1 Abstract Syntax

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

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

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

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

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

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 $\mathrm{IR}_{\mathbf{ES}}$

- 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

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

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

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

$$\frac{\sigma \vdash r \Rightarrow x,\ \sigma_0 \quad \texttt{deleted}(\sigma_0,\ x) = \sigma_1}{\sigma \vdash \texttt{delete}\ r \Rightarrow \sigma_1} \qquad \frac{\sigma \vdash r \Rightarrow a\texttt{[}v\texttt{]},\ \sigma_0 \quad \texttt{deleted}(\sigma_0,\ a\texttt{[}v\texttt{]}) = \sigma_1}{\sigma \vdash \texttt{delete}\ r \Rightarrow \sigma_1}$$

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

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

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

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow \texttt{true},\ \sigma_0 \quad \sigma_0 = (C,\ C^*,\ E_G,\ H) \quad C = (x,\ s,\ i^*,\ E_L) \\ C_0 = (x,\ s,\ \color{red}{i_0\ \texttt{+:}\ i^*},\ E_L) \quad \sigma_1 = (C_0,\ C^*,\ E_G,\ H)\end{array}}{\sigma \vdash \texttt{if}\ e\ i_0\ i_1 \Rightarrow \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow \texttt{false},\ \sigma_0 \quad \sigma_0 = (C,\ C^*,\ E_G,\ H) \quad C = (x,\ s,\ i^*,\ E_L) \\ C_0 = (x,\ s,\ \color{red}{i_1\ \texttt{+:}\ i^*},\ E_L) \quad \sigma_1 = (C_0,\ C^*,\ E_G,\ H)\end{array}}{\sigma \vdash \texttt{if}\ e\ i_0\ i_1 \Rightarrow \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow \texttt{true},\ \sigma_0 \quad \sigma_0 = (C,\ C^*,\ E_G,\ H) \quad C = (x,\ s,\ i^*,\ E_L) \\ C_0 = (x,\ s,\ \color{red}{i_b\ \texttt{+:}\ i^*},\ E_L) \quad \sigma_1 = (C_0,\ C^*,\ E_G,\ H)\end{array}}{\sigma \vdash \texttt{while}\ e\ i_b \Rightarrow \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e \Rightarrow \texttt{false},\ \sigma_0}{\sigma \vdash \texttt{while}\ e\ i_b \Rightarrow \sigma_0} \qquad \frac{\sigma \vdash e \Rightarrow \texttt{true},\ \sigma_0}{\sigma \vdash \texttt{assert}\ e \Rightarrow \sigma_0} \qquad \frac{\sigma \vdash e \Rightarrow v,\ \sigma_0}{\sigma \vdash \texttt{print}\ e \Rightarrow \sigma_0}$$

$$\frac{\begin{array}{c}\sigma = (C,\ C^*,\ E_G,\ H) \quad C = (x,\ s,\ i^*,\ E_L) \\ C_0 = (x,\ s,\ \color{red}{i_s^*\ \texttt{++}\ i^*},\ E_L) \quad \sigma_0 = (C_0,\ C^*,\ E_G,\ H)\end{array}}{\sigma \vdash \{i_s^*\} \Rightarrow \sigma_0}$$

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

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

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

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

$$\frac{\begin{array}{c} \sigma \vdash e_b \Rightarrow s,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e \Rightarrow v_p,\ \sigma_1 \\ \texttt{getStringProp}(s,\ v_p) = v_0 \quad \texttt{define}(\sigma_1,\ x,\ v_0) = \sigma_2 \end{array}}{\sigma \vdash \texttt{access}\ x\ \texttt{=}\ (e_b\ e) \Rightarrow \sigma_2}$$

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

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

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

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

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

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

TODO
access - ASTVal - Lexical
access - ASTVal

2.2.2 Expression

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

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

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

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

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

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

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow v,\ \sigma_0 \qquad \sigma_0 = (C,\ C^*,\ E,\ H) \\ \mathtt{allocSymbol}(H,\ v) = (a,\ H_0) \qquad \sigma_1 = (C,\ C^*,\ E,\ H_0)\end{array}}{\sigma \vdash \mathtt{new}\ e \Rightarrow a,\ \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash e_0 \Rightarrow v_0,\ \sigma_0 \qquad \cdots \qquad \sigma_{n-1} \vdash e_n \Rightarrow v_n,\ \sigma_n \qquad \sigma_n = (C,\ C^*,\ E,\ H) \\ \mathtt{allocList}(H,\ v^*) = (a,\ H_0) \qquad \sigma_{next} = (C,\ C^*,\ E,\ H_0)\end{array}}{\sigma \vdash \mathtt{new}\ \mathtt{[}e_0,\ \cdots,\ e_n\mathtt{]} \Rightarrow a,\ \sigma_{next}}$$

$$\frac{\begin{array}{c}\sigma = (C,\ C^*,\ E,\ H) \qquad \mathtt{allocMap}(H,\ t) = (a,\ H_0) \qquad \sigma_t = (C,\ C^*,\ E,\ H_0) \\ \sigma_t \vdash_{escape} e_{k_0} \Rightarrow v_{k_0},\ \sigma_{0_k} \qquad \sigma_{0_k} \vdash e_{v_0} \Rightarrow v_{v_0},\ \sigma_{0_v} \qquad \mathtt{updated}(\sigma_{0_v},\ a\mathtt{[}v_{k_0}\mathtt{]},\ v_{v_0}) = \sigma_0 \\ \cdots \\ \sigma_{n-1} \vdash_{escape} e_{k_n} \Rightarrow v_{k_n},\ \sigma_{n_k} \qquad \sigma_{n_k} \vdash e_{v_n} \Rightarrow v_{v_n},\ \sigma_{n_v} \qquad \mathtt{updated}(\sigma_{n_v},\ a\mathtt{[}v_{k_n}\mathtt{]},\ v_{v_n}) = \sigma_n\end{array}}{\sigma \vdash \mathtt{new}\ t\ \mathtt{(}e_{k_0}\ \mapsto\ e_{v_0},\ \cdots,\ e_{k_n}\ \mapsto\ e_{v_n}\mathtt{)} \Rightarrow a,\ \sigma_n}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e_0 \Rightarrow a,\ \sigma_0 \qquad \sigma_0 \vdash_{escape} e_1 \Rightarrow d,\ \sigma_1 \qquad \sigma_1 = (C,\ C^*,\ E,\ H) \qquad a \in \mathtt{Dom}(H) \\ H(a) = \mathtt{<}v^*\mathtt{>} \qquad d_l = |\mathtt{<}v^*\mathtt{>}| \qquad 0 \le d < d_l \qquad \mathtt{<}v^*\mathtt{>}(d) = v_0 \qquad o = \mathtt{<}v^*\mathtt{>}[0..d]\ \mathtt{++}\ \mathtt{<}v^*\mathtt{>}[(d+1)..d_l] \\ H_0 = H + (a \mapsto o) \qquad \sigma_2 = (C,\ C^*,\ E,\ H_0)\end{array}}{\sigma \vdash \mathtt{pop}\ e_0\ e_1 \Rightarrow v_0,\ \sigma_2}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow a,\ \sigma_0 \qquad \sigma_0 = (C,\ C^*,\ E,\ H) \qquad a \in \mathtt{Dom}(H) \qquad a_0 \notin \mathtt{Dom}(H) \\ H(a) = o \qquad H_0 = H + (a_0 \mapsto o) \qquad \sigma_1 = (C,\ C^*,\ E,\ H_0)\end{array}}{\sigma \vdash \mathtt{copy\text{-}obj}\ e \Rightarrow a_0,\ \sigma_1}$$

$$\frac{\begin{array}{c}\sigma \vdash_{escape} e \Rightarrow a,\ \sigma_0 \qquad \sigma_0 = (C,\ C^*,\ E,\ H) \qquad a \in \mathtt{Dom}(H) \qquad H(a) = t\ \{[v_k \mapsto v_v]^*\} \\ a_0 \notin \mathtt{Dom}(H) \qquad o = \mathtt{<}v_k^*\mathtt{>} \qquad H_0 = H + (a_0 \mapsto o) \qquad \sigma_1 = (C,\ C^*,\ E,\ H_0)\end{array}}{\sigma \vdash \mathtt{map\text{-}keys}\ e \Rightarrow a_1,\ \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e_0 \Rightarrow a,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e_1 \Rightarrow v_0,\ \sigma_1 \quad \sigma_1 = (C,\ C^*,\ E,\ H) \\ a \in H \quad H(a) = \texttt{<}v^*\texttt{>} \quad v_0 \in \{v\}}{\sigma \vdash \texttt{contains}\ e_0\ e_1 \Rightarrow \texttt{true},\ \sigma_1}$$

$$\frac{\sigma \vdash_{escape} e_0 \Rightarrow a,\ \sigma_0 \quad \sigma_0 \vdash_{escape} e_1 \Rightarrow v_0,\ \sigma_1 \quad \sigma_1 = (C,\ C^*,\ E,\ H) \\ a \in H \quad H(a) = \texttt{<}v^*\texttt{>} \quad v_0 \notin \{v\}}{\sigma \vdash \texttt{contains}\ e_0\ e_1 \Rightarrow \texttt{false},\ \sigma_1}$$

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

$$\frac{\sigma \vdash_{escape} e \Rightarrow s_0,\ \sigma_0 \quad s_0 = s}{\sigma \vdash \texttt{is-instance-of}\ e\ s \Rightarrow \texttt{true},\ \sigma_0} \qquad \frac{\sigma \vdash_{escape} e \Rightarrow s_0,\ \sigma_0 \quad s_0 \neq s}{\sigma \vdash \texttt{is-instance-of}\ e\ s \Rightarrow \texttt{false},\ \sigma_0}$$

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

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

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

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

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

TODO
fix rule related to AST value

- is-instance-of AST
- get-syntax
- get-elems
- parse-syntax

2.2.3 Reference

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

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

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

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

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

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

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

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

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

2.2.4 Reference Value

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

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

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

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \texttt{Dom}(H) \quad \texttt{getHeapProp}(H,\ a,\ v) = v_0}{\sigma \vdash a\texttt{[}v\texttt{]} \Rightarrow v_0,\ \sigma}$$

$$\frac{\texttt{getStringProp}(s,\ v) = v_0}{\sigma \vdash s\texttt{.}v \Rightarrow v_0,\ \sigma}$$

2.2.5 Helper Function

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad C = (x_{ret},\ s,\ i^*,\ E_L) \\ E_0 = E_L + (x \mapsto v) \quad C_0 = (x_{ret},\ s,\ i^*,\ E_0) \quad \sigma_0 = (C_0,\ C^*,\ E_G,\ H)}{\texttt{define}(\sigma,\ x,\ v) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad \texttt{assertGlobalId}(x) \quad E_0 = E_G + (x \mapsto v) \quad \sigma_0 = (C,\ C^*,\ E_0,\ H)}{\texttt{updated}(\sigma,\ x,\ v) = \sigma_0}$$

$$\frac{\texttt{assertLocalId}(x) \quad \texttt{define}(\sigma,\ x,\ v) = \sigma_0}{\texttt{updated}(\sigma,\ x,\ v) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \texttt{Dom}(H) \quad H(a) = t\ \{[v_k\ \mapsto\ v_v]^*\} \quad o = t\ \{[v_k\ \mapsto\ v_v]^* + (v_0 \mapsto v_1)\} \\ H_0 = H + (a \mapsto o) \quad \sigma_0 = (C,\ C^*,\ E_G,\ H_0)}{\texttt{updated}(\sigma,\ a\texttt{[}v_0\texttt{]},\ v_1) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad C = (x_{ret},\ s,\ i^*,\ E_L) \\ E_0 = E_L - x \quad C_0 = (x_{ret},\ s,\ i^*,\ E_0) \quad \sigma_0 = (C_0,\ C^*,\ E_G,\ H)}{\texttt{deleted}(\sigma,\ x) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \texttt{Dom}(H) \quad H(a) = t\ \{[v_k\ \mapsto\ v_v]^*\} \quad o = t\ \{[v_k\ \mapsto\ v_v]^* - v_0\} \\ H_0 = H + (a \mapsto o) \quad \sigma_0 = (C,\ C^*,\ E_G,\ H_0)}{\texttt{deleted}(\sigma,\ a\texttt{[}v_0\texttt{]}) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \texttt{Dom}(H) \quad H(a) = \texttt{<}v^*\texttt{>} \quad o = \texttt{<}v^*\texttt{>}\ \texttt{:+}\ v_0 \\ H_0 = H + (a \mapsto o) \quad \sigma_0 = (C,\ C^*,\ E_G,\ H_0)}{\texttt{append}(\sigma,\ a,\ v_0) = \sigma_0}$$

$$\frac{\sigma = (C,\ C^*,\ E_G,\ H) \quad a \in \texttt{Dom}(H) \quad H(a) = \texttt{<}v^*\texttt{>} \quad o = v_0\ \texttt{+:}\ \texttt{<}v^*\texttt{>} \\ H_0 = H + (a \mapsto o) \quad \sigma_0 = (C,\ C^*,\ E_G,\ H_0)}{\texttt{prepend}(\sigma,\ a,\ v_0) = \sigma_0}$$

$$\frac{\sigma = (C,\ \texttt{Nil},\ E_G,\ H) \quad C = (x_{ret},\ s,\ i^*,\ E_L) \\ E_0 = E_L + (x_{ret} \mapsto v) \quad C_0 = (x_{ret},\ s,\ \texttt{Nil},\ E_0) \quad \sigma_0 = (C_0,\ C^*,\ E_G,\ H)}{\texttt{return}(\sigma,\ v) = \sigma_0}$$

$$\frac{\sigma = (C,\ C_0\ \texttt{+:}\ C^*,\ E_G,\ H) \quad C_0 = (x_{ret},\ s,\ i^*,\ E_L) \\ E_0 = E_L + (x_{ret} \mapsto v) \quad C_1 = (x_{ret},\ s,\ i^*,\ E_0) \quad \sigma_0 = (C_1,\ C^*,\ E_G,\ H)}{\texttt{return}(\sigma,\ v) = \sigma_0}$$

$$\frac{a \in \texttt{Dom}(H) \qquad s = \text{"}\texttt{Description}\text{"} \qquad H(a) = \texttt{symbol}\ v}{\texttt{getHeapProp}(H,\ a,\ s) = v}$$

$$\frac{a \in \texttt{Dom}(H) \qquad H(a) = t\ \{[v_k \mapsto v_v]^*\} \qquad v \notin \{v_k\}}{\texttt{getHeapProp}(H,\ a,\ v) = \texttt{absent}}$$

$$\frac{a \in \texttt{Dom}(H) \qquad H(a) = t\ \{[v_k \mapsto v_v]^*\} \qquad v_p \in \{v_k\} \qquad [v_k\ \mapsto\ v_v]^*(v_p) = v}{\texttt{getHeapProp}(H,\ a,\ v_p) = v}$$

$$\frac{a \in \texttt{Dom}(H) \qquad s = \text{"}\texttt{length}\text{"} \qquad H(a) = \texttt{<}v^*\texttt{>} \qquad |\texttt{<}v^*\texttt{>}| = d}{\texttt{getHeapProp}(H,\ a,\ s) = d}$$

$$\frac{a \in \texttt{Dom}(H) \qquad H(a) = \texttt{<}v^*\texttt{>} \qquad d < 0}{\texttt{getHeapProp}(H,\ a,\ d) = \texttt{absent}} \qquad \frac{a \in \texttt{Dom}(H) \qquad H(a) = \texttt{<}v^*\texttt{>} \qquad |\texttt{<}v^*\texttt{>}| = d_0 \qquad d \geq d_0}{\texttt{getHeapProp}(H,\ a,\ d) = \texttt{absent}}$$

$$\frac{a \in \texttt{Dom}(H) \qquad H(a) = \texttt{<}v^*\texttt{>} \qquad |\texttt{<}v^*\texttt{>}| = d_0 \qquad 0 \leq d < d_0 \qquad \texttt{<}v^*\texttt{>}(d) = v_0}{\texttt{getHeapProp}(H,\ a,\ d) = v_0}$$

$$\frac{s_p = \texttt{"length"} \qquad v = |s|}{\texttt{getStringProp}(s,\ s_p) = v} \qquad \frac{\texttt{toInt}(n) = d \qquad \texttt{charAt}(s,\ d) = s_0}{\texttt{getStringProp}(s,\ n) = s_0} \qquad \frac{\texttt{charAt}(s,\ d) = s_0}{\texttt{getStringProp}(s,\ d) = s_0}$$

$$\frac{a \notin \texttt{Dom}(H) \qquad o = \texttt{symbol}\ s \qquad H_0 = H + (a \mapsto o)}{\texttt{allocSymbol}(H,\ s) = (a,\ H_0)}$$

$$\frac{a \notin \texttt{Dom}(H) \qquad o = \texttt{symbol undefined} \qquad H_0 = H + (a \mapsto o)}{\texttt{allocSymbol}(H,\ \texttt{undefined}) = (a,\ H_0)}$$

$$\frac{a \notin \texttt{Dom}(H) \qquad o = t\ \{\emptyset\} \qquad H_0 = H + (a \mapsto o)}{\texttt{allocMap}(H,\ t) = (a,\ H_0)}$$

$$\frac{a \notin \texttt{Dom}(H) \qquad o = \texttt{<}v^*\texttt{>} \qquad H_0 = H + (a \mapsto o)}{\texttt{allocList}(H,\ v^*) = (a,\ H_0)}$$

TODO
getType
getElmes
isKindOf
toString
getNewValue
assertValidParseRule