

1 Abstract Syntax

This section explains abstract syntax of IR_{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)
	let $x = e$	(let)
	$r := e$	(assign)
	delete r	(delete)
	append $e \leftarrow e$	(append)
	prepend $e \rightarrow e$	(prepend)
	return e	(return)
	if e i i	(if-then-else)
	while e i	(while)
	$\{i^*\}$	(sequence)
	assert e	(assert)
	print e	(print)
	app $x = (e \ e^*)$	(function application)
	access $x = (e \ e)$	(access)
	withcont $x \ (x^*) = i$	(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)
	undefined	(undefined)
	null	(null)
	absent	(absent)
	new e	(symbol)
	new $\langle e^* \rangle$	(list)
	new $t \{ [e \mapsto e]^* \}$	(map)
	pop $e \ e$	(pop)
	typeof e	(typeof)
	is-instance-of $e \ s$	(is-instance-of)
	get-elems $e \ s$	(get-elements)
	get-syntax e	(get-syntax)
	parse-syntax $e \ e \ e^*$	(parse-syntax)
	convert $e \triangleright e^*$	(convert)
	contains $e \ e$	(contains)
	copy-obj e	(copy-object)
	map-keys e	(map-keys)
	!!! 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 IR_{ES} .

2.1 Domain

Semantic domain of IR_{ES} .

State	σ	\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	Value
Address	a	\in	Address
Object	o	\in	Object

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

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

Object	o	$::=$	<code>symbol</code> v	(symbol)
			$ $ $t \{ [v \mapsto v]^* \}$	(map)
			$ $ $\langle v^* \rangle$	(list)
			$ $ <code>not-supported</code> s	(not supported)

Value	v	$::=$	a	(address)
			$ $ c	(constant)
			$ $ $\lambda(s, x^*, x, i)$	(function)
			$ $ $\kappa(C, C^*, x^*, i)$	(continuation)
			$ $ $ASTVal$	(AST value)
			$ $ $ASTMethod \lambda(s, x^*, x, i) E$	(AST method)

RefValue	rv	$::=$	x	(identifier)
			$ $ $a[v]$	(reference to value of map in heap)
			$ $ $s.v$	(reference to string field)

TODO

ASTValue notation

change ASTMethod notation

2.2 Semantic of IR_{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 s.v, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow v_0, \sigma_1 \quad \text{updated}(\sigma_1, s.v, v_0) = \sigma_2}{\sigma \vdash r := e \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash r \Rightarrow rv, \sigma_0 \quad \text{deleted}(\sigma_0, rv) = \sigma_1}{\sigma \vdash \text{delete } r \Rightarrow \sigma_1} \\
\\
\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 \text{append}(\sigma_0^{??}, v_1, v_0) = \sigma_2}{\sigma \vdash \text{append } e_0 \rightarrow e_1 \Rightarrow \sigma_2} \\
\\
\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 \text{prepend}(\sigma_0^{??}, v_1, v_0) = \sigma_2}{\sigma \vdash \text{prepend } e_0 \rightarrow e_1 \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash e \Rightarrow v, \sigma_0 \quad \text{updateCtxStack}(\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 \text{updateCtx}(\sigma_0, i_0) = \sigma_1}{\sigma \vdash \text{if } e \ i_0 \ i_1 \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{false}, \sigma_0 \quad \text{updateCtx}(\sigma_0, i_1) = \sigma_1}{\sigma \vdash \text{if } e \ i_0 \ i_1 \Rightarrow \sigma_1} \\
\\
\frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{true}, \sigma_0 \quad \text{updateCtx}(\sigma_0, i) = \sigma_1}{\sigma \vdash \text{while } e \ i \Rightarrow \sigma_1} \quad \frac{\sigma \vdash_{\text{escape}} e \Rightarrow \text{false}, \sigma_0}{\sigma \vdash \text{while } e \ i \Rightarrow \sigma_0} \\
\\
\frac{\text{updateCtx2}(\sigma, i^*) = \sigma_0}{\sigma \vdash \{i^*\} \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 \quad \text{print}(v)}{\sigma \vdash \text{print } e \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}
\frac{\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \lambda(s, x^*, x_{var}, i) \quad \sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \dots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \quad \text{updateCtxRetId}(\sigma_n, x) = \sigma_\alpha \quad \text{createCtx}(s, i, x^*, v^*) = C_\alpha \quad \text{pushCtx}(\sigma_\alpha, C_\alpha) = \sigma_{next}}{\sigma \vdash \text{app } x = (e_f e_0 \dots e_n) \Rightarrow \sigma_{next}} \\
\\
\frac{\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \text{ASTMethod } \lambda(s, x^*, x_{var}, i) E \quad \sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \dots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \quad \text{updateCtxRetId}(\sigma_n, x) = \sigma_\alpha \quad \text{createCtx2}(s, i, x^*, v^*, E) = C_\alpha \quad \text{pushCtx}(\sigma_\alpha, C_\alpha) = \sigma_{next}}{\sigma \vdash \text{app } x = (e_f e_0 \dots e_n) \Rightarrow \sigma_{next}} \\
\\
\frac{\sigma \vdash e_f \Rightarrow v_f, \sigma_f \quad v_f = \kappa(C, C^*, x^*, i) \quad \sigma_f \vdash e_0 \Rightarrow v_0, \sigma_0 \quad \dots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n, \sigma_n \quad \text{setCtxInst}(C, i) = C_0 \quad \text{updateCtxEnv}(C_0, x^*, v^*) = C_1 \quad \text{updateState}(\sigma_n, C_1, C^*) = \sigma_{next}}{\sigma \vdash \text{app } x = (e_f e_0 \dots e_n) \Rightarrow \sigma_{next}} \\
\\
\frac{\sigma \vdash e_b \Rightarrow a, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow s_p, \sigma_1 \quad s_p = \text{"length"} \quad \text{define}(\sigma_1, x, \text{length}(s)) = \sigma_2}{\sigma \vdash \text{access } x = (e_b e) \Rightarrow \sigma_2} \\
\\
\frac{\sigma \vdash e_b \Rightarrow s, \sigma_0 \quad \sigma_0 \vdash_{\text{escape}} e \Rightarrow s_p, \sigma_1 \quad s_p = \text{"length"} \quad \text{define}(\sigma_1, x, \text{length}(s)) = \sigma_2}{\sigma \vdash \text{access } x = (e_b e) \Rightarrow \sigma_2} \\
\\
\frac{\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{ToString}(d)) = \sigma_2}{\sigma \vdash \text{access } x = (e_b e) \Rightarrow \sigma_2}
\end{array}$$

TODO

access - addr

access - ASTVal - Lexical

access - ASTVal

define helper function OR change function to rule explicitly

2.2.2 Expression

$$\begin{array}{c}
\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 = (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_0}} \quad \sigma_{0_{k_0}} \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_n}} \quad \sigma_{n_{k_n}} \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}
\end{array}$$

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

TODO

reference + handle escapeCompletion

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

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

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

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

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

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

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

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

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

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

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