1 Abstract Syntax

This section explains abstract syntax of IR_{ES}.

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

$$
\begin{array}{rrcll}
\text{Program} & p & ::= & i;\ \cdots;\ i & \\
\\
\text{Instruction} & i & ::= & e & \text{(expression)} \\
 & & | & \texttt{let}\ x\ \texttt{=}\ e & \text{(let)} \\
 & & | & r\texttt{:=}e & \text{(assign)} \\
 & & | & \texttt{delete}\ r & \text{(delete)} \\
 & & | & \texttt{append}\ e\ \leftarrow\ e & \text{(append)} \\
 & & | & \texttt{prepend}\ e\ \rightarrow\ e & \text{(prepend)} \\
 & & | & \texttt{return}\ e & \text{(return)} \\
 & & | & \texttt{if}\ e\ i\ i & \text{(if-then-else)} \\
 & & | & \texttt{while}\ e\ i & \text{(while)} \\
 & & | & \{i^*\} & \text{(sequence)} \\
 & & | & \texttt{assert}\ e & \text{(assert)} \\
 & & | & \texttt{print}\ e & \text{(print)} \\
 & & | & \texttt{app}\ x\ \texttt{=}\ (e\ e^*) & \text{(function application)} \\
 & & | & \texttt{access}\ x\ \texttt{=}\ (e\ e) & \text{(access)} \\
 & & | & \texttt{withcont}\ x\ (x^*)\ =\ i & \text{(continuation)} \\
\\
\text{Reference} & r & ::= & x & \text{(identifier)} \\
 & & | & r\texttt{[}e\texttt{]} & \text{(reference to value of field in heap)}
\end{array}
$$

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

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

2 Operational Semantic

This section explains operational semantic of IR_{ES}.

2.1 Domain

Semantic domain of IR_{ES}.

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

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

$$
\begin{array}{rrcll}
\text{Constant} & c & ::= & n \mid d \mid s \mid b \mid \texttt{undefined} \mid \texttt{null} \mid \texttt{absent} & \\
\\
\text{Object} & o & ::= & \texttt{symbol}\ v & \text{(symbol)} \\
& & \mid & t\ \{[v\ \mapsto\ v]^*\} & \text{(map)} \\
& & \mid & \texttt{<}v^*\texttt{>} & \text{(list)} \\
& & \mid & \texttt{not-supported}\ s & \text{(not supported)} \\
\\
\text{Value} & v & ::= & a & \text{(address)} \\
& & \mid & c & \text{(constant)} \\
& & \mid & \lambda\texttt{(}s,\ x^*,\ x,\ i\texttt{)} & \text{(function)} \\
& & \mid & \kappa\texttt{(}C,\ C^*,\ x^*,\ i\texttt{)} & \text{(continuation)} \\
& & \mid & ASTVal & \text{(AST value)} \\
& & \mid & \texttt{ASTMethod}\ \lambda\texttt{(}s,\ x^*,\ x,\ i\texttt{)}\ E & \text{(AST method)} \\
\\
\text{RefValue} & rv & ::= & x & \text{(identifier)} \\
& & \mid & a\texttt{[}v\texttt{]} & \text{(reference to value of map in heap)} \\
& & \mid & s\texttt{.}v & \text{(reference to string field)}
\end{array}
$$

TODO
ASTValue notation
change ASTMethod notation

2.2 Semantic of $\mathbf{IR_{ES}}$

- program : [[description of program execution]]
- instruction : $\sigma \vdash i \Rightarrow \sigma$
- expression : $\sigma \vdash e \Rightarrow v,\ \sigma$
- expression - escape completion : $\sigma \vdash_{escape} e \Rightarrow v,\ \sigma$
- 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

$$\boxed{\sigma \vdash i \Rightarrow \sigma} \qquad \frac{\sigma \vdash e \Rightarrow v,\ \sigma_0}{\sigma \vdash e \Rightarrow \sigma_0} \qquad \frac{\sigma \vdash e \Rightarrow v,\ \sigma_0 \quad \texttt{define}(\sigma_0,\ x,\ v) = \sigma_1}{\sigma \vdash \texttt{let}\ x\ \texttt{=}\ e \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash r \Rightarrow x,\ \sigma_0 \quad \sigma_0 \vdash e \Rightarrow v,\ \sigma_1 \quad \texttt{updated}(\sigma_1,\ x,\ v) = \sigma_2}{\sigma \vdash r\ \texttt{:=}\ e \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow a\texttt{[}v\texttt{]},\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow v_0,\ \sigma_1 \quad \texttt{updated}(\sigma_1,\ a\texttt{[}v\texttt{]},\ v_0) = \sigma_2}{\sigma \vdash r\ \texttt{:=}\ e \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow s\texttt{.}v,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow v_0,\ \sigma_1 \quad \texttt{updated}(\sigma_1,\ s\texttt{.}v,\ v_0) = \sigma_2}{\sigma \vdash r\ \texttt{:=}\ e \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash r \Rightarrow rv,\ \sigma_0 \quad \texttt{deleted}(\sigma_0,\ rv) = \sigma_1}{\sigma \vdash \texttt{delete}\ r \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e_0 \Rightarrow v_0,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e_1 \Rightarrow v_1,\ \sigma_1 \quad \texttt{append}(\sigma_0??,\ v_1,\ v_0) = \sigma_2}{\sigma \vdash \texttt{append}\ e_0 \rightarrow e_1 \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash_{escape} e_0 \Rightarrow v_0,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e_1 \Rightarrow v_1,\ \sigma_1 \quad \texttt{prepend}(\sigma_0??,\ v_1,\ v_0) = \sigma_2}{\sigma \vdash \texttt{prepend}\ e_0 \rightarrow e_1 \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash e \Rightarrow v,\ \sigma_0 \quad \texttt{updateCtxStack}(\sigma_0,\ v) = \sigma_1}{\sigma \vdash \texttt{return}\ e \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow \texttt{true},\ \sigma_0 \quad \texttt{updateCtx}(\sigma_0,\ i_0) = \sigma_1}{\sigma \vdash \texttt{if}\ e\ i_0\ i_1 \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow \texttt{false},\ \sigma_0 \quad \texttt{updateCtx}(\sigma_0,\ i_1) = \sigma_1}{\sigma \vdash \texttt{if}\ e\ i_0\ i_1 \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow \texttt{true},\ \sigma_0 \quad \texttt{updateCtx}(\sigma_0,\ i) = \sigma_1}{\sigma \vdash \texttt{while}\ e\ i \Rightarrow \sigma_1} \qquad \frac{\sigma \vdash_{escape} e \Rightarrow \texttt{false},\ \sigma_0}{\sigma \vdash \texttt{while}\ e\ i \Rightarrow \sigma_0}$$

$$\frac{\texttt{updateCtx2}(\sigma,\ i^*) = \sigma_0}{\sigma \vdash \{i^*\} \Rightarrow \sigma_0} \qquad \frac{\sigma \vdash e \Rightarrow \texttt{true},\ \sigma_0}{\sigma \vdash \texttt{assert}\ e \Rightarrow \sigma_0} \qquad \frac{\sigma \vdash e \Rightarrow v,\ \sigma_0 \quad \texttt{print}(v)}{\sigma \vdash \texttt{print}\ e \Rightarrow \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E,\ H) \quad \texttt{define}(\sigma,\ x_{id},\ \kappa(C,\ C^*,\ x^*,\ i)) = \sigma_0}{\sigma \vdash \texttt{withcont}\ x_{id}\ \texttt{(}x^*\texttt{)}\ \texttt{=}\ i \Rightarrow \sigma_0}$$

$$\frac{\sigma \vdash e_f \Rightarrow v_f,\ \sigma_f \qquad v_f = \lambda(s,\ x^*,\ x_{var},\ i) \\ \sigma_f \vdash e_0 \Rightarrow v_0,\ \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n,\ \sigma_n \\ \texttt{updateCtxRetId}(\sigma_n,\ x) = \sigma_\alpha \quad \texttt{createCtx}(s,\ i,\ x^*,\ v^*) = C_\alpha \quad \texttt{pushCtx}(\sigma_\alpha,\ C_\alpha) = \sigma_{next}}{\sigma \vdash \texttt{app}\ x \texttt{ = } (e_f\ e_0\ \cdots\ e_n) \Rightarrow \sigma_{next}}$$

$$\frac{\sigma \vdash e_f \Rightarrow v_f,\ \sigma_f \qquad v_f = \texttt{ASTMethod}\ \lambda(s,\ x^*,\ x_{var},\ i)\ E \\ \sigma_f \vdash e_0 \Rightarrow v_0,\ \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n,\ \sigma_n \\ \texttt{updateCtxRetId}(\sigma_n,\ x) = \sigma_\alpha \quad \texttt{createCtx2}(s,\ i,\ x^*,\ v^*,\ E) = C_\alpha \quad \texttt{pushCtx}(\sigma_\alpha,\ C_\alpha) = \sigma_{next}}{\sigma \vdash \texttt{app}\ x \texttt{ = } (e_f\ e_0\ \cdots\ e_n) \Rightarrow \sigma_{next}}$$

$$\frac{\sigma \vdash e_f \Rightarrow v_f,\ \sigma_f \qquad v_f = \kappa(C,\ C^*,\ x^*,\ i) \\ \sigma_f \vdash e_0 \Rightarrow v_0,\ \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow v_n,\ \sigma_n \\ \texttt{setCtxInst}(C,\ i) = C_0 \quad \texttt{updateCtxEnv}(C_0,\ x^*,\ v^*) = C_1 \quad \texttt{updateState}(\sigma_n,\ C_1,\ C^*) = \sigma_{next}}{\sigma \vdash \texttt{app}\ x \texttt{ = } (e_f\ e_0\ \cdots\ e_n) \Rightarrow \sigma_{next}}$$

$$\frac{\sigma \vdash e_b \Rightarrow a,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow s_p,\ \sigma_1 \\ \texttt{assertEquals}(s_p,\ \texttt{"length"}) \quad \texttt{define}(\sigma_1,\ x,\ \texttt{length}(s)) = \sigma_2}{\sigma \vdash \texttt{access}\ x \texttt{ = } (e_b\ e) \Rightarrow \sigma_2}$$

$$\frac{\sigma \vdash e_b \Rightarrow s,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow d,\ \sigma_1 \quad \texttt{define}(\sigma_1,\ x,\ \texttt{charAt}(s,\ d)) = \sigma_2}{\sigma \vdash \texttt{access}\ x \texttt{ = } (e_b\ e) \Rightarrow \sigma_2}$$

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

$$\frac{\sigma \vdash e_b \Rightarrow a,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow v_p,\ \sigma_1 \quad \sigma_1 = (C,\ C^*,\ E,\ H) \\ a \in \texttt{Dom}(H) \quad H(a) = \texttt{<}v^*\texttt{>} \quad \texttt{getHeapProp}(H,\ a,\ v_p) = v_0 \\ \texttt{define}(\sigma_1,\ x,\ v_0) = \sigma_2}{\sigma \vdash \texttt{access}\ x \texttt{ = } (e_b\ e) \Rightarrow \sigma_2}$$

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

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

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

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

TODO
access - ASTVal - Lexical
access - ASTVal
define helper function OR change function to rule explictly

2.2.2 Expression

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

$$\sigma \vdash \texttt{undefined} \Rightarrow \texttt{undefined},\ \sigma \qquad \sigma \vdash \texttt{null} \Rightarrow \texttt{null},\ \sigma \qquad \sigma \vdash \texttt{absent} \Rightarrow \texttt{absent},\ \sigma$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\sigma \vdash_{escape} e \Rightarrow s_0,\ \sigma_0 \quad \texttt{equals}(s_0,\ s) = b}{\sigma \vdash \texttt{is-instance-of}\ e\ s \Rightarrow b,\ \sigma_0}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow ASTVal,\ \sigma_0 \quad \texttt{isKindOf}(ASTVal,\ s) = b}{\sigma \vdash \texttt{is-instance-of}\ e\ s \Rightarrow b,\ \sigma_0}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow ASTVal,\ \sigma_0 \quad \texttt{toString}(ASTVal) = s}{\sigma \vdash \texttt{get-syntax}\ e \Rightarrow s,\ \sigma_0}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow ASTVal,\ \sigma_0 \quad \texttt{getElems}(ASTVal) = v^* \quad \texttt{allocList}(\sigma_0,\ v^*) = (a,\ \sigma_1)}{\sigma \vdash \texttt{get-elems}\ e\ s \Rightarrow a,\ \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e_c \Rightarrow ASTVal,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e_r \Rightarrow s,\ \sigma_1 \quad \texttt{assertValidParseRule}(s) \\ \texttt{getNewValue}(ASTVal,\ s) = v\end{array}}{\sigma \vdash \texttt{parse-syntax}\ e_c\ e_r\ e^* \Rightarrow v,\ \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e_c \Rightarrow s_c,\ \sigma_c \quad \sigma_c \vdash_{escape} e_r \Rightarrow s_r,\ \sigma_r \\ \sigma_r \vdash e_0 \Rightarrow b_0,\ \sigma_0 \quad \cdots \quad \sigma_{n-1} \vdash e_n \Rightarrow b_n,\ \sigma_n \quad \texttt{getNewValue}(s_c,\ s_r,\ b^*) = v\end{array}}{\sigma \vdash \texttt{parse-syntax}\ e_c\ e_r\ e_0\ \cdots\ e_n \Rightarrow v,\ \sigma_n}$$

TODO
define helper function OR change function to rule explictly

2.2.3 Reference

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

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

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

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

$$\frac{\begin{array}{c}\sigma \vdash r \Rightarrow rv,\ \sigma_0 \quad \sigma_0 \vdash rv \Rightarrow a,\ \sigma_1 \quad \sigma_1 \vdash_{escape} e \Rightarrow v,\ \sigma_2 \quad \sigma_2 = (C,\ C^*,\ E,\ H) \\ a \in \texttt{Dom}(H) \quad H(a) = \texttt{<}v^*\texttt{>}\end{array}}{\sigma \vdash r\texttt{[}e\texttt{]} \Rightarrow a\texttt{[}v\texttt{]},\ \sigma_2}$$

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

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

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

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

2.2.4 Reference Value

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

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

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

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

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

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

$$\frac{\begin{array}{c}\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \mathtt{Dom}(H) \\ \mathtt{assertEquals}(v,\ \text{"}\mathtt{length}\text{"}) \quad H(a) = \mathtt{<}v^*\mathtt{>} \quad \mathtt{length}(\mathtt{<}v^*\mathtt{>}) = d\end{array}}{\sigma \vdash a\mathtt{[}v\mathtt{]} \Rightarrow d,\ \sigma}$$

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

$$\frac{\begin{array}{c}\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \mathtt{Dom}(H) \\ H(a) = \mathtt{<}v^*\mathtt{>} \quad \mathtt{assertValidIndex}(\mathtt{<}v^*\mathtt{>},\ d) \quad (\mathtt{<}v^*\mathtt{>})(d) = v_0\end{array}}{\sigma \vdash a\mathtt{[}d\mathtt{]} \Rightarrow v_0,\ \sigma}$$

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