } e \Rightarrow s, \sigma_0} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow ASTVal, \sigma_0 \quad \text{getElems}(ASTVal) = v^* \quad \text{allocList}(\sigma_0, v^*) = (a, \sigma_1)}{\sigma \vdash \text{get-elems } e \ s \Rightarrow a, \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e_c \Rightarrow ASTVal, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e_r \Rightarrow s, \sigma_1 \quad \text{assertValidParseRule}(s) \quad \text{getNewValue}(ASTVal, s) = v}{\sigma \vdash \text{parse-syntax } e_c \ e_r \ e^* \Rightarrow v, \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e_c \Rightarrow s_c, \sigma_c \quad \sigma_c \vdash_{\text{escape}} e_r \Rightarrow s_r, \sigma_r \quad \sigma_r \vdash e_0 \Rightarrow b_0, \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow b_n, \sigma_n \quad \text{getNewValue}(s_c, s_r, b^*) = v}{\sigma \vdash \text{parse-syntax } e_c \ e_r \ e_0 \ \cdots \ e_n \Rightarrow v, \sigma_n}
\end{array}$$

TODO

define helper function OR change function to rule explicitly

### 2.2.3 Reference

$$\boxed{\sigma \vdash r \Rightarrow rv, \sigma} \quad \sigma \vdash x \Rightarrow x, \sigma$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow s, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2}{\sigma \vdash r[e] \Rightarrow s.v, \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \notin \text{Dom}(H)}{\sigma \vdash r[e] \Rightarrow a[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = \text{symbol } v_s}{\sigma \vdash r[e] \Rightarrow a[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle}{\sigma \vdash r[e] \Rightarrow a[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertNotEquals}(t, t_{\text{completion}})}{\sigma \vdash r[e] \Rightarrow a[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \quad v \in \{v_k\}}{\sigma \vdash r[e] \Rightarrow a[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \quad v \notin \{v_k\} \quad \text{"Value"} \in \{v_k\} \quad [v_k \mapsto v_v]^*(\text{"Value"}) = a_0}{\sigma \vdash r[e] \Rightarrow a_0[v], \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a, \sigma_1 \quad \sigma_1 \vdash_{\text{escape}} e \Rightarrow v, \sigma_2 \quad \sigma_2 = (C, C^*, E, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad \text{assertEquals}(t, t_{\text{completion}}) \quad v \notin \{v_k\} \quad \text{"Value"} \in \{v_k\} \quad [v_k \mapsto v_v]^*(\text{"Value"}) = s}{\sigma \vdash r[e] \Rightarrow s.v, \sigma_2}$$

## 2.2.4 Reference Value

$$\boxed{\sigma \vdash rv \Rightarrow v, \sigma} \quad \frac{\sigma = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad x \in \text{Dom}(E_L) \quad E_L(x) = v}{\sigma \vdash x \Rightarrow v, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad x \notin \text{Dom}(E_L) \quad x \in \text{Dom}(E_G) \quad E_G(x) = v}{\sigma \vdash x \Rightarrow v, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad x \notin \text{Dom}(E_L) \quad x \notin \text{Dom}(E_G)}{\sigma \vdash x \Rightarrow \text{absent}, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad \text{assertEquals}(v, \text{"Description"}) \quad H(a) = \text{symbol } v_d}{\sigma \vdash a[v] \Rightarrow v_d, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{[v_k \mapsto v_v]^*\} \quad v \notin \{v_k\}}{\sigma \vdash a[v] \Rightarrow \text{absent}, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{[v_k \mapsto v_v]^*\} \quad v \in \{v_k\} \quad [v_k \mapsto v_v]^*(v) = v_0}{\sigma \vdash a[v] \Rightarrow v_0, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad \text{assertEquals}(v, \text{"length"}) \quad H(a) = \langle v^* \rangle \quad \text{length}(\langle v^* \rangle) = d}{\sigma \vdash a[v] \Rightarrow d, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle \quad \text{assertInvalidIndex}(\langle v^* \rangle, d)}{\sigma \vdash a[d] \Rightarrow \text{absent}, \sigma}$$

$$\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle \quad \text{assertValidIndex}(\langle v^* \rangle, d) \quad (\langle v^* \rangle)(d) = v_0}{\sigma \vdash a[d] \Rightarrow v_0, \sigma}$$

$$\frac{\text{assertEquals}(s_k, \text{"length"}) \quad \text{length}(s) = d}{\sigma \vdash s.s_k \Rightarrow d, \sigma} \quad \frac{\text{charAt}(s, d) = s_0}{\sigma \vdash s.d \Rightarrow s_0, \sigma} \quad \frac{\text{charAt}(s, \text{toInt}(n)) = s_0}{\sigma \vdash s.n \Rightarrow s_0, \sigma}$$

### 2.2.5 Helper Function

$$\begin{array}{c}
\frac{\sigma = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad E_0 = E_L + (x \mapsto v) \quad C_0 = (x_{ret}, s, i^*, E_0) \quad \sigma_0 = (C_0, C^*, E_G, H)}{\mathbf{define}(\sigma, x, v) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad \mathbf{assertGlobalId}(x) \quad E_0 = E_G + (x \mapsto v) \quad \sigma_0 = (C, C^*, E_0, H)}{\mathbf{updated}(\sigma, x, v) = \sigma_0} \\
\\
\frac{\mathbf{assertLocalId}(x) \quad \mathbf{define}(\sigma, x, v) = \sigma_0}{\mathbf{updated}(\sigma, x, v) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad o = t \{ [v_k \mapsto v_v]^* + (v_0 \mapsto v_1) \} \quad H_0 = H + (a \mapsto o) \quad \sigma_0 = (C, C^*, E_G, H_0)}{\mathbf{updated}(\sigma, a[v_0], v_1) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad E_0 = E_L - x \quad C_0 = (x_{ret}, s, i^*, E_0) \quad \sigma_0 = (C_0, C^*, E_G, H)}{\mathbf{deleted}(\sigma, x) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = t \{ [v_k \mapsto v_v]^* \} \quad o = t \{ [v_k \mapsto v_v]^* - v_0 \} \quad H_0 = H + (a \mapsto o) \quad \sigma_0 = (C, C^*, E_G, H_0)}{\mathbf{deleted}(\sigma, a[v_0]) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle \quad o = \langle v^* \rangle :+ v_0 \quad H_0 = H + (a \mapsto o) \quad \sigma_0 = (C, C^*, E_G, H_0)}{\mathbf{append}(\sigma, a, v_0) = \sigma_0} \\
\\
\frac{\sigma = (C, C^*, E_G, H) \quad a \in \text{Dom}(H) \quad H(a) = \langle v^* \rangle \quad o = v_0 +: \langle v^* \rangle \quad H_0 = H + (a \mapsto o) \quad \sigma_0 = (C, C^*, E_G, H_0)}{\mathbf{prepend}(\sigma, a, v_0) = \sigma_0} \\
\\
\frac{\sigma = (C, \text{Nil}, E_G, H) \quad C = (x_{ret}, s, i^*, E_L) \quad E_0 = E_L + (x_{ret} \mapsto v) \quad C_0 = (x_{ret}, s, \text{Nil}, E_0) \quad \sigma_0 = (C_0, C^*, E_G, H)}{\mathbf{return}(\sigma, v) = \sigma_0} \\
\\
\frac{\sigma = (C, C_0 +: C^*, E_G, H) \quad C_0 = (x_{ret}, s, i^*, E_L) \quad E_0 = E_L + (x_{ret} \mapsto v) \quad C_1 = (x_{ret}, s, i^*, E_0) \quad \sigma_0 = (C_1, C^*, E_G, H)}{\mathbf{return}(\sigma, v) = \sigma_0}
\end{array}$$