The SAFE Specification

Working Draft

Version 0.2

Sukyoung Ryu
Jaejun Choi
Woongsik Choi
Yoonseok Ko
Hongki Lee
Changhee Park

May 15, 2015

Contents

Chapter 1

Overview

The prevalent uses of JavaScript in web programming have revealed security vulnerability issues of JavaScript applications, which emphasizes the need for JavaScript analyzers to detect such issues. Recently, researchers have proposed several analyzers of JavaScript programs and some web service companies have developed various JavaScript engines. However, unfortunately, most of the tools are not documented well, thus it is very hard to understand and modify them. Or, such tools are often not open to the public.

In this specification, we present formal specification and implementation of SAFE, a scalable analysis framework for ECMAScript, developed for the JavaScript research community. This is the very first attempt to provide both formal specification and its open-source implementation for JavaScript, compared to the existing approaches focused on only one of them. To make it more amenable for other researchers to use our framework, we formally define three kinds of intermediate representations for JavaScript used in the framework, and we provide formal specifications of translations between them. To be adaptable for adventurous future research including modifications in the original JavaScript syntax, we actively use open-source tools to automatically generate parsers and some intermediate representations. To support a variety of program analyses in various compilation phases, we design the framework to be as flexible, scalable, and pluggable as possible. Finally, our framework is publicly available, and some collaborative research using the framework are in progress.

Chapter 2

AST

p	$::=$	$\mathit{fd}^*\ \mathit{vd}^*\ s^*$	`Program(TopLevel body)`
			`TopLevel(List<FunDecl> fds, List<VarDecl> vds, List<SourceElement> stmts)`

fd	$::=$	$\texttt{function}\ f\ \texttt{(}(x\texttt{,})^*\texttt{)}\ \texttt{\{}\mathit{fd}^*\ \mathit{vd}^*\ s^*\texttt{\}}$	`FunDecl(Id name, Functional ftn)`
			`Functional(List<FunDecl> fds, List<VarDecl> vds, List<SourceElement> stmts, List<Id> params)`

vd	$::=$	$\texttt{var}\ x$	`VarDecl(Id name, Option<Expr> expr)`
s	$::=$	$\texttt{\{}s^*\texttt{\}}$	`Block(List<Stmt> stmts, boolean internal = false)`
	$\mid$	$\texttt{var}\ \mathit{vd}(\texttt{,}\ \mathit{vd})^*\texttt{;}$	`VarStmt(List<VarDecl> vds)`
	$\mid$	$\texttt{;}$	`EmptyStmt()`
	$\mid$	$e\texttt{;}$	`ExprStmt(Expr expr)`
	$\mid$	$\texttt{if (}e\texttt{)}\ s\ (\texttt{else}\ s)^?$	`If(Expr cond, Stmt trueBranch, Option<Stmt> falseBranch)`
	$\mid$	$\texttt{switch (}e\texttt{)}\ \texttt{\{}\mathit{cc}^*\ (\texttt{default:}s^*)^?\ \mathit{cc}^*\texttt{\}}$	`Switch(Expr cond, List<Case> frontCases, Option<List<Stmt>> def, List<Case> backCases)`
	$\mid$	$\texttt{do}\ s\ \texttt{while (}e\texttt{);}$	`DoWhile(Stmt body, Expr cond)`
	$\mid$	$\texttt{while (}e\texttt{)}\ s$	`While(Expr cond, Stmt body)`
	$\mid$	$\texttt{for (}e^?\texttt{;}e^?\texttt{;}e^?\texttt{)}\ s$	`For(Option<Expr> init, Option<Expr> cond, Option<Expr> action, Stmt body)`
	$\mid$	$\texttt{for (}\mathit{lhs}\ \texttt{in}\ e\texttt{)}\ s$	`ForIn(LHS lhs, Expr expr, Stmt body)`
	$\mid$	$\texttt{for (var}\ \mathit{vd}(\texttt{,}\ \mathit{vd})^*\texttt{;}e^?\texttt{;}e^?\texttt{)}\ s$	`ForVar(List<VarDecl> vars, Option<Expr> cond, Option<Expr> action, Stmt body)`
	$\mid$	$\texttt{for (var}\ \mathit{vd}\ \texttt{in}\ e\texttt{)}\ s$	`ForVarIn(VarDecl var, Expr expr, Stmt body)`
	$\mid$	$\texttt{continue}\ x^?\texttt{;}$	`Continue(Option<Label> target)`
	$\mid$	$\texttt{break}\ x^?\texttt{;}$	`Break(Option<Label> target)`
	$\mid$	$\texttt{return}\ e^?\texttt{;}$	`Return(Option<Expr> expr)`
	$\mid$	$\texttt{with (}e\texttt{)}\ s$	`With(Expr expr, Stmt stmt)`
	$\mid$	$l\ \texttt{:}\ s$	`LabelStmt(Label label, Stmt stmt)`
	$\mid$	$\texttt{throw}\ e\texttt{;}$	`Throw(Expr expr)`

	$\mid$	$\texttt{try\{}s^*\texttt{\}}(\texttt{catch (}x\texttt{)}\ \texttt{\{}s^*\texttt{\}})^?(\texttt{finally\{}s^*\texttt{\}})^?$	`Try(Block body, Option<Catch> catchBlock, Option<Block> fin)`
			`Catch(Id id, Block body)`

	$\mid$	$\texttt{debugger;}$	`Debugger()`
cc	$::=$	$\texttt{case}\ e\ \texttt{:}\ s^*$	`Case(Expr cond, Block body)`
e	$::=$	$e\texttt{,}\ e$	`ExprList(List<Expr> exprs)`
	$\mid$	$e\ \texttt{?}\ e\ \texttt{:}\ e$	`Cond(Expr cond, Expr trueBranch, Expr falseBranch)`
	$\mid$	$e\ \otimes\ e$	`InfixOpApp(Expr left, Op op, Expr right)`
	$\mid$	$\ominus\ e$	`PrefixOpApp(Op op, Expr right)`
	$\mid$	$\mathit{lhs}\ \oslash$	`UnaryAssignOpApp(LHS lhs, Op op)`
	$\mid$	$\mathit{lhs}\ \odot\ e$	`AssignOpApp(LHS lhs, Op op, Expr right)`
	$\mid$	lhs	`LHS()`

lhs	::=	*lit*	Literal()
	|	x	VarRef(Id id)
	|	$[(e^?,)^*]$	ArrayExpr(List<Option<Expr>> elements)
	|	$\{(m,)^*\}$	ObjectExpr(List<Member> members)
	|	(e)	Parenthesized(Expr expr)
	|	**function** $x^?$ $((x,)^*)$ $\{fd^*\ vd^*\ s^*\}$	FunExpr(Option<Id> name, Functional ftn)
	|	*lhs* $[e]$	Bracket(LHS obj, Expr index)
	|	*lhs* . x	Dot(LHS obj, Id member)
	|	**new** *lhs*	New(LHS lhs)
	|	*lhs* $((e,)^*)$	FunApp(LHS fun, List<Expr> args)
lit	::=	**this**	This()
	|	**null**	Null()
	|	**true**	Bool(boolean bool)
	|	**false**	Bool(boolean bool)
	|	*num*	DoubleLiteral(ignoreForEquals String text, Double num)
			IntLiteral(BigInteger intVal, int radix)
	|	*str*	StringLiteral(String str, String quote)
	|	*reg*	RegularExpression(String reg)
m	::=	*pr* : e	Field(Property prop, Expr expr)
	|	**get** *pr* () $\{fd^*\ vd^*\ s^*\}$	GetProp(Property prop, Functional ftn)
	|	**set** *pr* (x) $\{fd^*\ vd^*\ s^*\}$	SetProp(Property prop, Functional ftn)
pr	::=	x	PropId(Id id)
	|	*str*	PropStr(String str)
	|	*num*	PropNum(NumberLiteral num)

$\odot$ ::= = | *= | /= | %= | += | -= | <<= | >>= | >>>= | &= | ^= | |=

$\otimes$ ::= && | || | | | & | ^ | << | >> | >>> | + | - | * | / | % | == | != | === | !== | < | > | <= | >=
| **instanceof** | **in**

$\ominus$ ::= ++ | -- | ~ | ! | + | - | **delete** | **void** | **typeof**

$\oslash$ ::= ++ | --

- VarDecl: The expr field is None after Hoister.
- VarStmt, ForVar, ForVarIn: Removed by Hoister.
- RegularExpression: Not yet supported.

- StmtUnit: Internally generated statement unit by Hoister.

Chapter 3

IR

$\underline{p}$	::=	$\underline{s}^*$	`IRRoot`
$\underline{s}$	::=	$\underline{x} = \underline{e}$	`IRExprStmt(IRId lhs, IRExpr right, boolean ref = false)`
	|	$\underline{x}$ = delete $\underline{x}$	`IRDelete(IRId lhs, IRId id)`
	|	$\underline{x}$ = delete $\underline{x}\,[\underline{x}]$	`IRDeleteProp(IRId lhs, IRId obj, IRId index)`
	|	$\underline{x}\,[\underline{x}]\ \ = \underline{e}$	`IRStore(IRId obj, IRId index, IRExpr rhs)`

	|	$\underline{x} = \{(\underline{m},)^*\}$	`IRObject(IRId lhs, List<IRMember> members,`
			`Option<IRId> proto)`

	|	$\underline{x} = [(\underline{e},)^*]$	`IRArray(IRId lhs, List<Option<IRExpr>> elements)`
			`IRArgs(IRId lhs, List<Option<IRExpr>> elements)`
	|	$\underline{x} = \underline{x}\,(\underline{x},\underline{x})$	`IRCall(IRId lhs, IRId fun, IRId thisB, IRId args)`

	|	$\underline{x} = \underline{x}\,(\underline{x}(,\underline{x})^?)$	`IRInternalCall(IRId lhs, IRId fun, IRExpr first,`
			`Option<IRId> second)`

			`toObject, toNumber, isObject, getBase, iteratorInit,`
			`iteratorHasNext, iteratorKey`

	|	$\underline{x}$ = new $\underline{x}\,((\underline{x},)^*)$	`IRNew(IRId lhs, IRId fun, List<IRId> args)`
	|	$\underline{x}$ = function $\underline{f}\,(\underline{x},\underline{x})$ $\{\underline{s}^*\}$	`IRFunExpr(IRId lhs, IRFunctional ftn)`

			`IRFunctional(IRId name, List<IRId> params,`
			`List<IRStmt> args, List<IRFunDecl> fds,`
			`List<IRVarStmt> vds, List<IRStmt> body)`

	|	function $\underline{f}\,(\underline{x},\underline{x})$ $\{\underline{s}^*\}$	`IRFunDecl(IRFunctional ftn)`
	|	$\underline{x}$ = eval ($\underline{e}$)	`IREval(IRId lhs, IRExpr arg)`
	|	break $\underline{x}$	`IRBreak(IRId label)`
	|	return $\underline{e}^?$	`IRReturn(Option<IRExpr> expr)`
	|	with ($\underline{x}$) $\underline{s}$	`IRWith(IRId id, IRStmt stmt)`
	|	$\underline{l}$: { $\underline{s}$ }	`IRLabelStmt(IRId label, IRStmt stmt)`
	|	var $\underline{x}$	`IRVarStmt(IRId lhs)`
	|	throw $\underline{e}$	`IRThrow(IRExpr expr)`
	|	$\underline{s}^*$	`IRSeq(List<IRStmt> stmts)`
	|	if ($\underline{e}$) then $\underline{s}$ (else $\underline{s}$)$^?$	`IRIf(IRExpr expr, IRStmt trueB, Option<IRStmt> falseB)`
	|	while ($\underline{e}$) $\underline{s}$	`IRWhile(IRExpr cond, IRStmt body)`

	|	try {$\underline{s}$} (catch ($\underline{x}$) {$\underline{s}$})$^?$ (finally {$\underline{s}$})$^?$	`IRTry(IRStmt body, Option<IRId> name,`
			`Option<IRStmt> catchB, Option<IRStmt> finallyB)`

	|	$\langle\underline{s}^*\rangle$	`IRStmtUnit(List<IRStmt> stmts)`
$\underline{e}$	::=	$\underline{e} \otimes \underline{e}$	`IRBin(IRExpr first, IROp op, IRExpr second)`
	|	$\ominus\underline{e}$	`IRUn(IROp op, IRExpr expr)`
	|	$\underline{x}\,[\underline{e}]$	`IRLoad(IRId obj, IRExpr index)`
	|	$\underline{x}$	`IRUserId(String text)`
	|	$\underline{\diamond\mathrm{x}}$	`IRTmpId(String text)`
	|	$\underline{num}$	`IRNumber(ignoreForEquals String text, Double num)`
	|	$\underline{str}$	`IRString(String str)`
	|	true	`IRBool(boolean bool)`
	|	false	`IRBool(boolean bool)`
	|	undefined	`IRUndef()`
	|	null	`IRNull()`
	|	this	`IRThis()`

$$\begin{array}{lcll} \underline{m} & ::= & \underline{x} : \underline{e} & \texttt{IRField(IRId prop, IRExpr expr)} \\ & | & \texttt{get}\ \underline{f}\,(\underline{x},\underline{x})\ \ \{\underline{s}^*\} & \texttt{IRGetProp(IRFunctional ftn)} \\ & | & \texttt{set}\ \underline{f}\,(\underline{x},\underline{x})\ \ \{\underline{s}^*\} & \texttt{IRSetProp(IRFunctional ftn)} \end{array}$$

Assumptions and notations:

- Functions and variables are hoisted to their closest enclosing functions or the top level via `Hoister`.
- Identifiers and labels that exist in the source program, except when they appear at top level or within the `with` statement, are already disambiguated via `Disambiguator`, so that they have unique names.
- We use Σ to disambiguate the generated labels and temporary variables in the AST to IR translation. For the presentation brevity, we simply add the newly generated names to Σ.
  - In the actual implementation, we need to create a unique id for each generated name and add the binding information from the general name to the unique id to Σ. For example, when we say "$\Sigma; \underline{\diamond\textsf{break}}$", we actually create a unique id for $\underline{\diamond\textsf{break}}$, say $\underline{\diamond\textsf{break}}_{42}$, and add it to Σ as $\Sigma; \underline{\diamond\textsf{break}} \mapsto \underline{\diamond\textsf{break}}_{42}$. When we look up the environment by $\Sigma(\underline{\diamond\textsf{break}})$, the unique $\underline{\diamond\textsf{break}}_{42}$ is returned.
  - In the scope when the generated name is created, we don't add it to the environment but use the unique id instead of the general name. For example, when we say "$\underline{\diamond\textsf{eq}} = \Sigma(\underline{\diamond\textsf{val}}) {=}{=}{=} \underline{\diamond\textsf{break}};$", we create a unique id for $\underline{\diamond\textsf{eq}}$, say $\underline{\diamond\textsf{eq}}_{910157}$, and it is acually "$\underline{\diamond\textsf{eq}}_{910157} = \Sigma(\underline{\diamond\textsf{val}}) {=}{=}{=} \underline{\diamond\textsf{break}}_{42};$".
  - To be clear, we use blue for the binding sites of such names and red for the use sites of such names.
- We denote a list as a possibly empty, semicolon-separated sequence, enclosed by $\langle$ and $\rangle$.
- We denote a series of list appends as superscripted $*$ such as s^*.
- We denote a fresh variable name as $\underline{\diamond}$ and its variants.
- We abuse our notations by mixing semicolon-separated sequences and lists.
- We use the following:
  $===$, $\diamond$toObject, $\diamond$toNumber, $\diamond$isObject, $\diamond$iteratorInit, $\diamond$iteratorHasNext, $\diamond$iteratorNext, $\diamond$global, $\diamond$getBase
- To denote an AST-level statement granularity in the translated IR statements, we use `IRStmtUnit` which is represented as green angle brackets $\langle\ \rangle$ in this document. To reduce the number of temporary variables, we use global variables to denote constants such as 1 and true which is represented in green 1 and true in this document.
- We wrap a possibly identical assignment with a box so that the actual implementation, `Translator`, can eliminate identical assignments.

Chapter 4

AST to IR

Σ	:	`Env`
$ast2ir_{p}$	:	`Program -> IRRoot`
$ast2ir_{fd}$	:	`FunDecl -> Env -> IRFunDecl`
$ast2ir_{vd}$	:	`VarDecl -> Env -> IRVarStmt`
$ast2ir_{s}$	:	`Stmt -> Env -> IRStmtUnit`
$ast2ir_{case}$	:	`List[Case] * Option[List[Stmt]] * List[Case] -> Env -> List[Option[Expr] * IRId] -> IRStmt`
$ast2ir_{scond}$	:	`List[Option[Expr] * IRId] -> Env -> IRStmt`
$ast2ir_{lval}$	:	`Expr -> Env -> List[IRStmt] -> IRExpr -> boolean -> List[IRStmt] * IRExpr`
$ast2ir_{e}$	:	`Expr -> Env -> IRId -> List[IRStmt] * IRExpr`
$ast2ir_{lhs}$	:	`LHS -> Env -> IRId -> List[IRStmt] * IRExpr`
$ast2ir_{lit}$	:	`LIT -> Env -> IRId -> List[IRStmt] * IRExpr`
$ast2ir_{m}$	:	`Member -> Env -> IRId -> List[IRStmt] * IRMember`
$ast2ir_{pr}$	:	`Property -> IRId`

$$\mathit{ast2ir}_p[\![\mathit{fd}^*\ \mathit{vd}^*\ s^*]\!] \quad = \quad \langle(\mathit{ast2ir}_{\mathit{fd}}[\![\mathit{fd}]\!](\langle\rangle))^*\ (\mathit{ast2ir}_{\mathit{vd}}[\![\mathit{vd}]\!](\langle\rangle))^*\ (\mathit{ast2ir}_s[\![s]\!](\langle\rangle))^*\rangle$$

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{fd}}[\![\mathtt{function}\ f\ \mathtt{(}(x\mathtt{,})^*\mathtt{)}\ \mathtt{\{}\mathit{fd}^*\,\mathit{vd}^*\,s^*\mathtt{\}}]\!](\Sigma) \quad = \quad & \mathsf{function}\ \underline{f}\ \mathtt{(}\underline{\diamond\mathsf{this}}\mathtt{,}\ \underline{\diamond\mathsf{arguments}}\mathtt{)}\ \mathtt{\{} \\
& \quad (\mathit{ast2ir}_{\mathit{fd}}[\![\mathit{fd}]\!](\Sigma))^* \\
& \quad (\mathsf{var}\ \underline{x_i})^* \\
& \quad (\mathit{ast2ir}_{\mathit{vd}}[\![\mathit{vd}]\!](\Sigma))^* \\
& \quad (\underline{x_i} = \underline{\diamond\mathsf{arguments}}\mathtt{["i"]})^* \qquad \textit{where } \underline{x_i} \textit{ is not the name of any of } \mathit{fd} \\
& \\
& \quad (\mathit{ast2ir}_s[\![s]\!](\Sigma;\underline{\diamond\mathsf{this}};\underline{\diamond\mathsf{arguments}}))^*\ \mathtt{\}}
\end{aligned}$$

A function always receives explicit "this" and "arguments" arguments so that the desugaring of `this` *and* `arguments` *is correct. Currently, "arguments" denotes copies of the arguments instead of their aliases. An early exit from a function using* `return` *statements is rewritten as a non-local jump to the label* $\underline{\diamond\mathsf{return}}$.

$$\mathit{ast2ir}_{\mathit{vd}}[\![\mathtt{var}\ x]\!](\Sigma) \quad = \quad \mathsf{var}\ \underline{x}$$

$$\begin{aligned}
\mathit{ast2ir}_s[\![\mathtt{\{}\,s^*\,\mathtt{\}}]\!](\Sigma) \quad &= \quad \langle(\mathit{ast2ir}_s[\![s]\!](\Sigma))^*\rangle \\
\mathit{ast2ir}_s[\![\mathtt{;}]\!](\Sigma) \quad &= \quad \langle\rangle \\
\mathit{ast2ir}_s[\![e\mathtt{;}]\!](\Sigma) \quad &= \quad \mathrm{LET}\ (\underline{s}^*,\underline{e}) = \mathit{ast2ir}_e[\![e]\!](\Sigma)(\underline{\diamond\_}) \\
& \qquad \mathrm{IN}\quad \langle\underline{s}^*\mathtt{;}\ \boxed{\underline{\diamond\_}\ \mathtt{=}\ \underline{e}}\rangle
\end{aligned}$$

Candidate for optimization

$$\begin{aligned}
\mathit{ast2ir}_s[\![\mathtt{if\ (}e_1\mathtt{\&\&}e_2\mathtt{)}\ s_1\ \mathtt{(else}\ s_2\mathtt{)}^?]\!](\Sigma) \quad = \quad & \mathrm{LET}\ (\underline{s}_1^*,\underline{e}_1) = \mathit{ast2ir}_e[\![e_1]\!](\Sigma)(\underline{\diamond\mathsf{new}_1}) \\
& \qquad\ \ (\underline{s}_2^*,\underline{e}_2) = \mathit{ast2ir}_e[\![e_2]\!](\Sigma)(\underline{\diamond\mathsf{new}_2}) \\
& \mathrm{IN}\quad \langle\underline{s}_1^*\mathtt{;} \\
& \qquad \underline{\diamond\mathsf{label}}\ \mathtt{:}\ \mathtt{\{} \\
& \qquad\quad \mathsf{if}\ (\underline{e}_1) \\
& \qquad\quad \mathsf{then}\ \langle\underline{s}_2^*\mathtt{;}\ \mathsf{if}\ (\underline{e}_2)\ \mathsf{then}\ \mathtt{\{}\mathit{ast2ir}_s[\![s_1]\!](\Sigma)\mathtt{;}\ \mathsf{break}\ \underline{\diamond\mathsf{label}}\mathtt{\}}\rangle\mathtt{;} \\
& \qquad\quad (\mathit{ast2ir}_s[\![s_2]\!](\Sigma))^?\ \mathtt{\}}\rangle
\end{aligned}$$

Candidate for optimization

$$\begin{aligned}
\mathit{ast2ir}_s[\![\mathtt{if\ (}e_1\mathtt{||}e_2\mathtt{)}\ s_1\ \mathtt{(else}\ s_2\mathtt{)}^?]\!](\Sigma) \quad = \quad & \mathrm{LET}\ (\underline{s}_1^*,\underline{e}_1) = \mathit{ast2ir}_e[\![e_1]\!](\Sigma)(\underline{\diamond\mathsf{new}_1}) \\
& \qquad\ \ (\underline{s}_2^*,\underline{e}_2) = \mathit{ast2ir}_e[\![e_2]\!](\Sigma)(\underline{\diamond\mathsf{new}_2}) \\
& \mathrm{IN}\quad \langle\underline{s}_1^*\mathtt{;} \\
& \qquad \underline{\diamond\mathsf{label}_2}\ \mathtt{:}\ \mathtt{\{} \\
& \qquad\quad \underline{\diamond\mathsf{label}_1}\ \mathtt{:}\ \mathtt{\{} \\
& \qquad\qquad \mathsf{if}\ (\underline{e}_1) \\
& \qquad\qquad \mathsf{then\ break}\ \underline{\diamond\mathsf{label}_1}\mathtt{;}\ \underline{s}_2^*\mathtt{;} \\
& \qquad\qquad \mathsf{if}\ (\underline{e}_2)\ \mathsf{then\ break}\ \underline{\diamond\mathsf{label}_1}\mathtt{;} \\
& \qquad\qquad (\mathit{ast2ir}_s[\![s_2]\!](\Sigma)\mathtt{;}\ \mathsf{break})^?\ \underline{\diamond\mathsf{label}_2} \\
& \qquad\quad \mathtt{\}}\mathtt{;}\ \mathit{ast2ir}_s[\![s_1]\!](\Sigma)\mathtt{\}}\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{if (}e\texttt{)}\ s_1\ (\texttt{else}\ s_2)^?\rrbracket(\Sigma) &&= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}})\\
& &&\quad \text{IN} \quad \langle \underline{s}^*\texttt{;}\ \mathsf{if}\ (\underline{e})\ \mathsf{then}\ \mathit{ast2ir}_s\llbracket s_1\rrbracket(\Sigma)\ (\mathsf{else}\ \mathit{ast2ir}_s\llbracket s_2\rrbracket(\Sigma))^?\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{switch (}e\texttt{) \{}cc_1^*\ (\texttt{default:}s^*)^?\ cc_2^*\texttt{\}}\rrbracket(\Sigma) &&= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{val}})\\
& &&\quad \text{IN} \quad \langle \underline{\diamond\mathsf{break}}\ \texttt{: \{}\\
& &&\qquad\qquad \underline{s}^*\texttt{;}\ \boxed{\underline{\diamond\mathsf{val}}\ \texttt{=}\ \underline{e}\texttt{;}}\\
& &&\qquad\qquad \mathit{ast2ir}_{case}\llbracket(\mathsf{rev}\ cc_2^*)(s^*)^?(\mathsf{rev}\ cc_1^*)\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{val}})\texttt{\}}\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_{case}\llbracket(\texttt{case}\ e\ \texttt{:}\ s_1^*) :: cc_2^*\ (s_2^*)^?\ cc_1^*\rrbracket(\Sigma)(c^*) &&= \langle\underline{\diamond\mathsf{label}}\ \texttt{: \{}\mathit{ast2ir}_{case}\llbracket cc_2^*\ (s_2^*)^?\ cc_1^*\rrbracket(\Sigma)((e, \underline{\diamond\mathsf{label}}) :: c^*)\texttt{\};}\\
& &&\quad (\mathit{ast2ir}_s\llbracket s_1\rrbracket(\Sigma))^*\rangle\\
&\mathit{ast2ir}_{case}\llbracket()\ (s^*)^?\ cc_1^*\rrbracket(\Sigma)(c^*) &&= \langle\underline{\diamond\mathsf{label}}\ \texttt{: \{}\mathit{ast2ir}_{case}\llbracket()\ ()\ cc_1^*\rrbracket(\Sigma)(c^*@[((), \underline{\diamond\mathsf{label}})])\texttt{\};}\\
& &&\quad ((\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma))^*)^?\rangle\\
&\mathit{ast2ir}_{case}\llbracket()\ ()\ (\texttt{case}\ e\ \texttt{:}\ s^*) :: cc_1^*\rrbracket(\Sigma)(c^*) &&= \langle\underline{\diamond\mathsf{label}}\ \texttt{: \{}\mathit{ast2ir}_{case}\llbracket()\ ()\ cc_1^*\rrbracket(\Sigma)((e, \underline{\diamond\mathsf{label}}) :: c^*)\texttt{\};}\\
& &&\quad (\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma))^*\rangle\\
&\mathit{ast2ir}_{case}\llbracket()\ ()\ ()\rrbracket(\Sigma)((e, \underline{l})^*) &&= \langle\mathit{ast2ir}_{scond}\llbracket(e, \underline{l})^*\rrbracket(\Sigma)\texttt{;}\\
& &&\quad \mathsf{break}\ \Sigma(\underline{\diamond\mathsf{break}})\rangle\\
&\mathit{ast2ir}_{scond}\llbracket(e, \underline{l}) :: (c^*)\rrbracket(\Sigma) &&= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{cond}})\\
& &&\quad \text{IN} \quad \langle\underline{s}^*\texttt{;}\\
& &&\qquad\quad \mathsf{if}\ (\Sigma(\underline{\diamond\mathsf{val}})\ \texttt{===}\ \underline{e})\ \mathsf{then\ break}\ \underline{l}\ \mathsf{else}\ \mathit{ast2ir}_{scond}\llbracket c^*\rrbracket(\Sigma)\rangle\\
&\mathit{ast2ir}_{scond}\llbracket[((), \underline{l})]\rrbracket(\Sigma) &&= \langle\mathsf{break}\ \underline{l}\rangle\\
&\mathit{ast2ir}_{scond}\llbracket()\rrbracket(\Sigma) &&= \langle\rangle
\end{aligned}$$

Where c is either $(e, \underline{l})$ or $((), \underline{l})$.

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{do}\ s\ \texttt{while (}e\texttt{);}\rrbracket(\Sigma) &&= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& &&\quad \text{IN} \quad \langle\underline{\diamond\mathsf{break}}\ \texttt{: \{}\\
& &&\qquad\qquad \underline{\diamond\mathsf{continue}}\ \texttt{: \{}\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{continue}})\texttt{\};}\\
& &&\qquad\qquad \underline{s}^*\texttt{;}\\
& &&\qquad\qquad \mathsf{while}\ (\underline{e})\ \texttt{\{}\\
& &&\qquad\qquad\quad \underline{\diamond\mathsf{continue}}\ \texttt{: \{}\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{continue}})\texttt{\};}\\
& &&\qquad\qquad\quad \underline{s}^*\texttt{;}\\
& &&\qquad\qquad \texttt{\}}\\
& &&\qquad \texttt{\}}\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{while (}e\texttt{)}\ s\rrbracket(\Sigma) &&= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& &&\quad \text{IN} \quad \langle\underline{\diamond\mathsf{break}}\ \texttt{: \{}\\
& &&\qquad\qquad \underline{s}^*\texttt{;}\\
& &&\qquad\qquad \mathsf{while}\ (\underline{e})\ \texttt{\{}\\
& &&\qquad\qquad\quad \underline{\diamond\mathsf{continue}}\ \texttt{: \{}\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{continue}})\texttt{\};}\\
& &&\qquad\qquad\quad \underline{s}^*\texttt{;}\\
& &&\qquad\qquad \texttt{\}}\\
& &&\qquad \texttt{\}}\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{for (}e_1^?\texttt{;;}e_3^?\texttt{)}\ s\rrbracket(\Sigma) &&= \text{LET } ((\underline{s}_1^*, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond_{\_}}))^?\\
& &&\qquad\quad ((\underline{s}_3^*, \underline{e}_3) = \mathit{ast2ir}_e\llbracket e_3\rrbracket(\Sigma)(\underline{\diamond_{\_}}))^?\\
& &&\quad \text{IN} \quad \langle\underline{\diamond\mathsf{break}}\ \texttt{: \{}\\
& &&\qquad\qquad (\underline{s}_1^*\texttt{;}\boxed{\underline{\diamond_{\_}} = \underline{e}_1})^?\\
& &&\qquad\qquad \mathsf{while}\ (\mathsf{true})\ \texttt{\{}\\
& &&\qquad\qquad\quad \underline{\diamond\mathsf{continue}}\ \texttt{: \{}\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{continue}})\texttt{\};}\\
& &&\qquad\qquad\quad (\underline{s}_3^*\texttt{;}\boxed{\underline{\diamond_{\_}} = \underline{e}_3})^?\\
& &&\qquad\qquad \texttt{\}}\\
& &&\qquad \texttt{\}}\rangle
\end{aligned}$$

$$\begin{aligned}
&\mathit{ast2ir}_s\llbracket \texttt{for (}e_1^?\texttt{;}e_2\texttt{;}e_3^?\texttt{)}\ s\rrbracket(\Sigma) &&= \text{LET } ((\underline{s}_1^*, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond_{\_}}))^?\\
& &&\qquad\quad (\underline{s}_2^*, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_2})\\
& &&\qquad\quad ((\underline{s}_3^*, \underline{e}_3) = \mathit{ast2ir}_e\llbracket e_3\rrbracket(\Sigma)(\underline{\diamond_{\_}}))^?\\
& &&\quad \text{IN} \quad \langle\underline{\diamond\mathsf{break}}\ \texttt{: \{}\\
& &&\qquad\qquad (\underline{s}_1^*\texttt{;}\boxed{\underline{\diamond_{\_}} = \underline{e}_1})^?\\
& &&\qquad\qquad \underline{s}_2^*\texttt{;}\\
& &&\qquad\qquad \mathsf{while}\ (\underline{e}_2)\ \texttt{\{}\\
& &&\qquad\qquad\quad \underline{\diamond\mathsf{continue}}\ \texttt{: \{}\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{break}}; \underline{\diamond\mathsf{continue}})\texttt{\};}\\
& &&\qquad\qquad\quad (\underline{s}_3^*\texttt{;}\boxed{\underline{\diamond_{\_}} = \underline{e}_3})^?\\
& &&\qquad\qquad\quad \underline{s}_2^*\texttt{;}\\
& &&\qquad\qquad \texttt{\}}\\
& &&\qquad \texttt{\}}\rangle
\end{aligned}$$

$$
\begin{array}{ll}
\mathit{ast2ir}_s\llbracket\texttt{for (}\mathit{lhs}\ \texttt{in}\ e\texttt{)}\ s\rrbracket(\Sigma) & = \text{LET}\ (\underline{s}^*,\underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& \quad \text{IN}\quad \langle\underline{\diamond\mathsf{break}}\ \texttt{:}\ \{\\
& \qquad\qquad \underline{s}^*\texttt{;}\\
& \qquad\qquad \underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}\,(\underline{e})\,\texttt{;}\\
& \qquad\qquad \underline{\diamond\mathsf{iterator}} = \diamond\mathsf{iteratorInit}(\underline{\diamond\mathsf{obj}})\texttt{;}\\
& \qquad\qquad \underline{\diamond\mathsf{cond}_1} = \diamond\mathsf{iteratorHasNext}(\underline{\diamond\mathsf{obj}},\underline{\diamond\mathsf{iterator}})\texttt{;}\\
& \qquad\qquad \mathsf{while}\ (\underline{\diamond\mathsf{cond}_1})\ \ \{\\
& \qquad\qquad\quad \underline{\diamond\mathsf{key}} = \diamond\mathsf{iteratorNext}(\underline{\diamond\mathsf{obj}},\underline{\diamond\mathsf{iterator}})\texttt{;}\\
& \qquad\qquad\quad \mathit{ast2ir}_{\mathit{lval}}\llbracket\mathit{lhs}\rrbracket(\Sigma)(;\ \underline{\diamond\mathsf{key}})(\text{false})._1\texttt{;}\\
& \qquad\qquad\quad \underline{\diamond\mathsf{continue}}\texttt{:}\ \{\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma;\underline{\diamond\mathsf{break}};\underline{\diamond\mathsf{continue}})\}\ \texttt{;}\\
& \qquad\qquad\quad \underline{\diamond\mathsf{cond}_1} = \diamond\mathsf{iteratorHasNext}(\underline{\diamond\mathsf{obj}},\underline{\diamond\mathsf{iterator}})\texttt{;}\\
& \qquad\qquad \}\\
& \qquad\ \}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{continue;}\rrbracket(\Sigma) & = \langle\mathsf{break}\ \Sigma(\underline{\diamond\mathsf{continue}})\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{continue}\ l\texttt{;}\rrbracket(\Sigma) & = \langle\mathsf{break}\ \underline{l}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{break;}\rrbracket(\Sigma) & = \langle\mathsf{break}\ \Sigma(\underline{\diamond\mathsf{break}})\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{break}\ l\texttt{;}\rrbracket(\Sigma) & = \langle\mathsf{break}\ \underline{l}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{return;}\rrbracket(\Sigma) & = \langle\mathsf{return}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{return}\ e\texttt{;}\rrbracket(\Sigma) & = \text{LET}\ (\underline{s}^*,\underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& \quad \text{IN}\quad \langle\underline{s}^*\texttt{;}\ \mathsf{return}\ \underline{e}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{with (}e\texttt{)}\ s\rrbracket(\Sigma) & = \text{LET}\ (\underline{s}^*,\underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& \quad \text{IN}\quad \langle\underline{s}^*\texttt{;}\\
& \qquad\quad \underline{\diamond\mathsf{new}_2} = \diamond\mathsf{toObject}(\underline{e});\\
& \qquad\quad \mathsf{with}\ (\underline{\diamond\mathsf{new}_2})\ \mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma)\rangle\\
\mathit{ast2ir}_s\llbracket l\ \texttt{:}\ s\rrbracket(\Sigma) & = \langle\underline{l}\ \texttt{:}\ \{\ \mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma)\ \}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{throw}\ e\texttt{;}\rrbracket(\Sigma) & = \text{LET}\ (\underline{s}^*,\underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
& \quad \text{IN}\quad \langle\underline{s}^*\texttt{;}\ \mathsf{throw}\ \underline{e}\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{try \{}s_1^*\texttt{\}}\ (\texttt{catch(}x\texttt{)\{}s_2^*\texttt{\}})^?(\texttt{finally \{}s_3^*\texttt{\}})^?\rrbracket(\Sigma) & = \langle\mathsf{try}\ \texttt{\{}(\mathit{ast2ir}_s\llbracket s_1\rrbracket(\Sigma))^*\texttt{\}}\\
& \quad (\texttt{catch(}\underline{x}\texttt{)\{}(\mathit{ast2ir}_s\llbracket s_2\rrbracket(\Sigma))^*\texttt{\}})^?\\
& \quad (\texttt{finally \{}(\mathit{ast2ir}_s\llbracket s_3\rrbracket(\Sigma))^*\texttt{\}})^?\rangle\\
\mathit{ast2ir}_s\llbracket\texttt{debugger;}\rrbracket(\Sigma) & = \langle\rangle
\end{array}
$$

$$
\begin{array}{ll}
\mathit{ast2ir}_{\mathit{lval}}\llbracket\texttt{(}e\texttt{)}\rrbracket(\Sigma)(\underline{s}^*;\underline{e}')(\text{keepOld}) & = \mathit{ast2ir}_{\mathit{lval}}\llbracket e\rrbracket(\Sigma)(\underline{s}^*;\underline{e}')(\text{keepOld})\\
\mathit{ast2ir}_{\mathit{lval}}\llbracket x\rrbracket(\Sigma)(\underline{s}^*;\underline{e})(\text{keepOld}) & = \text{IF keepOld THEN}\ (\langle\underline{\diamond\mathsf{old}} = \underline{x}\texttt{;}\ \underline{s}^*\texttt{;}\ \underline{x} = \underline{e}\rangle,\underline{x})\\
& \quad \text{ELSE}\ \langle\underline{s}^*\texttt{;}\ \underline{x} = \underline{e}\rangle\\
\mathit{ast2ir}_{\mathit{lval}}\llbracket\mathit{lhs}\,\texttt{.}\,x\rrbracket(\Sigma)(\underline{s}^*;\underline{e})(\text{keepOld}) & = \mathit{ast2ir}_{\mathit{lval}}\llbracket\mathit{lhs}\,\texttt{["}x\texttt{"]}\rrbracket(\Sigma)(\underline{s}^*;\underline{e})(\text{keepOld})\\
\mathit{ast2ir}_{\mathit{lval}}\llbracket\mathit{lhs}\,\texttt{[}e\texttt{]}\rrbracket(\Sigma)(\underline{s}^*;\underline{e}')(\text{keepOld}) & = \text{LET}\ (\underline{s}_1^*,\underline{e}_1) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket\mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{obj}_1})\\
& \qquad\quad (\underline{s}_2^*,\underline{e}_2) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{field}_1})\\
& \quad \text{IN}\quad \text{IF keepOld}\\
& \qquad\quad \text{THEN}\ (\langle\underline{s}_1^*\texttt{;}\underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_1)\texttt{;}\,\underline{s}_2^*\texttt{;}\\
& \qquad\qquad\quad \underline{\diamond\mathsf{old}} = \underline{\diamond\mathsf{obj}}\,\texttt{[}\underline{e}_2\texttt{]}\,\texttt{;}\underline{s}^*\texttt{;}\underline{\diamond\mathsf{obj}}\,\texttt{[}\underline{e}_2\texttt{]}\ = \underline{e}'\rangle,\\
& \qquad\qquad\ \underline{\diamond\mathsf{obj}}\,\texttt{[}\underline{e}_2\texttt{]})\\
& \qquad\quad \text{ELSE}\ (\langle\underline{s}_1^*\texttt{;}\underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_1)\texttt{;}\,\underline{s}_2^*\texttt{;}\\
& \qquad\qquad\quad \underline{s}^*\texttt{;}\underline{\diamond\mathsf{obj}}\,\texttt{[}\underline{e}_2\texttt{]}\ = \underline{e}'\rangle,\underline{\diamond\mathsf{obj}}\,\texttt{[}\underline{e}_2\texttt{]})\\
\mathit{ast2ir}_{\mathit{lval}}\llbracket e\rrbracket(\Sigma)(\underline{s}^*;\underline{e})(\text{keepOld}) & = \text{Warning: ReferenceError!}
\end{array}
$$

$$
\begin{array}{ll}
\mathit{ast2ir}_e\llbracket e_1\texttt{,}\ e_2\rrbracket(\Sigma)(\underline{x}) & = \text{LET}\ (\underline{s}_1^*,\underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}})\\
& \qquad\quad (\underline{s}_2^*,\underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{x})\\
& \quad \text{IN}\quad (\underline{s}_1^*\texttt{;}\underline{\diamond\mathsf{y}} = \underline{e}_1\texttt{;}\,\underline{s}_2^*,\underline{e}_2)
\end{array}
$$

Candidate for optimization

$$
\begin{array}{ll}
\mathit{ast2ir}_e\llbracket e_a\,\texttt{\&\&}\,e_b\ \texttt{?}\ e_2\ \texttt{:}\ e_3\rrbracket(\Sigma)(\underline{x}) & = \text{LET}\ (\underline{s}_a^*,\underline{e}_a) = \mathit{ast2ir}_e\llbracket e_a\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_a})\\
& \qquad\quad (\underline{s}_b^*,\underline{e}_b) = \mathit{ast2ir}_e\llbracket e_b\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_b})\\
& \qquad\quad (\underline{s}_2^*,\underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{x})\\
& \qquad\quad (\underline{s}_3^*,\underline{e}_3) = \mathit{ast2ir}_e\llbracket e_3\rrbracket(\Sigma)(\underline{x})\\
& \quad \text{IN}\quad (\underline{s}_a^*\texttt{;}\\
& \qquad\quad \underline{\diamond\mathsf{label}}\ \texttt{:}\ \{\\
& \qquad\qquad \mathsf{if}\ (\underline{e}_a)\\
& \qquad\qquad \mathsf{then}\ \langle\underline{s}_b^*\texttt{;}\ \mathsf{if}\ (\underline{e}_b)\ \mathsf{then}\ \{\underline{s}_2^*\texttt{;}\ \boxed{\underline{x} = \underline{e}_2}\texttt{;}\ \mathsf{break}\ \underline{\diamond\mathsf{label}}\}\rangle\texttt{;}\\
& \qquad\qquad \underline{s}_3^*\texttt{;}\ \boxed{\underline{x} = \underline{e}_3}\},\underline{x})
\end{array}
$$

Candidate for optimization

$$
\begin{aligned}
\mathit{ast2ir}_e\llbracket e_a \mid\mid e_b \;?\; e_2 : e_3\rrbracket(\Sigma)(\underline{x}) = \text{LET } & (\underline{s}^*_a, \underline{e}_a) = \mathit{ast2ir}_e\llbracket e_a\rrbracket(\Sigma)(\underline{\diamond\mathsf{new_a}}) \\
& (\underline{s}^*_b, \underline{e}_b) = \mathit{ast2ir}_e\llbracket e_b\rrbracket(\Sigma)(\underline{\diamond\mathsf{new_b}}) \\
& (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{x}) \\
& (\underline{s}^*_3, \underline{e}_3) = \mathit{ast2ir}_e\llbracket e_3\rrbracket(\Sigma)(\underline{x}) \\
\text{IN } & (\underline{s}^*_a\,; \\
& \quad \underline{\diamond\mathsf{label}_2} : \{ \\
& \quad\quad \underline{\diamond\mathsf{label}_1} : \{ \\
& \quad\quad\quad \mathsf{if}\ (\underline{e}_a) \\
& \quad\quad\quad \mathsf{then\ break}\ \underline{\diamond\mathsf{label}_1}\,;\,\underline{s}^*_b\,; \\
& \quad\quad\quad \mathsf{if}\ (\underline{e}_b)\ \mathsf{then\ break}\ \underline{\diamond\mathsf{label}_1}\,; \\
& \quad\quad\quad \underline{s}^*_3\,;\ \boxed{\underline{x} = \underline{e}_3}\,;\ \mathsf{break}\ \underline{\diamond\mathsf{label}_2} \\
& \quad\quad \}\,;\ \underline{s}^*_2\,;\ \boxed{\underline{x} = \underline{e}_2}\,\}\,, \underline{x})
\end{aligned}
$$

$$
\begin{aligned}
\mathit{ast2ir}_e\llbracket e_1 \;?\; e_2 : e_3\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*_1, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond\mathsf{new_1}}) \\
&\qquad\quad (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{x}) \\
&\qquad\quad (\underline{s}^*_3, \underline{e}_3) = \mathit{ast2ir}_e\llbracket e_3\rrbracket(\Sigma)(\underline{x}) \\
&\quad\ \text{IN}\quad (\underline{s}^*_1\,; \mathsf{if}\ (\underline{e}_1)\ \mathsf{then}\ \{\underline{s}^*_2\,;\ \boxed{\underline{x} = \underline{e}_2}\,\}\ \mathsf{else}\ \{\underline{s}^*_3\,;\ \boxed{\underline{x} = \underline{e}_3}\,\}\,, \underline{x}) \\
\mathit{ast2ir}_e\llbracket \mathit{lhs} = e\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{x}) \\
&\quad\ \text{IN}\quad \text{IF } \underline{e} \text{ contains } \mathit{lhs} \\
&\qquad\quad \text{THEN } \mathit{ast2ir}_{\mathit{lval}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\ \underline{s}^*;\ \underline{e}\ )(\text{false}) \\
&\qquad\quad \text{ELSE } (\mathit{ast2ir}_{\mathit{lval}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\ \underline{s}^*;\ \underline{e}\ )(\text{false})._1, \underline{e}) \\
\mathit{ast2ir}_e\llbracket \mathit{lhs} \odot = e\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\quad\ \text{IN}\quad (\mathit{ast2ir}_{\mathit{lval}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\ \underline{s}^*;\ \underline{\diamond\mathsf{old}} \odot \underline{e})(\text{true})._1, \underline{\diamond\mathsf{old}} \odot \underline{e}) \\
\mathit{ast2ir}_e\llbracket {+}{+}e\rrbracket(\Sigma)(\underline{x}) &= (\mathit{ast2ir}_{\mathit{lval}}\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}} = \diamond\mathsf{toNumber}(\underline{\diamond\mathsf{old}});\ \underline{\diamond\mathsf{new}}\ {+}1)(\text{true})._1, \underline{\diamond\mathsf{new}}\ {+}1) \\
\mathit{ast2ir}_e\llbracket {-}{-}e\rrbracket(\Sigma)(\underline{x}) &= (\mathit{ast2ir}_{\mathit{lval}}\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}} = \diamond\mathsf{toNumber}(\underline{\diamond\mathsf{old}});\ \underline{\diamond\mathsf{new}}\ {-}1)(\text{true})._1, \underline{\diamond\mathsf{new}}\ {-}1) \\
\mathit{ast2ir}_e\llbracket \mathsf{delete}\ x\rrbracket(\Sigma)(\underline{y}) &= (\langle \underline{y} =\ \mathsf{delete}\ x\rangle, \underline{y}) \\
\mathit{ast2ir}_e\llbracket \mathsf{delete}\ (x)\ \rrbracket(\Sigma)(\underline{y}) &= (\langle \underline{y} =\ \mathsf{delete}\ x\rangle, \underline{y}) \\
\mathit{ast2ir}_e\llbracket \mathsf{delete}\ \mathit{lhs}\,.\,x\rrbracket(\Sigma)(\underline{y}) &= \mathit{ast2ir}_e\llbracket \mathsf{delete}\ \mathit{lhs}\,[\,\texttt{"}x\texttt{"}\,]\,\rrbracket(\Sigma)(\underline{y}) \\
\mathit{ast2ir}_e\llbracket \mathsf{delete}\ \mathit{lhs}\,[\,e\,]\,\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*_1, \underline{e}_1) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{obj_1}}) \\
&\qquad\quad (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{field_1}}) \\
&\quad\ \text{IN}\quad (\underline{s}^*_1\,; \underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_1)\,; \underline{s}^*_2\,; \\
&\qquad\qquad \underline{x} =\ \mathsf{delete}\ \underline{\diamond\mathsf{obj}}\,[\,\underline{e}_2\,]\,, \underline{x}) \\
\mathit{ast2ir}_e\llbracket \mathsf{delete}\ e\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\quad\ \text{IN}\quad (\underline{s}^*\,; \underline{\diamond_} =\ \underline{e}, \mathsf{true}) \\
\mathit{ast2ir}_e\llbracket \ominus e\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\quad\ \text{IN}\quad (\underline{s}^*, \ominus\underline{e}) \\
\mathit{ast2ir}_e\llbracket \mathit{lhs}{+}{+}\rrbracket(\Sigma)(\underline{x}) &= (\mathit{ast2ir}_{\mathit{lval}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}} = \diamond\mathsf{toNumber}(\underline{\diamond\mathsf{old}});\ \underline{\diamond\mathsf{new}}\ {+}1)(\text{true})._1, \underline{\diamond\mathsf{new}}) \\
\mathit{ast2ir}_e\llbracket \mathit{lhs}{-}{-}\rrbracket(\Sigma)(\underline{x}) &= (\mathit{ast2ir}_{\mathit{lval}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}} = \diamond\mathsf{toNumber}(\underline{\diamond\mathsf{old}});\ \underline{\diamond\mathsf{new}}\ {-}1)(\text{true})._1, \underline{\diamond\mathsf{new}})
\end{aligned}
$$

Candidate for optimization

$$
\begin{aligned}
\mathit{ast2ir}_e\llbracket e_1 \,\&\&\, e_2\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*_1, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\qquad\quad (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{\diamond\mathsf{z}}) \\
&\quad\ \text{IN}\quad (\underline{s}^*_1\,; \mathsf{if}\ (\underline{e}_1)\ \mathsf{then}\ \underline{s}^*_2\,;\ \underline{x} = \underline{e}_2\ \mathsf{else}\ \underline{x} = \underline{e}_1, \underline{x})
\end{aligned}
$$

Candidate for optimization

$$
\begin{aligned}
\mathit{ast2ir}_e\llbracket e_1 \mid\mid e_2\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*_1, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\qquad\quad (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{\diamond\mathsf{z}}) \\
&\quad\ \text{IN}\quad (\underline{s}^*_1\,; \mathsf{if}\ (\underline{e}_1)\ \mathsf{then}\ \underline{x} = \underline{e}_1 \mathsf{else}\ \underline{s}^*_2\,;\ \underline{x} = \underline{e}_2, \underline{x})
\end{aligned}
$$

In order to preserve the semantics when the evaluation of $\underline{e}_1$ throws an exception, we force to evaluate $\underline{e}_1$ before evaluating $\underline{s}^*_2$ by introducing an assignment “$\underline{\diamond\mathsf{new_1}} = \underline{e}_1$” to avoid any side effects by $\underline{s}^*_2$. Note that we add the assignment only when $\underline{s}^*_2$ is not empty for a simple optimization.

Candidate for optimization

$$
\begin{aligned}
\mathit{ast2ir}_e\llbracket e_1 \otimes e_2\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*_1, \underline{e}_1) = \mathit{ast2ir}_e\llbracket e_1\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}) \\
&\qquad\quad (\underline{s}^*_2, \underline{e}_2) = \mathit{ast2ir}_e\llbracket e_2\rrbracket(\Sigma)(\underline{\diamond\mathsf{z}}) \\
&\quad\ \text{IN}\quad \text{IF } \underline{s}^*_2 \text{ is empty} \\
&\qquad\quad \text{THEN } (\underline{s}^*_1, \underline{e}_1 \otimes \underline{e}_2) \\
&\qquad\quad \text{ELSE } (\underline{s}^*_1\,;\ \underline{\diamond\mathsf{y}} = \underline{e}_1\,;\ \underline{s}^*_2, \underline{\diamond\mathsf{y}} \otimes \underline{e}_2) \\
\mathit{ast2ir}_e\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{x}) &= \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{x})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lit} \rrbracket(\Sigma)(\underline{x}) &&= \mathit{ast2ir}_{\mathit{lit}}\llbracket \mathit{lit} \rrbracket(\Sigma)(\underline{x})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{arguments} \rrbracket(\Sigma)(\underline{x}) &&= (\langle\rangle, \Sigma(\underline{\diamond\mathsf{arguments}}))\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket x \rrbracket(\Sigma)(\underline{y}) &&= (\langle\rangle, \underline{x})
\end{aligned}
$$

Candidate for optimization

$$
\begin{aligned}
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{[}(e^{?}\texttt{,})^{*}\texttt{]} \rrbracket(\Sigma)(\underline{x}) &&= \text{LET } ((\underline{s}^{*}, \underline{e}) = \mathit{ast2ir}_{e}\llbracket e \rrbracket(\Sigma)(\underline{\diamond\mathsf{elem}}))^{*}\\
& &&\quad \text{IN } \quad ((\underline{s}^{*}\texttt{;}\ \underline{\diamond\mathsf{elem}} = \underline{e})^{*}\texttt{;}\ \underline{x} = \texttt{[}(\underline{\diamond\mathsf{elem}}\texttt{,})^{*}\texttt{]}, \underline{x})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{\{}(m\texttt{,})^{*}\texttt{\}} \rrbracket(\Sigma)(\underline{x}) &&= \text{LET } ((\underline{s}^{*}, \underline{\mathit{mem}}) = \mathit{ast2ir}_{m}\llbracket m \rrbracket(\Sigma)(\underline{\diamond\mathsf{member}}))^{*}\\
& &&\quad \text{IN } \quad ((\underline{s}^{*})^{*}\texttt{;}\ \underline{x} = \texttt{\{}(\underline{\mathit{mem}}\texttt{,})^{*}\texttt{\}}, \underline{x})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{(}e\texttt{)} \rrbracket(\Sigma)(\underline{x}) &&= \mathit{ast2ir}_{e}\llbracket e \rrbracket(\Sigma)(\underline{x})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{function}\ f^{?}\ \texttt{(}(x\texttt{,})^{*}\texttt{)}\ \texttt{\{}\mathit{fd}^{*}\,\mathit{vd}^{*}\,s^{*}\texttt{\}} \rrbracket(\Sigma)(\underline{y}) &&= (\langle \underline{y} = \mathsf{function}\ \underline{f}^{?}\ \texttt{(}\underline{\diamond\mathsf{this}}\texttt{,}\ \underline{\diamond\mathsf{arguments}}\texttt{)}\ \texttt{\{}\\
& &&\qquad\qquad (\mathit{ast2ir}_{\mathit{fd}}\llbracket \mathit{fd} \rrbracket(\Sigma))^{*}\\
& &&\qquad\qquad (\mathsf{var}\ \underline{x_i})^{*}\\
& &&\qquad\qquad (\mathit{ast2ir}_{\mathit{vd}}\llbracket \mathit{vd} \rrbracket(\Sigma))^{*}\\
& &&\qquad\qquad (\underline{x_i} = \underline{\diamond\mathsf{arguments}}\texttt{["i"]})^{*} \qquad \textit{where } \underline{x_i} \textit{ is not the name of any of fd}\\
& &&\qquad\qquad (\mathit{ast2ir}_{s}\llbracket s \rrbracket(\Sigma; \underline{\diamond\mathsf{this}}; \underline{\diamond\mathsf{arguments}}))^{*}\,\texttt{\}}\rangle, \underline{y})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,\texttt{.}\,x \rrbracket(\Sigma)(\underline{y}) &&= \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,\texttt{["}x\texttt{"]} \rrbracket(\Sigma)(\underline{y})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,\texttt{["}x\texttt{"]} \rrbracket(\Sigma)(\underline{y}) &&= \text{LET } (\underline{s}_1^{*}, \underline{e}_1) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs} \rrbracket(\Sigma)(\underline{\diamond\mathsf{obj}_1})\\
& &&\quad \text{IN } \quad (\underline{s}_1^{*}\texttt{;}\ \underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_1), \underline{\diamond\mathsf{obj}}\texttt{["x"]})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,\texttt{[}e\texttt{]} \rrbracket(\Sigma)(\underline{x}) &&= \text{LET } (\underline{s}_1^{*}, \underline{e}_1) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs} \rrbracket(\Sigma)(\underline{\diamond\mathsf{obj}_1})\\
& &&\qquad\quad (\underline{s}_2^{*}, \underline{e}_2) = \mathit{ast2ir}_{e}\llbracket e \rrbracket(\Sigma)(\underline{\diamond\mathsf{field}_1})\\
& &&\quad \text{IN } \quad (\underline{s}_1^{*}\texttt{;}\ \underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_1)\texttt{;}\ \underline{s}_2^{*}, \underline{\diamond\mathsf{obj}}\texttt{[}\underline{e}_2\texttt{]})
\end{aligned}
$$

Candidate for optimization

$$
\begin{aligned}
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{new}\ \mathit{lhs}\,\texttt{(}(e,)^{*}\texttt{)} \rrbracket(\Sigma)(\underline{x}) &&= \text{LET } (\underline{s}_l^{*}, \underline{e}_l) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs} \rrbracket(\Sigma)(\underline{\diamond\mathsf{fun}_1})\\
& &&\qquad\quad ((\underline{s}^{*}, \underline{e}) = \mathit{ast2ir}_{e}\llbracket e \rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}))^{*}\\
& &&\quad \text{IN } \quad (\underline{s}_l^{*}\texttt{;}\ \underline{\diamond\mathsf{fun}} = \diamond\mathsf{toObject}(\underline{e}_l)\texttt{;}\ (\underline{s}^{*}\texttt{;}\ \underline{\diamond\mathsf{y}} = \underline{e})^{*}\texttt{;}\\
& &&\qquad\quad \underline{\diamond\mathsf{arguments}} = \texttt{[}(\underline{\diamond\mathsf{y}_i}\texttt{,})^{*}\texttt{];}\\
& &&\qquad\quad \underline{\diamond\mathsf{proto}} = \underline{\diamond\mathsf{fun}}\texttt{["prototype"];}\\
& &&\qquad\quad \underline{\diamond\mathsf{obj}} = \texttt{\{[[Prototype]] = }\underline{\diamond\mathsf{proto}}\texttt{\};}\\
& &&\qquad\quad \underline{\diamond\mathsf{newObj}} = \mathsf{new}\ \underline{\diamond\mathsf{fun}}\ \texttt{(}\underline{\diamond\mathsf{obj}}\texttt{,}\ \underline{\diamond\mathsf{arguments}}\texttt{);}\\
& &&\qquad\quad \underline{\diamond\mathsf{cond}} = \diamond\mathsf{isObject}(\underline{\diamond\mathsf{newObj}})\texttt{;}\\
& &&\qquad\quad \mathsf{if}(\underline{\diamond\mathsf{cond}})\ \mathsf{then}\ \underline{x} = \underline{\diamond\mathsf{newObj}}\ \ \mathsf{else}\ \underline{x} = \underline{\diamond\mathsf{obj}}, \underline{x})\\
&\mathit{ast2ir}_{\mathit{lhs}}\llbracket \texttt{new}\ \mathit{lhs} \rrbracket(\Sigma)(\underline{x}) &&= \text{LET } (\underline{s}^{*}, \underline{e}) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs} \rrbracket(\Sigma)(\underline{\diamond\mathsf{fun}_1})\\
& &&\quad \text{IN } \quad (\underline{s}^{*}\texttt{;}\ \underline{\diamond\mathsf{fun}} = \diamond\mathsf{toObject}(\underline{e})\texttt{;}\\
& &&\qquad\quad \underline{\diamond\mathsf{arguments}} = \texttt{[];}\\
& &&\qquad\quad \underline{\diamond\mathsf{proto}} = \underline{\diamond\mathsf{fun}}\texttt{["prototype"];}\\
& &&\qquad\quad \underline{\diamond\mathsf{obj}} = \texttt{\{[[Prototype]] = }\underline{\diamond\mathsf{proto}}\texttt{\};}\\
& &&\qquad\quad \underline{\diamond\mathsf{newObj}} = \mathsf{new}\ \underline{\diamond\mathsf{fun}}\ \texttt{(}\underline{\diamond\mathsf{obj}}\texttt{,}\ \underline{\diamond\mathsf{arguments}}\texttt{);}\\
& &&\qquad\quad \underline{\diamond\mathsf{cond}} = \diamond\mathsf{isObject}(\underline{\diamond\mathsf{newObj}})\texttt{;}\\
& &&\qquad\quad \mathsf{if}(\underline{\diamond\mathsf{cond}})\ \mathsf{then}\ \underline{x} = \underline{\diamond\mathsf{newObj}}\ \ \mathsf{else}\ \underline{x} = \underline{\diamond\mathsf{obj}}, \underline{x})
\end{aligned}
$$

$$
\begin{aligned}
\odot &::= \star \mid \texttt{/} \mid \texttt{\%} \mid \texttt{+} \mid \texttt{-} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{|}\\
\ominus &::= \texttt{\~{}} \mid \texttt{!} \mid \texttt{+} \mid \texttt{-} \mid \texttt{delete} \mid \texttt{void} \mid \texttt{typeof}\\
\otimes &::= \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \mid \texttt{+} \mid \texttt{-} \mid \star \mid \texttt{/} \mid \texttt{\%} \mid \texttt{==} \mid \texttt{!=} \mid \texttt{===} \mid \texttt{!==} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{instanceof} \mid \texttt{in}
\end{aligned}
$$

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{eval}\,(\,e\,)\,\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{new}_1})\\
&\quad \text{IN}\quad (\underline{s}^*\,;\,\underline{x} = \mathsf{eval}\,(\,\underline{e}\,)\,,\,\underline{x})\\
\mathit{ast2ir}_{\mathit{lhs}}\llbracket\,(\,f\,)\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x}) &= \mathit{ast2ir}_{\mathit{lhs}}\llbracket f\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x})
\end{aligned}$$

Candidate for optimization

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{lhs}}\llbracket f\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x}) &= \text{LET } ((\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}))^*\\
&\quad \text{IN}\quad (\underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{f})\,;\ (\underline{s}^*\,;\ \underline{\diamond\mathsf{y}} = \underline{e})^*\,;\\
&\qquad\qquad \underline{\diamond\mathsf{arguments}} = \ [\,(\underline{\diamond\mathsf{y}_\mathsf{i}}\,,\,)^*\,]\,;\\
&\qquad\qquad \underline{\diamond\mathsf{fun}} = \underline{\diamond\mathsf{getBase}}\,(\,\underline{f}\,)\,;\\
&\qquad\qquad \underline{x} = \underline{\diamond\mathsf{obj}}\,(\underline{\diamond\mathsf{fun}}, \underline{\diamond\mathsf{arguments}})\,,\,\underline{x})\\
\mathit{ast2ir}_{\mathit{lhs}}\llbracket\,(\,\mathit{lhs}\,.\,x\,)\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{y}) &= \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,[\,\texttt{"}x\texttt{"}\,]\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{y})\\
\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,.\,x\,((e,)^*)\,\rrbracket(\Sigma)(\underline{y}) &= \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,[\,\texttt{"}x\texttt{"}\,]\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{y})\\
\mathit{ast2ir}_{\mathit{lhs}}\llbracket\,(\,\mathit{lhs}\,[\,e'\,]\,)\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x}) &= \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,[\,e'\,]\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x})
\end{aligned}$$

Candidate for optimization

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,[\,e'\,]\ ((e,)^*)\,\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}_l^*, \underline{e}_l) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{obj}_1})\\
&\qquad\quad (\underline{s}'^*, \underline{e}') = \mathit{ast2ir}_e\llbracket e'\rrbracket(\Sigma)(\underline{\diamond\mathsf{field}_1})\\
&\qquad\quad ((\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}))^*\\
&\quad \text{IN}\quad (\underline{s}_l^*\,;\,\underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_l)\,;\,\underline{s}'^*\,;\\
&\qquad\qquad (\underline{s}^*\,;\ \underline{\diamond\mathsf{y}} = \underline{e})^*\,;\\
&\qquad\qquad \underline{\diamond\mathsf{arguments}} = \ [\,(\underline{\diamond\mathsf{y}_\mathsf{i}}\,,\,)^*\,]\,;\\
&\qquad\qquad \underline{\diamond\mathsf{fun}} = \diamond\mathsf{toObject}(\underline{\diamond\mathsf{obj}}\,[\,\underline{e}'\,]\,)\,;\\
&\qquad\qquad \underline{x} = \underline{\diamond\mathsf{fun}}\,(\underline{\diamond\mathsf{obj}}\,,\ \ \underline{\diamond\mathsf{arguments}})\,,\,\underline{x})
\end{aligned}$$

Candidate for optimization

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\,((e,)^*)\,\rrbracket(\Sigma)(\underline{x}) &= \text{LET } (\underline{s}_l^*, \underline{e}_l) = \mathit{ast2ir}_{\mathit{lhs}}\llbracket \mathit{lhs}\rrbracket(\Sigma)(\underline{\diamond\mathsf{obj}_1})\\
&\qquad\quad ((\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{\diamond\mathsf{y}}))^*\\
&\quad \text{IN}\quad (\underline{s}_l^*\,;\,\underline{\diamond\mathsf{obj}} = \diamond\mathsf{toObject}(\underline{e}_l)\,;\ (\underline{s}^*\,;\ \underline{\diamond\mathsf{y}} = \underline{e})^*\,;\\
&\qquad\qquad \underline{\diamond\mathsf{arguments}} = \ [\,(\underline{\diamond\mathsf{y}_\mathsf{i}}\,,\,)^*\,]\,;\\
&\qquad\qquad \underline{x} = \underline{\diamond\mathsf{obj}}\,(\underline{\diamond\mathsf{global}}\,,\ \ \underline{\diamond\mathsf{arguments}})\,,\,\underline{x})
\end{aligned}$$

$$\begin{aligned}
\mathit{ast2ir}_{\mathit{lit}}\llbracket\texttt{this}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \Sigma(\underline{\diamond\mathsf{this}}))\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket\texttt{null}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \mathsf{null})\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket\texttt{true}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \mathsf{true})\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket\texttt{false}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \mathsf{false})\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket \mathit{num}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \underline{\mathit{num}})\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket \mathit{str}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \underline{\mathit{str}})\\
\mathit{ast2ir}_{\mathit{lit}}\llbracket \mathit{reg}\rrbracket(\Sigma) &=
\end{aligned}$$

$$\begin{aligned}
\mathit{ast2ir}_m\llbracket \mathit{pr}\ \texttt{:}\ e\rrbracket(\Sigma)(\underline{y}) &= \text{LET } (\underline{s}^*, \underline{e}) = \mathit{ast2ir}_e\llbracket e\rrbracket(\Sigma)(\underline{y})\\
&\quad \text{IN}\quad (\underline{s}^*\,,\,\mathit{ast2ir}_{\mathit{pr}}\llbracket \mathit{pr}\rrbracket\ \texttt{:}\ \underline{e})\\
\mathit{ast2ir}_m\llbracket \mathsf{get}\ \mathit{pr}\,(\,)\ \{\mathit{fd}^*\,\mathit{vd}^*\,s^*\,\}\rrbracket(\Sigma)(\underline{x}) &= (\langle\rangle, \mathsf{get}\ \mathit{ast2ir}_{\mathit{pr}}\llbracket \mathit{pr}\rrbracket\ (\underline{\diamond\mathsf{this}}\,,\ \underline{\diamond\mathsf{arguments}})\ \{\\
&\qquad\qquad (\mathit{ast2ir}_{\mathit{fd}}\llbracket \mathit{fd}\rrbracket(\Sigma))^*\\
&\qquad\qquad (\mathit{ast2ir}_{\mathit{vd}}\llbracket \mathit{vd}\rrbracket(\Sigma))^*\\
&\qquad\qquad (\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{this}}; \underline{\diamond\mathsf{arguments}}))^*\,\}\,)\\
\mathit{ast2ir}_m\llbracket \mathsf{set}\ \mathit{pr}\,(\,x\,)\ \{\mathit{fd}^*\,\mathit{vd}^*\,s^*\,\}\rrbracket(\Sigma)(\underline{y}) &= (\langle\rangle, \mathsf{set}\ \mathit{ast2ir}_{\mathit{pr}}\llbracket \mathit{pr}\rrbracket\ (\underline{\diamond\mathsf{this}}\,,\ \underline{\diamond\mathsf{arguments}})\ \{\\
&\qquad\qquad (\mathit{ast2ir}_{\mathit{fd}}\llbracket \mathit{fd}\rrbracket(\Sigma))^*\\
&\qquad\qquad \mathsf{var}\ \underline{x}\\
&\qquad\qquad (\mathit{ast2ir}_{\mathit{vd}}\llbracket \mathit{vd}\rrbracket(\Sigma))^*\\
&\qquad\qquad \underline{x} = \underline{\diamond\mathsf{arguments}}\,[\,\texttt{"0"}\,]\,;\qquad \textit{where } \underline{x} \textit{ is not the name of any of fd}\\
&\qquad\qquad (\mathit{ast2ir}_s\llbracket s\rrbracket(\Sigma; \underline{\diamond\mathsf{this}}; \underline{\diamond\mathsf{arguments}}))^*\,\}\,)
\end{aligned}$$

Chapter 5

IR Semantics

- Environments in the semantics are references.

5.1 Domains

$$\begin{array}{rclcl}
b & \in & \mathit{Bool} & ::= & \texttt{true} \mid \texttt{false} \\
n & \in & \mathit{Num} & ::= & \texttt{NaN} \mid \texttt{Infinity} \mid \texttt{0} \mid \texttt{1} \mid \ldots \\
s & \in & \mathit{Str} & ::= & \text{“}\texttt{foo}\text{”} \mid \text{“}\texttt{bar}\text{”} \mid \ldots \\
x, y, z & \in & \mathit{Var} & ::= & \texttt{this} \mid \texttt{foo} \mid \texttt{bar} \mid \ldots \\
p & \in & \mathit{PName} & = & \mathit{Str} \cup \mathit{Var} \\
pv & \in & \mathit{PVal} & = & \{\texttt{undefined},\ \texttt{null}\} \cup \mathit{Bool} \cup \mathit{Num} \cup \mathit{Str} \\
l & \in & \mathit{Loc} & ::= & \#\texttt{Global} \mid \#\texttt{ObjProto} \mid \#\texttt{FtnProto} \mid \#\texttt{ArrProto} \mid \#\texttt{StrProto} \mid \#\texttt{BoolProto} \mid \#\texttt{NumProto} \\
 & & & \mid & \#\texttt{Null} \mid l_1 \mid \ldots \\
v & \in & \mathit{Val} & = & \mathit{Loc} \cup \mathit{PVal} \\
fv & \in & \mathit{FVal} & ::= & \textsf{function}\ \underline{f}\ (\underline{\mathit{this}}, \underline{\mathit{arguments}})\ \ \{\underline{s}\} \ \mid\ \textsf{get}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \ \mid\ \textsf{set}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}
\end{array}$$

4.2 Language Overview: `Error`, `EvalError`, `RangeError`, `ReferenceError`, `SyntaxError`, `TypeError` and `URIError`

$$\begin{array}{rclcl}
err & \in & \mathit{Error} & = & \{\texttt{Error}, \texttt{EvalError}, \texttt{RangeError}, \texttt{ReferenceError}, \texttt{SyntaxError}, \texttt{TypeError}, \texttt{URIError}\} \\
ve & \in & \mathit{ValError} & = & \mathit{Val} \cup \mathit{Error}
\end{array}$$

8.6.2 Object Internal Properties and Methods: Table 8–Internal Properties Common to All Objects
`[[Prototype]]`, `[[Class]]`, `[[Extensible]]`, `[[Get]]`, `[[GetOwnProperty]]`, `[[GetProperty]]`, `[[Put]]`, `[[CanPut]]`, `[[HasProperty]]`, `[[Delete]]`, `[[DefaultValue]]`, `[[DefineOwnProperty]]`

$$\begin{array}{rclcl}
o & \in & \mathit{Object} & ::= & \{[[\texttt{Class}]] : \mathit{Str}, \\
 & & & & \ [[\texttt{Extensible}]] : \mathit{Bool}, \\
 & & & & \ [[\texttt{Prototype}]] : \mathit{Loc}, \\
 & & & & \ @\texttt{property} : \mathit{PName} \mapsto \mathit{ObjectValue} \\
 & & & & \ (, [[\texttt{Code}]] : \mathit{FVal}, \\
 & & & & \ \ \ [[\texttt{Scope}]] : \mathit{Env})^{?}\}
\end{array}$$

8.6.1 Property Attributes

$$\begin{array}{rclcl}
ov & \in & \mathit{ObjectValue} & = & \mathit{DataProp} \cup \mathit{AccessorProp}
\end{array}$$

Table 5–Attributes of a Named Data Property: `[[Value]]`, `[[Writable]]`, `[[Enumerable]]`, `[[Configurable]]`

$$\begin{array}{rclcl}
dp & \in & \mathit{DataProp} & ::= & \{[[\texttt{Value}]] : \mathit{Val}, \\
 & & & & \ [[\texttt{Writable}]] : \mathit{Bool}, \\
 & & & & \ [[\texttt{Enumerable}]] : \mathit{Bool}, \\
 & & & & \ [[\texttt{Configurable}]] : \mathit{Bool}\}
\end{array}$$

Table 6–Attributes of a Named Accessor Property: `[[Get]]`, `[[Set]]`, `[[Enumerable]]`, `[[Configurable]]`

$$\begin{array}{rclcl}
ap & \in & \mathit{AccessorProp} & ::= & \{[[\texttt{Get}]] : \mathit{Val}, \\
 & & & & \ [[\texttt{Set}]] : \mathit{Val}, \\
 & & & & \ [[\texttt{Enumerable}]] : \mathit{Bool}, \\
 & & & & \ [[\texttt{Configurable}]] : \mathit{Bool}\} \\
sv & \in & \mathit{StoreValue} & ::= & \{[[\texttt{Value}]] : \mathit{ValError} \cup \{\bot\},\ [[\texttt{Mutable}]] : \mathit{Bool}, [[\texttt{Configurable}]] : \mathit{Bool}\} \\
H, K & \in & \mathit{Heap} & = & \mathit{Loc} \overset{\text{fin}}{\to} \mathit{Object} \\
 & & & & \#\texttt{Null} \notin \mathit{Dom}(H)
\end{array}$$

10.2 Lexical Environments: A Lexical Environment consists of an Environment Record and a possibly null reference to an outer Lexical Environment.

$$\begin{array}{rclcl} A, B & \in & \mathit{Env} & ::= & \texttt{\#Global} \mid \mathit{er} :: A \end{array}$$

10.2.1 Environment Records: An environment record is either a declarative environment record or an object environment record.

$$\begin{array}{rclcl} \mathit{er} & \in & \mathit{EnvRec} & = & \mathit{DeclEnvRec} \cup \mathit{ObjEnvRec} \end{array}$$

10.2.1.1 Declarative Environment Records: A declarative environment record binds the set of identifiers defined by the declarations contained within its scope.

$$\begin{array}{rclcl} \sigma & \in & \mathit{DeclEnvRec} & = & \mathit{Var} \overset{\mathsf{fin}}{\rightarrow} \mathit{StoreValue} \end{array}$$

10.2.1.2 Object Environment Records: Each object environment record is associated with an object called its binding object.

$$\begin{array}{rclcl} l & \in & \mathit{ObjEnvRec} & = & \mathit{Loc} \\ \mathit{bs} & \in & \mathit{EnvRec} \cup \mathit{Loc} & & \\ \mathit{bv} & \in & \mathit{BindingValue} & = & \mathit{StoreValue} \cup \mathit{ObjectValue} \\ \mathit{tb} & \in & \mathit{ThisBinding} & = & \mathit{Loc} \\ (H, A, \mathit{tb}) & \in & \mathit{State} & = & \mathit{Heap} \times \mathit{Env} \times \mathit{ThisBinding} \end{array}$$

8.9 The Completion Specification Type

$$\begin{array}{rclcl} \mathit{ct} & \in & \mathit{Completion} & ::= & \mathit{nc} \mid \mathit{ac} \\ \mathit{vt} & \in & \mathit{Val} \cup \{\texttt{empty}\} & & \\ \mathit{nc} & \in & \mathit{NormalCompletion} & ::= & \texttt{Normal}(\mathit{vt}) \end{array}$$

The term “abrupt completion” referes to any completion with a type other than normal.

$$\begin{array}{rclcl} \mathit{ac} & \in & \mathit{AbruptCompletion} & ::= & \texttt{Break}(\mathit{vt}, x) \mid \texttt{Return}(\mathit{vt}) \mid \texttt{Throw}(\mathit{ve}) \end{array}$$

5.2 Our Own Helpers

$$\mathit{UndefVB} = (\{[[\texttt{Value}]]:\texttt{undefined}, [[\texttt{Writable}]]:\texttt{false}, [[\texttt{Enumerable}]]:\texttt{false}, [[\texttt{Configurable}]]:\texttt{false}\}, \texttt{\#Null})$$

$$\begin{array}{ll} \mathit{isIndex} & : \mathit{Str} \rightarrow \mathit{Bool} \\ \mathit{isIndex}(s) & = \begin{cases} \texttt{false} & \text{if } s \neq \mathit{ToString}(\mathit{ToUint32}(s)) \\ \texttt{true} & \text{if } s = \mathit{ToString}(\mathit{ToUint32}(s)) \end{cases} \end{array}$$

$$\begin{array}{ll} \mathit{NewLoc} & : () \rightarrow \mathit{Loc} \\ \mathit{NewLoc}() & = l_{\mathit{new}} \end{array}$$

$$\begin{array}{ll} \mathit{Inherit} & : \mathit{Heap} \times \mathit{Loc} \times \mathit{Loc} \rightarrow \mathit{Bool} \\ \mathit{Inherit}(H, l_1, l_2) & = \begin{cases} \texttt{false} & \text{if } l_1 = \texttt{\#Null} \\ \texttt{true} & \text{if } l_1 \neq \texttt{\#Null} \wedge l_1 = l_2 \\ \mathit{Inherit}(H, H(l_1).[[\texttt{Prototype}]], l_2) & \text{if } l_1 \neq \texttt{\#Null} \wedge l_1 \neq l_2 \end{cases} \end{array}$$

11.4.3 The `typeof` Operator

$$\begin{array}{ll} \mathit{TypeTag} & : \mathit{Heap} \times \mathit{Val} \rightarrow \mathit{Str} \\ \mathit{TypeTag}(H, v) & = \begin{cases} \texttt{"undefined"} & \text{if } v = \texttt{undefined} \\ \texttt{"object"} & \text{if } v = \texttt{null} \\ \texttt{"boolean"} & \text{if } v \in \mathit{Bool} \\ \texttt{"number"} & \text{if } v \in \mathit{Num} \\ \texttt{"string"} & \text{if } v \in \mathit{Str} \\ \texttt{"object"} & \text{if } v \in \mathit{Loc} \wedge \neg \mathit{IsCallable}(H, v) \\ \texttt{"function"} & \text{if } v \in \mathit{Loc} \wedge \mathit{IsCallable}(H, v) \end{cases} \end{array}$$

10.4.3 Entering Function Code

$$\begin{array}{ll} \mathit{GetThis} & : \mathit{Heap} \times \mathit{Val} \rightarrow \mathit{Heap} \times \mathit{Loc} \\ \mathit{GetThis}(H, v) & = \begin{cases} (H, \texttt{\#Global}) & \text{if } v = \texttt{undefined} \vee v = \texttt{null} \\ \mathit{ToObject}(H, v) & \text{if } v \in \mathit{Bool} \cup \mathit{Num} \cup \mathit{Str} \\ (H, v) & \text{if } v \in \mathit{Loc} \end{cases} \end{array}$$

$$\begin{array}{ll} \mathit{ParamsSize} & : \mathit{FVal} \rightarrow \mathit{Num} \\ \mathit{ParamsSize}(\mathit{fv}) & = |\underline{s_{\mathit{params}}}| \quad \text{where } \mathit{fv} = \_\ \underline{f}\ (\underline{\mathit{this}}, \underline{\mathit{arguments}})\ \ \{\underline{s_{\mathit{params}}}\ \underline{s_{\mathit{vds}}}\ \underline{s_{\mathit{fds}}}\ \underline{s_{\mathit{stmts}}}\} \end{array}$$

$$\begin{array}{ll} \mathit{GetBody} & : \mathit{FVal} \rightarrow \mathit{Stmt} \\ \mathit{GetBody}(\mathit{fv}) & = \underline{s} \quad \text{where } \mathit{fv} = \_\ \underline{f}\ (\underline{\mathit{this}}, \underline{\mathit{arguments}})\ \ \{\underline{s}\} \end{array}$$

15.4 Array Objects

$$\mathit{IsArrayIndex} \quad : \mathit{Val} \rightarrow \mathit{Bool}$$

$$\mathit{IsArrayIndex}(v) = \begin{cases} \texttt{true} & \text{if } \mathit{ToString}(\mathit{ToUnit32}(\mathit{ToString}(v))) = \mathit{ToString}(v) \wedge \mathit{ToUnit32}(\mathit{ToString}(v)) \neq 2^{32}-1 \\ \texttt{false} & \text{otherwise} \end{cases}$$

15.9 Date Objects

$$\mathit{IsDate} \quad : \mathit{Heap} \times \mathit{Val} \rightarrow \mathit{Bool}$$

$$\mathit{IsDate}(H, v) = \begin{cases} \texttt{true} & \text{if } v \in \mathit{Loc} \wedge H(v).[[\texttt{Class}]] = \text{“}\texttt{Date}\text{”} \\ \texttt{false} & \text{otherwise} \end{cases}$$

12.6.4 The `for-in` Statement

$$\mathit{IteratorInit} \quad : \wp(\mathit{PName}) \rightarrow \mathit{Object}$$

$$\mathit{IteratorInit}(P) = \mathit{NewObj}().@\texttt{property}[\text{“}\texttt{length}\text{”} \mapsto n,\ \text{“}\texttt{@i}\text{”} \mapsto 0,\ \text{“}\texttt{0}\text{”} \mapsto pn_0, \ldots \text{“}\texttt{n}-\texttt{1}\text{”} \mapsto pn_{n-1}]$$
$$\text{where } P = \{pn_0, \ldots, pn_{n-1}\}$$

$$\mathit{CollectProps}(H, l) = \begin{cases} \mathit{Dom}(H(l).@\texttt{property}) \cup \mathit{CollectProps}(H, H(l).[[\texttt{Prototype}]]) & \text{if } l \in \mathit{Dom}(H) \\ \{\} & \text{if } l \notin \mathit{Dom}(H) \end{cases}$$

$$\mathit{IsEnumerable} \quad : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \rightarrow \mathit{Bool}$$

$$\mathit{IsEnumerable}(H, l, x) = \begin{cases} H(l).@\texttt{property}(x).[[\texttt{Enumerable}]] & \text{if } l \in \mathit{Dom}(H) \wedge x \in \mathit{Dom}(H(l)) \\ \mathit{IsEnumerable}(H, H(l).[[\texttt{Prototype}]], x) & \text{if } l \in \mathit{Dom}(H) \wedge x \notin \mathit{Dom}(H(l)) \\ \texttt{false} & \text{if } l \notin \mathit{Dom}(H) \end{cases}$$

$$\mathit{Next} \quad : \mathit{Heap} \times \mathit{Object} \times \mathit{Num} \times \mathit{Loc} \rightarrow \mathit{Num}$$

$$\mathit{Next}(H, o, n, l) = \begin{cases} n & \text{if } n \notin \mathit{Dom}(o) \wedge n \geq o.@\texttt{property}(\text{“}\texttt{length}\text{”}) \\ \mathit{Next}(H, o, n+1, l) & \text{if } n \notin \mathit{Dom}(o) \wedge n < o.@\texttt{property}(\text{“}\texttt{length}\text{”}) \\ n & \text{if } n \in \mathit{Dom}(o) \wedge \mathit{IsEnumerable}(H, l, o.@\texttt{property}(n)) \\ \mathit{Next}(H, o, n+1, l) & \text{if } n \in \mathit{Dom}(o) \wedge \neg \mathit{IsEnumerable}(H, l, o.@\texttt{property}(n)) \end{cases}$$

$$\mathit{Negate} \quad : \mathit{Num} \rightarrow \mathit{Num}$$

$$\mathit{Negate}(n) = \begin{cases} \texttt{NaN} & \text{if } n = \texttt{NaN} \\ 0 - n & \text{otherwise} \end{cases}$$

$$\mathit{Negate} \quad : \mathit{Bool} \rightarrow \mathit{Bool}$$

$$\mathit{Negate}(b) = \begin{cases} \texttt{false} & \text{if } b = \texttt{true} \\ \texttt{true} & \text{otherwise} \end{cases}$$

$$\mathit{ExnLoc} \quad : \mathit{ValError} \rightarrow \mathit{Val}$$

$$\mathit{ExnLoc}(\mathit{ve}) = \begin{cases} \#\texttt{Error} & \text{if } \mathit{ve} = \texttt{Error} \\ \#\texttt{EvalError} & \text{if } \mathit{ve} = \texttt{EvalError} \\ \#\texttt{RangeError} & \text{if } \mathit{ve} = \texttt{RangeError} \\ \#\texttt{ReferenceError} & \text{if } \mathit{ve} = \texttt{ReferenceError} \\ \#\texttt{SyntaxError} & \text{if } \mathit{ve} = \texttt{SyntaxError} \\ \#\texttt{TypeError} & \text{if } \mathit{ve} = \texttt{TypeError} \\ \#\texttt{URIError} & \text{if } \mathit{ve} = \texttt{URIError} \\ v & \text{if } \mathit{ve} \in \mathit{Val} \end{cases}$$

5.3 Helpers from the Specification

8.7 The Reference Specification Type

8.7.2 `PutValue (V, W)`

$$\mathit{PutValue} \quad : \mathit{Heap} \times \mathit{Env} \times \mathit{Var} \times \mathit{Val} \times \texttt{strict} \rightarrow \mathit{Heap} \times \mathit{Env} \times \mathit{ValError}$$

$$\mathit{PutValue}(H, A, x, v, b) = \begin{cases} (H, A, \texttt{ReferenceError}) & \text{if } \mathit{Lookup}(H, A, x, \texttt{strict}) = l \wedge l = \#\texttt{Null} \wedge b \\ \mathit{Put}(H, A, \#\texttt{Global}, x, v, \texttt{false}) & \text{if } \mathit{Lookup}(H, A, x, \texttt{strict}) = l \wedge l = \#\texttt{Null} \wedge \neg b \\ \mathit{Put}(H, A, l, x, v, b) & \text{if } \mathit{Lookup}(H, A, x, \texttt{strict}) = l \wedge l \neq \#\texttt{Null} \\ \mathit{SetBindingDER}(H, A, x, v, b) & \text{if } \mathit{Lookup}(H, A, x, \texttt{strict}) = \sigma \\ & \text{For primitive base values, see 8.7.2.} \end{cases}$$

8.12.1 `[[GetOwnProperty]](P)`: Let X be O's own property named P. $x \in \mathit{Dom}(H(l))$

$$\mathit{Dom}(o) = \{\ x \mid x \mapsto \mathit{ov} \in o.@\texttt{property}\ \}$$

8.12.1 `[[GetOwnProperty]](P)`: A `String` object has a more elaborate `[[GetOwnProperty]]` internal method (15.5.5.2).

$$\mathit{GetOwnProperty} : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \rightarrow \mathit{ObjectValue} \times \mathit{Loc}$$

$$\mathit{GetOwnProperty}(H, l, x) = \begin{cases} \mathit{UndefVB} & \text{if } l \notin \mathit{Dom}(H) \\ \mathit{UndefVB} & \text{if } l \in \mathit{Dom}(H) \wedge x \notin \mathit{Dom}(H(l)) \\ (\mathit{copy}(H(l).@\texttt{property}(x)), l) & \text{if } l \in \mathit{Dom}(H) \wedge x \in \mathit{Dom}(H(l)) \end{cases}$$

8.12 Algorithms for Object Internal Methods

8.12.2 `[[GetProperty]](P)`

$$\mathit{GetProperty} : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \rightarrow \mathit{ObjectValue} \times \mathit{Loc}$$

$$\mathit{GetProperty}(H, l, x) = \begin{cases} \mathit{UndefVB} & \text{if } l \notin \mathit{Dom}(H) \\ \mathit{UndefVB} & \text{if } l \in \mathit{Dom}(H) \wedge x \notin \mathit{Dom}(H(l)) \wedge H(l).[[\texttt{Prototype}]] = \#\texttt{Null} \\ \mathit{GetProperty}(H, H(l).[[\texttt{Prototype}]], x) & \text{if } l \in \mathit{Dom}(H) \wedge x \notin \mathit{Dom}(H(l)) \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \\ \mathit{GetOwnProperty}(H, l, x) & \text{if } l \in \mathit{Dom}(H) \wedge x \in \mathit{Dom}(H(l)) \end{cases}$$

8.12.3 `[[Get]](P)` $H(l).[[\texttt{Get}]](\texttt{P})$

$$\mathit{Get} : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \rightarrow \mathit{Val}$$

$$\mathit{Get}(H, l, x) = \begin{cases} \texttt{undefined} & \text{if } \mathit{GetProperty}(H, l, x) = \mathit{UndefVB} \\ \mathit{dp}.[[\texttt{Value}]] & \text{if } \mathit{GetProperty}(H, l, x) = (\mathit{dp}, \_) \\ \texttt{undefined} & \text{if } \mathit{GetProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Get}]] = \texttt{undefined} \\ \mathit{ap}.[[\texttt{Get}]].[[\texttt{Call}]](H(l), []) & \text{if } \mathit{GetProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Get}]] \neq \texttt{undefined} \end{cases}$$

8.12.4 `[[CanPut]](P)`

$$\mathit{CanPut} : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \rightarrow \mathit{Bool}$$

$$\mathit{CanPut}(H, l, x) = \begin{cases} \texttt{false} & \text{if } \mathit{GetOwnProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Set}]] = \texttt{undefined} \\ \texttt{true} & \text{if } \mathit{GetOwnProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Set}]] \neq \texttt{undefined} \\ \mathit{dp}.[[\texttt{Writable}]] & \text{if } \mathit{GetOwnProperty}(H, l, x) = (\mathit{dp}, \_) \\ H(l).[[\texttt{Extensible}]] & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] = \#\texttt{Null} \\ H(l).[[\texttt{Extensible}]] & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = \mathit{UndefVB} \\ \texttt{false} & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Set}]] = \texttt{undefined} \\ \texttt{true} & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{ap}, \_) \wedge \mathit{ap}.[[\texttt{Set}]] \neq \texttt{undefined} \\ \texttt{false} & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{dp}, \_) \wedge \neg H(l).[[\texttt{Extensible}]] \\ \mathit{dp}.[[\texttt{Writable}]] & \text{if } \mathit{GetOwnProperty}(H, l, x) = \mathit{UndefVB} \wedge H(l).[[\texttt{Prototype}]] \neq \#\texttt{Null} \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{dp}, \_) \wedge H(l).[[\texttt{Extensible}]] \end{cases}$$

8.12.5 `[[Put]](P,V,Throw)`

$$\mathit{Put} : \mathit{Heap} \times \mathit{Env} \times \mathit{Loc} \times \mathit{PName} \times \mathit{Val} \times \mathit{Bool} \rightarrow \mathit{Heap} \times \mathit{Env} \times \mathit{ValError}$$

$$\mathit{Put}(H, A, l, x, v, b) = \begin{cases} (H, A, \texttt{TypeError}) & \text{if } \neg\mathit{CanPut}(H, l, x) \wedge b \\ (H, A, v) & \text{if } \neg\mathit{CanPut}(H, l, x) \wedge \neg b \\ \mathit{DefineOwnProperty}(H, A, l, x, \mathit{dp}', b) & \text{if } \mathit{CanPut}(H, l, x) \wedge \\ & \quad \mathit{GetOwnProperty}(H, l, x) = (\mathit{dp}, \_) \neq \mathit{UndefVB} \wedge \\ & \quad \mathit{dp}' = \{[[\texttt{Value}]] : v\} \\ \mathit{ap}.[[\texttt{Set}]].[[\texttt{Call}]](H(l), v) & \text{if } \mathit{CanPut}(H, l, x) \wedge \mathit{GetOwnProperty}(H, l, x) \neq (\mathit{dp}, \_) \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{ap}, \_) \\ \mathit{DefineOwnProperty}(H, A, l, x, \mathit{dp}', b) & \text{if } \mathit{CanPut}(H, l, x) \wedge \mathit{GetOwnProperty}(H, l, x) \neq (\mathit{dp}, \_) \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\mathit{dp}, \_) \wedge \\ & \quad \mathit{dp}' = \{[[\texttt{Value}]] : v, [[\texttt{Writable}]] : \texttt{true}, \\ & \qquad [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\} \end{cases}$$

8.12.6 `[[HasProperty]](P)`

$$\begin{array}{ll}
\mathit{HasProperty} & : \mathit{Heap} \times \mathit{Loc} \times \mathit{PName} \to \mathit{Bool} \\
\mathit{HasProperty}(H, l, x) & = \begin{cases} \texttt{false} & \text{if } \mathit{GetProperty}(H, l, x) = \mathit{UndefVB} \\ \texttt{true} & \text{if } \mathit{GetProperty}(H, l, x) \neq \mathit{UndefVB} \end{cases}
\end{array}$$

10.2.1.1.5 `DeleteBinding(N)`
10.2.1.2.5 `DeleteBinding(N)`

$$\begin{array}{ll}
\mathit{DeleteBinding} & : \mathit{Heap} \times \mathit{Env} \times \mathit{Str} \times \texttt{strict} \to \mathit{Heap} \times \mathit{Env} \times (\mathit{Bool} \cup \mathit{Error}) \\
\mathit{DeleteBinding}(H, A, s, b) & = \begin{cases} (H, A, \texttt{true}) & \text{if } A = \#\texttt{Global} \wedge s \notin \mathit{Dom}(H(\#\mathit{Global})) \\ \mathit{Delete}(H, A, \#\texttt{Global}, s, b) & \text{if } A = \#\texttt{Global} \wedge s \in \mathit{Dom}(H(\#\mathit{Global})) \\ (H, (\sigma - s) :: A', \texttt{true}) & \text{if } A = \sigma :: A' \wedge s \in \mathit{Dom}(\sigma) \wedge \sigma(s).[[\texttt{Configurable}]] \\ (H, A, \texttt{false}) & \text{if } A = \sigma :: A' \wedge s \in \mathit{Dom}(\sigma) \wedge \neg\sigma(s).[[\texttt{Configurable}]] \\ (H', \sigma :: A'', \mathit{ve}) & \text{if } A = \sigma :: A' \wedge s \notin \mathit{Dom}(\sigma) \wedge \\ & \quad \mathit{DeleteBinding}(\mathrm{H, A', s, b}) = (\mathrm{H', A''}, \mathit{ve}) \\ \mathit{Delete}(H, A, l, s, b) & \text{if } A = l :: A' \wedge s \in \mathit{Dom}(H(l)) \\ (H', l :: A'', \mathit{ve}) & \text{if } A = l :: A' \wedge s \notin \mathit{Dom}(H(l)) \wedge \\ & \quad \mathit{DeleteBinding}(H, A', s, b) = (H', A'', \mathit{ve}) \end{cases}
\end{array}$$

8.12.7 `[[Delete]](P, Throw)`

$$\begin{array}{ll}
\mathit{Delete} & : \mathit{Heap} \times \mathit{Env} \times \mathit{Loc} \times \mathit{Str} \times \mathit{Bool} \to \mathit{Heap} \times \mathit{Env} \times \mathit{ValError} \\
\mathit{Delete}(H, A, l, s, b) & = \begin{cases} (H, A, \texttt{true}) & \text{if } \mathit{GetOwnProperty}(H, l, s) = \mathit{UndefVB} \\ (H', A, \texttt{true}) & \text{if } \mathit{GetOwnProperty}(H, l, s) = (\mathit{ov}, \_) \wedge \\ & \quad \mathit{ov}.[[\texttt{Configurable}]] \wedge \\ & \quad H' = H[l \mapsto H(l) - s] \\ (H, A, \texttt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H, l, s) = (\mathit{ov}, \_) \wedge \neg\mathit{ov}.[[\texttt{Configurable}]] \wedge b \\ (H, A, \texttt{false}) & \text{otherwise} \end{cases}
\end{array}$$

Less precise but simpler!

8.12.8 `[[DefaultValue]](hint)`

$$\begin{array}{ll}
\mathit{DefaultValue} & : \mathit{Heap} \times \mathit{Loc} \times \mathit{Str} \to \mathit{PVal} \cup \mathit{Error} \\
\mathit{DefaultValue}(H, l, \texttt{String}) & = \begin{cases} s & \text{if } ``H(l)" = s \wedge s \in \mathit{PVal} \\ v & \text{if } ``H(l)" \notin \mathit{PVal} \wedge \texttt{valueOf}(H(l)) = v \wedge v \in \mathit{PVal} \\ \texttt{TypeError} & \text{otherwise} \end{cases} \\
\mathit{DefaultValue}(H, l, \texttt{Number}) & = \begin{cases} v & \text{if } \texttt{valueOf}(H(l)) = v \wedge v \in \mathit{PVal} \\ s & \text{if } \texttt{valueOf}(H(l)) \notin \mathit{PVal} \wedge ``H(l)" = s \wedge s \in \mathit{PVal} \\ \texttt{TypeError} & \text{otherwise} \end{cases}
\end{array}$$

Precise but too complicated!

8.12.8 `[[DefaultValue]](hint)`

$$\begin{array}{ll}
\mathit{DefaultValue} & : \mathit{Heap} \times \mathit{Loc} \times \mathit{Str} \to \mathit{PVal} \cup \mathit{Error} \\
\mathit{DefaultValue}(H, l, \texttt{String}) & = \begin{cases} s & \text{if } \mathit{Get}(H, l, \texttt{toString}) = v \wedge \mathit{IsCallable}(H, v) \wedge v.[[\texttt{Call}]](H(l), []) = s \wedge s \in \mathit{PVal} \\ v'' & \text{if } \mathit{Get}(H, l, \texttt{toString}) = v \wedge (\neg\mathit{IsCallable}(H, v) \vee v.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \wedge \\ & \quad \mathit{Get}(H, l, \texttt{valueOf}) = v' \wedge \mathit{IsCallable}(H, v') \wedge v'.[[\texttt{Call}]](H(l), []) = v'' \wedge v'' \in \mathit{PVal} \\ \texttt{TypeError} & \text{if } \mathit{Get}(H, l, \texttt{toString}) = v \wedge (\neg\mathit{IsCallable}(H, v) \vee v.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \wedge \\ & \quad \mathit{Get}(H, l, \texttt{valueOf}) = v' \wedge (\neg\mathit{IsCallable}(H, v') \vee v'.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \end{cases} \\
\mathit{DefaultValue}(H, l, \texttt{Number}) & = \begin{cases} v' & \text{if } \mathit{Get}(H, l, \texttt{valueOf}) = v \wedge \mathit{IsCallable}(H, v) \wedge v.[[\texttt{Call}]](H(l), []) = v' \wedge v' \in \mathit{PVal} \\ s & \text{if } \mathit{Get}(H, l, \texttt{valueOf}) = v \wedge (\neg\mathit{IsCallable}(H, v) \vee v.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \wedge \\ & \quad \mathit{Get}(H, l, \texttt{toString}) = v' \wedge \mathit{IsCallable}(H, v') \wedge v'.[[\texttt{Call}]](H(l), []) = s \wedge s \in \mathit{PVal} \\ \texttt{TypeError} & \text{if } \mathit{Get}(H, l, \texttt{valueOf}) = v \wedge (\neg\mathit{IsCallable}(H, v) \vee v.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \wedge \\ & \quad \mathit{Get}(H, l, \texttt{toString}) = v' \wedge (\neg\mathit{IsCallable}(H, v') \vee v'.[[\texttt{Call}]](H(l), []) \notin \mathit{PVal}) \end{cases}
\end{array}$$

8.12.9 `[[DefineOwnProperty]](P,Desc,Throw)`

$\mathit{DefineOwnProperty} \quad : \mathit{Heap} \times \mathit{Env} \times \mathit{Loc} \times \mathit{Var} \times \mathit{ObjectValue} \times \mathit{Bool} \to \mathit{Heap} \times \mathit{Env} \times \mathit{ValError}$

$\mathit{DefineOwnProperty}(H, A, l, x, ov, b) =$

$$
\begin{cases}
\text{Step 3} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = \mathit{UndefVB} \wedge \neg H(l).[[\mathtt{Extensible}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = \mathit{UndefVB} \wedge \neg H(l).[[\mathtt{Extensible}]] \wedge \neg b \\
\text{Step 4} & \\
(H[l \mapsto H(l).\mathtt{@property}[x \mapsto \mathit{copy}(ov)]], & \text{if } \mathit{GetOwnProperty}(H,l,x) = \mathit{UndefVB} \wedge H(l).[[\mathtt{Extensible}]] \\
\quad A, \mathtt{true}) & \\
\text{Steps 5\&6} & \\
(H, A, \mathtt{true}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge (ov = \emptyset \vee ov \subseteq ov') \\
\text{Step 7-a} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge \neg ov'.[[\mathtt{Configurable}]] \wedge \\
& \quad ov.[[\mathtt{Configurable}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge \neg ov'.[[\mathtt{Configurable}]] \wedge \\
& \quad ov.[[\mathtt{Configurable}]] \wedge \neg b \\
\text{Step 7-b} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge \neg ov'.[[\mathtt{Configurable}]] \wedge \\
& \quad ov.[[\mathtt{Enumerable}]] \neq ov'.[[\mathtt{Enumerable}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge \neg ov'.[[\mathtt{Configurable}]] \wedge \\
& \quad ov.[[\mathtt{Enumerable}]] \neq ov'.[[\mathtt{Enumerable}]] \wedge \neg b \\
\text{Step 9-a} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg b \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg b \\
\text{Step 9-b-i} & \\
(H[l \mapsto H(l).\mathtt{@property}[x \mapsto \mathit{copy}(ov)]], & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{DataProp} \\
\quad A, \mathtt{true}) & \\
\text{Step 9-c-i} & \\
(H[l \mapsto H(l).\mathtt{@property}[x \mapsto \mathit{copy}(ov)]], & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{AccProp} \\
\quad A, \mathtt{true}) & \\
\text{Step 10-a-i} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg ov'.[[\mathtt{Writable}]] \wedge ov.[[\mathtt{Writable}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg ov'.[[\mathtt{Writable}]] \wedge ov.[[\mathtt{Writable}]] \neg \wedge b \\
\text{Step 10-a-ii} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg ov'.[[\mathtt{Writable}]] \wedge ov.[[\mathtt{Value}]] \neq ov'.[[\mathtt{Value}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{DataProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge \neg ov'.[[\mathtt{Writable}]] \wedge ov.[[\mathtt{Value}]] \neq ov'.[[\mathtt{Value}]] \wedge \neg b \\
\text{Step 10-b} & \\
(H[l \mapsto H(l).\mathtt{@property}[x \mapsto \mathit{copy}(ov)]], & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{DataProp} \wedge ov' \in \mathit{DataProp} \wedge \\
\quad A, \mathtt{true}) & \\
& \quad ov'.[[\mathtt{Configurable}]] \\
\text{Step 11-a-i} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge ov'.[[\mathtt{Set}]] \neq ov.[[\mathtt{Set}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge ov'.[[\mathtt{Set}]] \neq ov.[[\mathtt{Set}]] \wedge \neg b \\
\text{Step 11-a-ii} & \\
(H, A, \mathtt{TypeError}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge ov'.[[\mathtt{Get}]] \neq ov.[[\mathtt{Get}]] \wedge b \\
(H, A, \mathtt{false}) & \text{if } \mathit{GetOwnProperty}(H,l,x) = (ov', _) \wedge ov \in \mathit{AccProp} \wedge ov' \in \mathit{AccProp} \wedge \\
& \quad \neg ov'.[[\mathtt{Configurable}]] \wedge ov'.[[\mathtt{Get}]] \neq ov.[[\mathtt{Get}]] \wedge \neg b \\
\text{Step 12} & \\
(H[l \mapsto H(l).\mathtt{@property}[x \mapsto \mathit{copy}(ov)]], & \text{otherwise} \\
\quad A, \mathtt{true}) & \\
& \text{For } \mathtt{Array} \text{ objects, see 15.4.5.1.}
\end{cases}
$$

$$
\begin{array}{ll}
\mathit{DeleteArray} : \mathit{Heap} \times \mathit{Env} \times \mathit{Loc} \times \mathit{Num} \times \mathit{Num} \times \mathit{ObjectValue} \times \mathit{Bool} \times \mathit{Bool} \rightarrow \mathit{Heap} \times \mathit{Env} \times (\mathit{Bool} \cup \mathit{Error}) & \\
\mathit{ov}' & = \mathit{ov}.\{[[\mathtt{Value}]] \mapsto \mathit{oldLen}\} \\
\mathit{ov}'' & = \mathit{ov}'.\{[[\mathtt{Writable}]] \mapsto \mathtt{false}\} \\
\mathit{DeleteArray}(H, A, l, \mathit{newLen}, \mathit{oldLen}, \mathit{ov}, b, b') \ = &
\end{array}
$$

$$
\left\{
\begin{array}{ll}
\text{Step 3.l} & \\
(H, A, \mathtt{true}) & \text{if } \ \mathit{newLen} \geq \mathit{oldLen} \\
\text{Step 3.l.ii} & \\
(H, A, \mathit{err}) & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathit{err}) \\
\text{Step 3.l.iii / writable = true / DefineOwnProperty = err} & \\
(H, A, \mathit{err}) & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathtt{fals} \\
& \quad \wedge b = \mathtt{true} \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\mathtt{length}\text{”}, \mathit{ov}', \mathtt{false}) = (H'', A'', \mathit{err}) \\
\text{Step 3.l.iii / writable = true / DefineOwnProperty} \neq \text{err} & \\
(H, A, \text{if } b' \ \mathtt{TypeError} \text{ else } \mathtt{false}) & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathtt{fals} \\
& \quad \wedge b = \mathtt{true} \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\mathtt{length}\text{”}, \mathit{ov}', \mathtt{false}) = (H'', A'', v) \\
\text{Step 3.l.iii / writable = fase / DefineOwnProperty = err} & \\
(H, A, \mathit{err}) & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathtt{fals} \\
& \quad \wedge b = \mathtt{false} \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\mathtt{length}\text{”}, \mathit{ov}'', \mathtt{false}) = (H'', A'', \mathit{err}) \\
\text{Step 3.l.iii / writable = fase / DefineOwnProperty} \neq \text{err} & \\
(H, A, \text{if } b' \ \mathtt{TypeError} \text{ else } \mathtt{false}) & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathtt{fals} \\
& \quad \wedge b = \mathtt{false} \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\mathtt{length}\text{”}, \mathit{ov}'', \mathtt{false}) = (H'', A'', v) \\
\text{Step 3.l.ii} & \\
\mathit{DeleteArray}(H', A', l, \mathit{newLen}, & \text{if } \ \mathit{newLen} < \mathit{oldLen} \wedge \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, \mathit{oldLen} - 1), \mathtt{false}) = (H', A', \mathtt{true} \\
\qquad \mathit{oldLen} - 1, \mathit{ov}, b) &
\end{array}
\right.
$$

15.4.5.1 `[[DefineOwnProperty]](P,Desc,Throw)` for Array

$$
\begin{array}{ll}
\mathit{DefineOwnPropertyArray} & : \mathit{Heap} \times \mathit{Env} \times \mathit{Loc} \times \mathit{Var} \times \mathit{ObjectValue} \times \mathit{Bool} \rightarrow \mathit{Heap} \times \mathit{Env} \times (\mathit{Bool} \cup \mathit{Error}) \\
\mathit{DefineOwnPropertyArray}(H, A, l, x, \mathit{ov}, b) = & \\
\text{where} & \\
\mathit{index} & = \mathit{ToUint32}(H, x) \\
\mathit{newLenDesc} & = \mathit{copy}(\mathit{ov}) \\
\mathit{newLen} & = \mathit{ToUint32}(\mathit{ov}.[[\mathtt{Value}]]) \\
\mathit{newLenDesc}' & = \mathit{newLenDesc}\{[[\mathtt{Value}]] \mapsto \mathit{newLen}\} \\
\mathit{newLenDesc}'' & = \mathit{newLenDesc}'\{[[\mathtt{Writable}]] \mapsto \mathtt{true}\} \\
\mathit{newLenNum} & = \mathit{ToNumber}(H, \mathit{ov}.[[\mathtt{Value}]]) \\
\mathit{oldLenDesc} & = \mathit{GetOwnProperty}(H, l, \text{“}\mathtt{length}\text{”}) \\
\mathit{oldLen} & = \mathit{oldLenDesc}.[[\mathtt{Value}]] \\
\mathit{oldLenDesc}' & = \mathit{oldLenDesc}\{[[\mathtt{Value}]] \mapsto \mathit{index} + 1\}
\end{array}
$$

$$
\left\{
\begin{array}{ll}
\text{Step 3.a} & \\
\mathit{DefineOwnProperty}(H, A, l, \text{“}\mathtt{length}\text{”}, \mathit{ov}, b) & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \notin \mathit{ov} \\
\text{Step 3.d} & \\
(H, A, \mathtt{RangeError}) & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} \neq \mathit{newLenNum} \\
\text{Step 3.f.i} & \\
\mathit{DefineOwnProperty}(H, A, l, \text{“}\mathtt{length}\text{”}, & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
\qquad \mathit{newLenDesc}', b) & \quad \wedge \ \mathit{newLen} \geq \mathit{oldLen} \\
\text{Step 3.g} & \\
(H, A, \text{if } b \ \mathtt{TypeError} \text{ else } \mathtt{false}) & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \quad \wedge \mathit{newLen} < \mathit{oldLen} \wedge \neg \mathit{oldLenDesc}.[[\mathtt{Writable}]] \\
\text{Step 3.h : set newWritable = true \& 3.j} & \\
(H', A', \mathit{err}) & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \quad \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\mathtt{Writable}]] \\
& \quad \wedge ([[\mathtt{Writable}]] \notin \mathit{newLenDesc} \vee \mathit{newLenDesc}.[[\mathtt{Writable}]]) \\
& \quad \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\mathtt{length}\text{”}, \mathit{newLenDesc}', b) = (H', A', \mathit{err}) \\
\text{Step 3.h : set newWritable = true \& 3.k} & \\
(H', A', \mathtt{false}) & \text{if } \ \mathit{ToString}(H, x) = \text{“}\mathtt{length}\text{”} \wedge [[\mathtt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \quad \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\mathtt{Writable}]] \\
& \quad \wedge ([[\mathtt{Writable}]] \notin \mathit{newLenDesc} \vee \mathit{newLenDesc}.[[\mathtt{Writable}]]) \\
& \quad \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\mathtt{length}\text{”}, \mathit{newLenDesc}', b) = (H', A', \mathtt{false})
\end{array}
\right.
$$

$$
\begin{cases}
\text{Step 3.h : set newWritable = true \& 3.l} & \\
(H'', A'', \mathit{err}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \\
& \wedge([[\texttt{Writable}]] \notin \mathit{newLenDesc} \vee \mathit{newLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}', \texttt{true}, b) = (H'', A'', \mathit{err}) \\
\text{Step 3.h : set newWritable = true \& 3.l} & \\
(H'', A'', \text{if } b \; \texttt{TypeError} \text{ else } \texttt{false}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \\
& \wedge([[\texttt{Writable}]] \notin \mathit{newLenDesc} \vee \mathit{newLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}', \texttt{true}, b) = (H'', A'', \texttt{false}) \\
\text{Step 3.h : set newWritable = true \& 3.l} & \\
(H'', A'', \texttt{true}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \\
& \wedge([[\texttt{Writable}]] \notin \mathit{newLenDesc} \vee \mathit{newLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}', \texttt{true}, b) = (H'', A'', \texttt{true}) \\
\text{Step 3.h : set newWritable = false \& 3.j} & \\
(H', A', \mathit{err}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \wedge \neg \mathit{newLenDesc}.[[\texttt{Writable}]] \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}'', b) = (H', A', \mathit{err}) \\
\text{Step 3.h : set newWritable = false \& 3.k} & \\
(H', A', \texttt{false}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \wedge \neg \mathit{newLenDesc}.[[\texttt{Writable}]] \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}'', b) = (H', A', \texttt{false}) \\
\text{Step 3.h : set newWritable = false \& 3.l} & \\
(H'', A'', \mathit{err}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \wedge \neg \mathit{newLenDesc}.[[\texttt{Writable}]] \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}'', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}'', \texttt{false}, b) = (H'', A'', \mathit{err}) \\
\text{Step 3.h : set newWritable = false \& 3.l} & \\
(H'', A'', \text{if } b \; \texttt{TypeError} \text{ else } \texttt{false}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \wedge \neg \mathit{newLenDesc}.[[\texttt{Writable}]] \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}'', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}'', \texttt{false}, b) = (H'', A'', \texttt{false}) \\
\text{Step 3.h : set newWritable = false \& 3.m} & \\
(H^r, A^r, \mathit{ve}) & \text{if } \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge [[\texttt{Value}]] \in \mathit{ov} \wedge \mathit{newLen} = \mathit{newLenNum} \\
& \wedge \mathit{newLen} < \mathit{oldLen} \wedge \mathit{oldLenDesc}.[[\texttt{Writable}]] \wedge \neg \mathit{newLenDesc}.[[\texttt{Writable}]] \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, \text{“}\texttt{length}\text{”}, \mathit{newLenDesc}'', b) = (H', A', \texttt{true}) \\
& \wedge \mathit{DeleteArray}(H', A', l, \mathit{newLen}, \mathit{oldLen}, \mathit{newLenDesc}'', \texttt{false}, b) = (H'', A'', \texttt{true}) \\
& \wedge \mathit{DefineOwnProperty}(H'', A'', l, \text{“}\texttt{length}\text{”}, \{[[\texttt{Writable}]] : \texttt{false}\}, \texttt{false}) = \\
& (H^r, A^r, \mathit{ve}) \\
\text{Step 4.b} & \\
(H, A, \text{if } b \; \texttt{TypeError} \text{ else } \texttt{false}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge \mathit{index} \geq \mathit{oldLen} \wedge \neg \mathit{oldLenDesc}.[[\texttt{Writable}]] \\
\text{Step 4.c} & \\
(H', A', \mathit{err}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge (\mathit{index} < \mathit{oldLen} \vee \mathit{oldLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, x, \mathit{ov}, \texttt{false}) = (H', A', \mathit{err}) \\
\text{Step 4.d} & \\
(H', A', \text{if } b \; \texttt{TypeError} \text{ else } \texttt{false}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge (\mathit{index} < \mathit{oldLen} \vee \mathit{oldLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, x, \mathit{ov}, \texttt{false}) = (H', A', \texttt{false}) \\
\text{Step 4.e.ii} & \\
(H', A', \mathit{err}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge (\mathit{index} < \mathit{oldLen} \vee \mathit{oldLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, x, \mathit{ov}, \texttt{false}) = (H', A', \texttt{true}) \wedge \mathit{index} \geq \mathit{oldLen} \\
& \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\texttt{length}\text{”}, \mathit{oldLenDesc}', \texttt{false}) = (H'', A'', \mathit{err}) \\
\text{Step 4.e.ii} & \\
(H', A', \texttt{true}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge (\mathit{index} < \mathit{oldLen} \vee \mathit{oldLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, x, \mathit{ov}, \texttt{false}) = (H', A', \texttt{true}) \wedge \mathit{index} \geq \mathit{oldLen} \\
& \wedge \mathit{DefineOwnProperty}(H', A', l, \text{“}\texttt{length}\text{”}, \mathit{oldLenDesc}', \texttt{false}) = (H'', A'', v) \\
\text{Step 4.f} & \\
(H', A', \texttt{true}) & \text{if } \mathit{isIndex}(\mathit{ToString}(H,x)) \wedge (\mathit{index} < \mathit{oldLen} \vee \mathit{oldLenDesc}.[[\texttt{Writable}]]) \\
& \wedge \mathit{DefineOwnProperty}(H, A, l, x, \mathit{ov}, \texttt{false}) = (H', A', \texttt{true}) \wedge \mathit{index} < \mathit{oldLen} \\
\text{Step 5} & \\
\mathit{DefinOwnPropety}(H, A, l, x, \mathit{ov}, b) & \text{if } \neg \mathit{ToString}(H,x) = \text{“}\texttt{length}\text{”} \wedge \neg \mathit{isIndex}(\mathit{ToString}(H,x))
\end{cases}
$$

9 Type Conversion and Testing

9.1 `ToPrimitive`

$$\begin{array}{ll} \mathit{ToPrimitive} & : \mathit{Heap} \times \mathit{Val} \times \mathit{Str} \to \mathit{PVal} \\ \mathit{ToPrimitive}(H,v,s) & = \begin{cases} \mathit{DefaultValue}(H,v,s) & \text{if } v \in \mathit{Loc} \\ v & \text{otherwise} \end{cases} \end{array}$$

9.2 `ToBoolean`

$$\begin{array}{ll} \mathit{ToBoolean} & : \mathit{Val} \to \mathit{Bool} \\ \mathit{ToBoolean}(v) & = \begin{cases} \texttt{false} & \text{if } v \ = \texttt{undefined} \\ \texttt{false} & \text{if } v \ = \texttt{null} \\ v & \text{if } v \ \in \mathit{Bool} \\ \texttt{false} & \text{if } v \in \{+0, -0, \texttt{NaN}\} \\ \texttt{true} & \text{if } v \in \mathit{Num} \setminus \{+0, -0, \texttt{NaN}\} \\ \texttt{false} & \text{if } v = \texttt{""} \\ \texttt{true} & \text{if } v \in \mathit{Str} \setminus \{\text{""}\} \\ \texttt{true} & \text{if } v \in \mathit{Loc} \end{cases} \end{array}$$

9.3 `ToNumber`

$$\begin{array}{ll} \mathit{ToNumber} & : \mathit{Heap} \times \mathit{Val} \to \mathit{Num} \\ \mathit{ToNumber}(H,v) & = \begin{cases} \texttt{NaN} & \text{if } v \ = \texttt{undefined} \\ +0 & \text{if } v \ = \texttt{null} \lor v \ = \texttt{false} \\ 1 & \text{if } v \ = \texttt{true} \\ v & \text{if } v \in \mathit{Num} \\ v.\texttt{toString} & \text{if } v \in \mathit{Str} \quad \mathit{See}9.3.1. \\ \mathit{ToNumber}(H, \mathit{ToPrimitive}(H, v, \texttt{Number})) & \text{if } v \in \mathit{Loc} \end{cases} \end{array}$$

9.5 `ToInt32`

9.6 `ToUint32`

SKIP!

9.8 `ToString`

$$\begin{array}{ll} \mathit{ToString} & : \mathit{Heap} \times \mathit{Val} \to \mathit{Str} \\ \mathit{ToString}(H,v) & = \begin{cases} \texttt{"undefined"} & \text{if } v \ = \texttt{undefined} \\ \texttt{"null"} & \text{if } v \ = \texttt{null} \\ \texttt{"}v\texttt{"} & \text{if } v \ \in \mathit{Bool} \\ \texttt{"}v\texttt{"} & \text{if } v \in \mathit{Num} \quad \text{See 9.8.1.} \\ v & \text{if } v \in \mathit{Str} \\ \mathit{ToString}(H, \mathit{ToPrimitive}(H, v, \texttt{String})) & \text{if } v \in \mathit{Loc} \end{cases} \end{array}$$

9.9 `ToObject`

$$\begin{array}{ll} \mathit{ToObject} & : \mathit{Heap} \times \mathit{Val} \to \mathit{Heap} \times (\mathit{Loc} \cup \mathit{Error}) \\ \mathit{ToObject}(H,v) & = \begin{cases} (H, \texttt{TypeError}) & \text{if } v \ = \texttt{undefined} \lor v \ = \texttt{null} \\ (H[l \mapsto \mathit{NewBoolObject}(v)], l) & \text{if } v \ \in \mathit{Bool} \land l = \mathit{NewLoc}() \\ (H[l \mapsto \mathit{NewNumObject}(v)], l) & \text{if } v \ \in \mathit{Num} \land l = \mathit{NewLoc}() \\ (H[l \mapsto \mathit{NewStrObject}(v)], l) & \text{if } v \ \in \mathit{Str} \land l = \mathit{NewLoc}() \\ (H, v) & \text{if } v \ \in \mathit{Loc} \end{cases} \end{array}$$

9.10 `CheckObjectCoercible`

$$\begin{array}{ll} \mathit{CheckObjectCoercible} & : \mathit{Val} \to \mathit{ValError} \\ \mathit{CheckObjectCoercible}(v) & = \begin{cases} \texttt{TypeError} & \text{if } v \ = \texttt{undefined} \lor v \ = \texttt{null} \\ v & \text{if } v \ \neq \texttt{undefined} \land v \ \neq \texttt{null} \end{cases} \end{array}$$

9.11 `IsCallable`

$$\begin{array}{ll} \mathit{IsCallable} & : \mathit{Heap} \times \mathit{Val} \to \mathit{Bool} \\ \mathit{IsCallable}(H,v) & = v \in \mathit{Dom}(H) \land [[\texttt{Code}]] \in \mathit{Dom}(H(l)) \end{array}$$

10.2.1 Environment Records

10.2.1.1 Declarative Environment Records

10.2.1.1.1 `HasBinding(N)` $x \in \mathit{Dom}(\sigma)$

$$\begin{array}{ll} \mathit{HasBinding} & : \mathit{Env} \times \mathit{Var} \to \mathit{Bool} \\ \mathit{HasBinding}(A,x) & = \begin{cases} \texttt{false} & \text{if } \ A = \#\texttt{Global} \\ x \in \mathit{Dom}(\sigma) & \text{if } \ A = \sigma :: A' \\ \mathit{HasBinding}(A', x) & \text{if } \ A = l :: A' \end{cases} \\ \mathit{Dom}(\sigma) & = \{ \ x \ \mid \ x \mapsto \mathit{sv} \in \sigma \ \} \end{array}$$

10.2.1.2 Object Environment Records

10.2.1.2.1 `HasBinding(N)` 8.12.6 `[[HasProperty]](P)` $\mathit{HasProperty}(H,l,x)$

$$\mathit{Dom}(H) = \{\ l \mid l\mapsto o \in H\ \}$$

10.2.1.1.2 `CreateMutableBinding(N,D)` Assert: $x \notin \mathit{Dom}(\sigma)$

$$\mathit{CreateBinding} : \mathit{Heap}\times\mathit{Env}\times\mathit{Var}\times\mathtt{eval}\to\mathit{Heap}\times\mathit{Env}$$

$$\mathit{CreateBinding}(H,A,x,b) = \begin{cases} (H[\#\mathtt{Global}\mapsto H(\#\mathtt{Global}).\mathtt{@property}[x\mapsto\{[[\mathtt{Value}]]:\mathtt{undefined}, \\ \quad [[\mathtt{Writable}]]:\mathtt{true}, \\ \quad [[\mathtt{Enumerable}]]:\mathtt{true}, \\ \quad [[\mathtt{Configurable}]]:b\}]], A) & \text{if } A=\#\mathtt{Global} \\ \mathit{InterpreterError} & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma) \\ (H,\sigma[x\mapsto\{[[\mathtt{Value}]]:\mathtt{undefined}, [[\mathtt{Mutable}]]:\mathtt{true}, \\ \quad [[\mathtt{Configurable}]]:b\}]::A') & \text{if } A=\sigma::A'\wedge x\notin \mathit{Dom}(\sigma) \\ (H',l::A'') & \text{if } A=l::A'\wedge \\ & \quad \mathit{CreateBinding}(H,A',x,b)=(H',A'') \end{cases}$$

10.2.1.1.3 `SetMutableBinding(N,V,S)`

$$\mathit{SetBinding} : \mathit{Heap}\times\mathit{Env}\times\mathit{Var}\times\mathit{Val}\times\mathtt{strict}\to\mathit{Heap}\times\mathit{Env}\times\mathit{ValError}$$

$$\mathit{SetBinding}(H,A,x,v,b) = \begin{cases} (H[\#\mathtt{Global}\mapsto H(\#\mathtt{Global}).\mathtt{@property}[x\mapsto v]],A,v) & \text{if } A=\#\mathtt{Global} \\ (H,\sigma[x\mapsto\{[[\mathtt{Value}]]:v, [[\mathtt{Mutable}]]:\mathtt{true}, \\ \quad [[\mathtt{Configurable}]]:b\}]::A',v) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \sigma(x).[[\mathtt{Mutable}]] \\ (H,A,\mathtt{TypeError}) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \neg\sigma(x).[[\mathtt{Mutable}]]\wedge b \\ (H,A,v) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \neg\sigma(x).[[\mathtt{Mutable}]]\wedge \neg b \\ \mathit{SetBinding}(H',A',x,v,b) & \text{if } A=\sigma::A'\wedge x\notin \mathit{Dom}(\sigma)\wedge \\ & \quad \mathit{CreateBinding}(H,A,x,b)=(H',A') \\ (H',l::A'',\mathit{ve}) & \text{if } A=l::A' \\ & \quad \mathit{SetBinding}(H,A',x,v,b)=(H',A'',\mathit{ve}) \end{cases}$$

$$\mathit{SetBindingDER} : \mathit{Heap}\times\mathit{Env}\times\mathit{Var}\times\mathit{Val}\times\mathtt{strict}\to\mathit{Heap}\times\mathit{Env}\times\mathit{ValError}$$

$$\mathit{SetBindingDER}(H,A,x,v,b) = \begin{cases} \mathit{InterpreterError} & \text{if } A=\#\mathtt{Global} \\ (H,\sigma[x\mapsto\{[[\mathtt{Value}]]:v, [[\mathtt{Mutable}]]:\mathtt{true}, \\ \quad [[\mathtt{Configurable}]]:b\}]::A',v) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \sigma(x).[[\mathtt{Mutable}]] \\ (H,A,\mathtt{TypeError}) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \neg\sigma(x).[[\mathtt{Mutable}]]\wedge b \\ (H,A,v) & \text{if } A=\sigma::A'\wedge x\in \mathit{Dom}(\sigma)\wedge \\ & \quad \neg\sigma(x).[[\mathtt{Mutable}]]\wedge \neg b \\ (H',\sigma::A'',\mathit{ve}) & \text{if } A=\sigma::A'\wedge x\notin \mathit{Dom}(\sigma)\wedge \\ & \quad \mathit{SetBindingDER}(H,A',x,v,b)=(H',A'',\mathit{ve}) \\ (H',l::A'',\mathit{ve}) & \text{if } A=l::A' \\ & \quad \mathit{SetBindingDER}(H,A',x,v,b)=(H',A'',\mathit{ve}) \end{cases}$$

10.2.1.1.4 `GetBindingValue(N,S)` Assert: $x \in \mathit{Dom}(\sigma)$

$$\mathit{GetBindingValue} : \mathit{Heap}\times\mathit{DeclEnvRec}\times\mathit{Var}\times\mathit{Bool}\to\mathit{ValError}$$

$$\mathit{GetBindingValue}(H,\sigma,x,b) = \begin{cases} \mathtt{undefined} & \text{if } \sigma(x).[[\mathtt{Value}]]=\bot\wedge\neg\sigma(x).[[\mathtt{Mutable}]]\wedge\neg b \\ \mathtt{ReferenceError} & \text{if } \sigma(x).[[\mathtt{Value}]]=\bot\wedge\neg\sigma(x).[[\mathtt{Mutable}]]\wedge b \\ \sigma(x).[[\mathtt{Value}]] & \text{if } \sigma(x).[[\mathtt{Value}]]\neq\bot\vee\sigma(x).[[\mathtt{Mutable}]] \end{cases}$$

10.2.1.1.7 `CreateImmutableBinding(N)` Assert: $x \notin \mathit{Dom}(\sigma)$

$$\mathit{CreateImmutableBinding} : \mathit{Heap}\times\mathit{Env}\times\mathit{Var}\times\mathtt{eval}\to\mathit{Heap}\times\mathit{Env}$$

$$\mathit{CreateImmutableBinding}(H,A,x,b) = \begin{cases} (H,\sigma[x\mapsto\{[[\mathtt{Value}]]:\mathtt{undefined}, [[\mathtt{Mutable}]]:\mathtt{false}, [[\mathtt{Configurable}]]:b\}]::A') & \text{if } A=\sigma::A'\wedge x\notin \mathit{Dom}(\sigma) \\ \mathit{InterpreterError} & \text{otherwise} \end{cases}$$

10.2.1.1.8 `InitializeImmutableBinding(N,V)` Assert: $x \in \mathit{Dom}(\sigma)$

$$\mathit{InitializeImmutableBinding} \quad : \mathit{Heap} \times \mathit{Env} \times \mathit{Var} \times \mathit{Val} \times \texttt{strict} \to \mathit{Heap} \times \mathit{Env} \times \mathit{ValError}$$

$$\mathit{InitializeImmutableBinding}(H, A, x, v, b) = \begin{cases} (H, \sigma[x \mapsto \{[[\texttt{Value}]] : v, [[\texttt{Mutable}]] : \texttt{false}, [[\texttt{Configurable}]] : b\}] :: A', v) & \text{if } A = \sigma :: A' \wedge x \in \mathit{Dom}(\sigma) \\ \mathit{InterpreterError} & \text{otherwise} \end{cases}$$

10.2.1.2.4 `GetBindingValue(N,S)`

$$\mathit{GetBindingValue} \quad : \mathit{Heap} \times \mathit{ObjEnvRec} \times \mathit{Var} \times \mathit{Bool} \to \mathit{ValError}$$

$$\mathit{GetBindingValue}(H, l, x, b) = \begin{cases} \texttt{ReferenceError} & \text{if } l = \#\texttt{Null} \\ \texttt{undefined} & \text{if } \neg \mathit{HasProperty}(H, l, x) \wedge \neg b \\ \texttt{ReferenceError} & \text{if } \neg \mathit{HasProperty}(H, l, x) \wedge b \\ \mathit{Get}(H, l, x) & \text{if } \mathit{HasProperty}(H, l, x) \end{cases}$$

10.2.2.1 `GetIdentifierReference(lex, name,strict)`

$$\mathit{Lookup} \quad : \mathit{Heap} \times \mathit{Env} \times \mathit{PName} \times \texttt{strict} \to \mathit{EnvRec}$$

$$\mathit{Lookup}(H, A, x, \texttt{strict}) = \begin{cases} \#\texttt{Null} & \text{if } A = \#\texttt{Global} \wedge \neg \mathit{HasProperty}(H, \#\mathit{Global}, x) \\ l & \text{if } A = \#\texttt{Global} \wedge \mathit{HasProperty}(H, \#\mathit{Global}, x) \wedge \\ & \quad \mathit{GetProperty}(H, \#\mathit{Global}, x) = (\_, l) \\ \sigma & \text{if } A = \sigma :: A' \wedge x \in \mathit{Dom}(\sigma) \\ \mathit{Lookup}(H, A', x, \texttt{strict}) & \text{if } A = \sigma :: A' \wedge x \notin \mathit{Dom}(\sigma) \\ l' & \text{if } A = l :: A' \wedge \mathit{HasProperty}(H, l, x) \wedge \\ & \quad \mathit{GetProperty}(H, l, x) = (\_, l') \\ \mathit{Lookup}(H, A', x, \texttt{strict}) & \text{if } A = l :: A' \wedge \neg \mathit{HasProperty}(H, l, x) \end{cases}$$

15 Standard Built-in ECMAScript Objects

$\mathit{InitHeap} = \{$

15.1 The Global Object

$\#\texttt{Global} \mapsto \{\}$,

15.2.3.1 `Object.prototype`

15.2.4 Properties of the Object Prototype Object

$\#\texttt{ObjProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Object}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \texttt{null}\}$,

15.3.3.1 `Function.prototype`

15.3.4 Properties of the Function Prototype Object

$\#\texttt{FtnProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Function}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto},$
$[[\texttt{Code}]] : \text{function } \_(\underline{\mathit{this}}, \underline{\mathit{arguments}})\{\text{return } \texttt{undefined}\}, \texttt{length} : 0\}$,

15.4.3.1 `Array.prototype`

15.4.4 Properties of the Array Prototype Object

$\#\texttt{ArrProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Array}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto}, \texttt{length} : 0\}$,

15.5.3.1 `String.prototype`

15.5.4 Properties of the String Prototype Object

$\#\texttt{StrProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{String}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto}, [[\texttt{PrimitiveValue}]] : \text{“”}\}$,

15.6.3.1 `Boolean.prototype`

15.6.4 Properties of the Boolean Prototype Object

$\#\texttt{BoolProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Boolean}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto}, [[\texttt{PrimitiveValue}]] : \texttt{fals}$

15.7.3.1 `Number.prototype`

15.7.4 Properties of the Number Prototype Object

$\#\texttt{NumProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Number}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto}, [[\texttt{PrimitiveValue}]] : +0\}$,

15.11.3.1 `Error.prototype`

15.11.4 Properties of the Error Prototype Object

$\#\texttt{ErrProto} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ObjProto}\}$,

15.11 `Error` Objects

15.11.1 `Error(message)`

$\#\texttt{Error} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\}$,

15.11.6 Native Error Types Used in This Standard
15.11.7 NativeError Object Structure
15.11.7.5 Properties of the NativeError

$$
\begin{array}{l}
\quad\#\texttt{EvalError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\},\\
\quad\#\texttt{RangeError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\},\\
\quad\#\texttt{ReferenceError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\},\\
\quad\#\texttt{SyntaxError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\},\\
\quad\#\texttt{TypeError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\},\\
\quad\#\texttt{URIError} \mapsto \{[[\texttt{Class}]] : \text{“}\texttt{Error}\text{”}, [[\texttt{Extensible}]] : \texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ErrProto}\}\\
\}
\end{array}
$$

15.4 Array Objects
15.4.2.1 new Array([item0[, item1[, ...]]])

$$
\begin{aligned}
\mathit{NewArrObject} :\ & \mathit{Num} \to \mathit{Object}\\
\mathit{NewArrObject}(n) =\ & \{[[\texttt{Class}]] : \text{“}\texttt{Array}\text{”}, [[\texttt{Extensible}]] :\texttt{true}, [[\texttt{Prototype}]] : \#\texttt{ArrProto},\\
& \ @\texttt{property} : \{\text{“}\texttt{length}\text{”} \mapsto \{[[\texttt{Value}]] :n, [[\texttt{Writable}]] :\texttt{true},\\
& \qquad\qquad\qquad\qquad\quad [[\texttt{Enumerable}]] :\texttt{false}, [[\texttt{Configurable}]] :\texttt{false}\}\}\}
\end{aligned}
$$

15.5 String Objects
15.5.2.1 new String([value])

$$
\begin{aligned}
\mathit{NewStrObject} :\ & \mathit{Str} \to \mathit{Object}\\
\mathit{NewStrObject}(s) =\ & \{[[\texttt{Class}]] : \text{“}\texttt{String}\text{”}, [[\texttt{Extensible}]] :\texttt{true}, [[\texttt{Prototype}]] : \#\texttt{StrProto}, [[\texttt{PrimitiveValue}]] :s\}
\end{aligned}
$$

15.6 Boolean Objects
15.6.2.1 new Boolean(value)

$$
\begin{aligned}
\mathit{NewBoolObject} :\ & \mathit{Bool} \to \mathit{Object}\\
\mathit{NewBoolObject}(b) =\ & \{[[\texttt{Class}]] : \text{“}\texttt{Boolean}\text{”}, [[\texttt{Extensible}]] :\texttt{true}, [[\texttt{Prototype}]] : \#\texttt{BoolProto}, [[\texttt{PrimitiveValue}]] :b\}
\end{aligned}
$$

15.7 Number Objects
15.7.2.1 new Number([value])

$$
\begin{aligned}
\mathit{NewNumObject} :\ & \mathit{Num} \to \mathit{Object}\\
\mathit{NewNumObject}(n) =\ & \{[[\texttt{Class}]] : \text{“}\texttt{Number}\text{”}, [[\texttt{Extensible}]] :\texttt{true}, [[\texttt{Prototype}]] : \#\texttt{NumProto}, [[\texttt{PrimitiveValue}]] :n\}
\end{aligned}
$$

13.2 `Creating Function Objects`: The following properties are omitted for now:

`[[Get]]`: 15.3.5.4 `[[Get]](P)`
`[[Call]]`: 13.2.1 `[[Call]]`
`[[Construct]]`: 13.2.2 `[[Construct]]`
`[[HasInstance]]`: 15.3.5.3 `[[HasInstance]](V)`
`[[FormalParameters]]`

$$
\begin{aligned}
\mathit{NewFtnObject} :\ & \mathit{Heap} \times \mathit{Env} \times \mathit{FVal} \to \mathit{Heap} \times \mathit{Loc}\\
\mathit{NewFtnObject}(H, A, \mathit{fv}) =\ & (H[l \mapsto \mathit{NewFtnObj}(\mathit{fv}, A, l', \texttt{strict}), l' \mapsto o], l)\\
\text{where}\ \ & l = \mathit{NewLoc}() \qquad l' = \mathit{NewLoc}()\\
& o = \mathit{NewObj}().@\texttt{property}[\text{“}\texttt{constructor}\text{”} \mapsto \{[[\texttt{Value}]] :l, [[\texttt{Writable}]] :\texttt{true},\\
& \qquad\qquad\qquad\qquad\qquad\qquad\quad [[\texttt{Enumerable}]] :\texttt{false}, [[\texttt{Configurable}]] :\texttt{true}\}]
\end{aligned}
$$

$$
\begin{aligned}
\mathit{NewFtnObj} :\ & \mathit{FVal} \times \mathit{Env} \times \mathit{Loc} \times \texttt{strict} \to \mathit{Object}\\
\mathit{NewFtnObj}(\mathit{fv}, A, l, b) =\ & \{[[\texttt{Class}]] : \text{“}\texttt{Function}\text{”},\\
& \ [[\texttt{Extensible}]] :\texttt{true},\\
& \ [[\texttt{Prototype}]] : \#\texttt{FtnProto},\\
& \ @\texttt{property} : \{\text{“}\texttt{prototype}\text{”} \mapsto \{[[\texttt{Value}]] :l, [[\texttt{Writable}]] :\texttt{true},\\
& \qquad\qquad\qquad\qquad\qquad [[\texttt{Enumerable}]] :\texttt{false}, [[\texttt{Configurable}]] :\texttt{false}\},\\
& \qquad\qquad\quad \text{“}\texttt{length}\text{”} \mapsto \{[[\texttt{Value}]] :\mathit{ParamsSize}(\mathit{fv}), [[\texttt{Writable}]] :\texttt{false},\\
& \qquad\qquad\qquad\qquad\quad [[\texttt{Enumerable}]] :\texttt{false}, [[\texttt{Configurable}]] :\texttt{false}\}\},\\
& \ [[\texttt{Code}]] :\mathit{fv},\\
& \ [[\texttt{Scope}]] :A\}
\end{aligned}
$$

15.2.2.1 `new Object([value])`

$$
\begin{aligned}
\mathit{NewObj} :\quad & () \rightarrow \mathit{Object}\\
\mathit{NewObj}() =\quad & \{[[\texttt{Class}]] : \text{“}\texttt{Object}\text{”},\\
& \ [[\texttt{Extensible}]] : \texttt{true},\\
& \ [[\texttt{Prototype}]] : \#\texttt{ObjProto},\\
& \ @\texttt{property} : \{\}\}
\end{aligned}
$$

10.6 `Arguments Object`

$$
\begin{aligned}
\mathit{NewArgObject} :\quad & \mathit{Loc} \times \mathit{Num} \times \mathit{Object} \times \texttt{strict} \rightarrow \mathit{Object}\\
\mathit{NewArgObject}(l_f, n_p, o, b) =\quad & \begin{cases} \mathit{NewArgObj}(l_f, n_p, o).@\texttt{property}[\text{“}\texttt{callee}\text{”} \mapsto \{[[\texttt{Value}]] : l_f, [[\texttt{Writable}]] : \texttt{true}, & \text{if } \ b\\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad [[\texttt{Enumerable}]] : \texttt{false}, & \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad [[\texttt{Configurable}]] : \texttt{true}\}] & \\ \mathit{NewArgObj}(l_f, n_p, o) & \text{if } \ \neg b \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{NewArgObj} :\quad & \mathit{Loc} \times \mathit{Num} \times \mathit{Object} \rightarrow \mathit{Object}\\
\mathit{NewArgObj}(l_f, n_p, o) =\quad & \{[[\texttt{Class}]] : \text{“}\texttt{Arguments}\text{”},\\
& \ [[\texttt{Extensible}]] : \texttt{true},\\
& \ [[\texttt{Prototype}]] : \#\texttt{ObjProto},\\
& \ @\texttt{property} : \{\\
& \qquad \text{“}0\text{”} \mapsto \{[[\texttt{Value}]] : v_0, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\},\\
& \qquad \ldots\\
& \qquad \text{“}n-1\text{”} \mapsto \{[[\texttt{Value}]] : v_{n-1}, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\\
& \qquad \text{“}\texttt{length}\text{”} \mapsto \{[[\texttt{Value}]] : n_a, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{false}, [[\texttt{Configurable}]] : \texttt{true}\\
\text{where}\quad & n_a = o.@\texttt{property}(\text{“}\texttt{length}\text{”}) \qquad n = \mathit{Max}(n_a, n_p) \qquad v_i = \mathit{GetArg}(o, i, n_a, n_p)
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Max} :\quad & \mathit{Num} \times \mathit{Num} \rightarrow \mathit{Num}\\
\mathit{Max}(n_1, n_2) =\quad & \begin{cases} n_1 & \text{if } \ n_1 \geq n_2\\ n_2 & \text{if } \ n_1 < n_2 \end{cases}\\
\mathit{GetArg} :\quad & \mathit{Object} \times \mathit{Num} \times \mathit{Num} \times \mathit{Num} \rightarrow \mathit{Val}\\
\mathit{GetArg}(o, i, n_a, n_p) =\quad & \begin{cases} o.@\texttt{property}(\text{“}i\text{”}) & \text{if } \ n_a \geq n_p\\ o.@\texttt{property}(\text{“}i\text{”}) & \text{if } \ n_a < n_p \ \wedge \ 0 \leq i < n_a\\ \texttt{undefined} & \text{if } \ n_a < n_p \ \wedge \ n_a \leq i < n_p \end{cases}
\end{aligned}
$$

5.4 Evaluation Rules

5.4.1 Program

$$\boxed{\underline{p} \rightarrow_{\underline{p}} (H, A), \mathit{ct}}$$

`IRRoot(List<IRFunDecl> fds, List<IRVarStmt> vds, List<IRStmt> irs)`

14 Program

10.4.1.1 Initial Global Execution Context

$$\frac{(\mathit{InitHeap}, \#\texttt{Global}, \#\texttt{Global}), \underline{p} \rightarrow_{\underline{s}} (H, A), \mathit{ct}}{\underline{p} \rightarrow_{\underline{p}} (H, A), \mathit{ct}}$$

5.4.2 Statements

$$\boxed{(H, A, \mathit{tb}), \underline{s} \rightarrow_{\underline{s}} (H, A), \mathit{ct}}$$

`IRExprStmt(IRId lhs, IRExpr right, boolean ref = false)`

12.4 Expression Statement

11.13 Assignment Operators: `PutValue(lref, rval)`

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x} = \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v \qquad \mathit{PutValue}(H, A, \underline{x}, v, \texttt{strict}) = (H', A', \mathit{err})}{(H, A, \mathit{tb}), \underline{x} = \underline{e} \rightarrow_{\underline{s}} (H', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v \qquad \mathit{PutValue}(H, A, \underline{x}, v, \texttt{strict}) = (H', A', v')}{(H, A, \mathit{tb}), \underline{x} = \underline{e} \rightarrow_{\underline{s}} (H', A'), \texttt{Normal}(v')}$$

11.4.1 The `delete` Operator

`IRDelete(IRId lhs, IRId id)`

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v \qquad \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, y), \texttt{strict}) = (H', A', \mathit{err})}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y} \rightarrow_{\underline{s}} (H', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v \qquad \mathit{DeleteBinding}(H, A, \mathit{ToString}(H, y), \texttt{strict}) = (H', A', b) \qquad (H', A', \mathit{tb}), \underline{x} = b \rightarrow_{\underline{s}} (H'', A''), \mathit{ct}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y} \rightarrow_{\underline{s}} (H'', A''), \mathit{ct}}$$

`IRDeleteProp(IRId lhs, IRId obj, IRId index)`

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y}[\underline{z}] \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v \qquad (H, A, \mathit{tb}), \underline{z} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y}[\underline{z}] \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) = \mathit{err}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y}[\underline{z}] \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ \mathit{Delete}(H', A, l, \mathit{ToString}(H', v_2), \texttt{strict}) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y}[\underline{z}] \rightarrow_{\underline{s}} (H'', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ \mathit{Delete}(H', A, l, \mathit{ToString}(H', v_2), \texttt{strict}) = (H'', A', b) \qquad (H'', A', \mathit{tb}), \underline{x} = b \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}\end{array}}{(H, A, \mathit{tb}), \underline{x} = \textsf{delete } \underline{y}[\underline{z}] \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}}$$

`IRStore(IRId obj, IRId index, IRExpr rhs)`

11.2.1 Property Accessors: `IRStore`
11.13 Assignment Operators
8.7.2 `PutValue(V, W)`
8.7.2 `[[Put]](P, V, Throw)`

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) = \mathit{err}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H', A), \texttt{Throw}(\mathit{err})}$$

[[Put]] 2.

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \qquad v_1 \in \{\texttt{Boolean}, \texttt{String}, \texttt{Number}\} \qquad \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \texttt{false}\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H', A), \text{if } \texttt{strict}\ \texttt{Throw}(\texttt{TypeError}) \text{ else } \texttt{Normal}(v_3)}$$

[[Put]] 4.

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \qquad v_1 \in \{\texttt{Boolean}, \texttt{String}, \texttt{Number}\} \qquad \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \texttt{true} \\ \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{dp}, \_)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H', A), \text{if } \texttt{strict}\ \texttt{Throw}(\texttt{TypeError}) \text{ else } \texttt{Normal}(v_3)}$$

[[Put]] 6.

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \qquad v_1 \in \{\texttt{Boolean}, \texttt{String}, \texttt{Number}\} \qquad \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \texttt{true} \\ \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \qquad \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{ap}, \_) \\ \mathit{ap}.[[\texttt{Set}]].[[\texttt{Call}]](v_1, [v_3]) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \texttt{Throw}(\mathit{err})}$$

[[Put]] 6.

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \qquad v_1 \in \{\texttt{Boolean}, \texttt{String}, \texttt{Number}\} \qquad \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \texttt{true} \\ \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \qquad \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{ap}, \_) \\ \mathit{ap}.[[\texttt{Set}]].[[\texttt{Call}]](v_1, [v_3]) = (H'', A', v)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \texttt{Normal}(v_3)}$$

[[Put]] 7.

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \qquad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \qquad v_1 \in \{\texttt{Boolean}, \texttt{String}, \texttt{Number}\} \qquad \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \texttt{true} \\ \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \qquad \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{ap}, \_)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H', A), \text{if } \texttt{strict}\ \texttt{Throw}(\texttt{TypeError}) \text{ else } \texttt{Normal}(v_3)}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] \neq \text{“}\mathtt{Array}\text{”} \\ \mathit{Put}(H', A, l, \mathit{ToString}(H', v_2), v_3, \mathtt{strict}) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] \neq \text{“}\mathtt{Array}\text{”} \\ \mathit{Put}(H', A, l, \mathit{ToString}(H', v_2), v_3, \mathtt{strict}) = (H'', A', v)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Normal}(v_3)}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \mathtt{false}\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H', A), \text{if } \mathtt{strict}\ \mathtt{Throw(TypeError)} \text{ else } \mathtt{Normal}(v_3)}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{dp}, \_) \quad \mathit{valueDesc} = \{[[\mathtt{Value}]] : v_3\} \\ \mathit{DefineOwnPropertyArray}(H', A, l, \mathit{ToString}(H', v_2), \mathit{valueDesc}, \mathtt{strict}) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2)) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{dp}, \_) \quad \mathit{valueDesc} = \{[[\mathtt{Value}]] : v_3\} \\ \mathit{DefineOwnPropertyArray}(H', A, l, \mathit{ToString}(H', v_2), \mathit{valueDesc}, \mathtt{strict}) = (H'', A', v)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Normal}(v_3)}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \\ \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{ap}, \_) \quad \mathit{ap}.[[\mathtt{Set}]].[[\mathtt{Call}]](H'(l), v_3) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \\ \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) = (\mathit{ap}, \_) \quad \mathit{ap}.[[\mathtt{Set}]].[[\mathtt{Call}]](H'(l), v_3) = (H'', A', v)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Normal}(v_3)}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \\ \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{ap}, \_) \\ \mathit{newDesc} = \{[[\mathtt{Value}]] : v_3, [[\mathtt{Writable}]] : \mathtt{true}, [[\mathtt{Enumerable}]] : \mathtt{true}, [[\mathtt{Conigurable}]] : \mathtt{true}\} \\ \mathit{DefineOwnPropertyArray}(H', A, l, \mathit{ToString}(H', v_2), \mathit{newDesc}, \mathtt{strict}) = (H'', A', \mathit{err})\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v_1 \quad (H, A, \mathit{tb}), \underline{y} \rightarrow_{\underline{e}} v_2 \quad \mathit{CheckObjectCoercible}(v_1) \neq \mathit{err} \quad \mathit{ToObject}(H, v_1) = (H', l) \\ (H', A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v_3 \quad v_1 \notin \{\mathtt{Boolean}, \mathtt{String}, \mathtt{Number}\} \quad H'(l).[[\mathtt{Class}]] = \text{“}\mathtt{Array}\text{”} \\ \mathit{CanPut}(H', l, \mathit{ToString}(H', v_2) = \mathtt{true} \quad \mathit{GetOwnPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{dp}, \_) \\ \mathit{GetPropety}(H', l, \mathit{ToString}(H', v_2)) \neq (\mathit{ap}, \_) \\ \mathit{newDesc} = \{[[\mathtt{Value}]] : v_3, [[\mathtt{Writable}]] : \mathtt{true}, [[\mathtt{Enumerable}]] : \mathtt{true}, [[\mathtt{Conigurable}]] : \mathtt{true}\} \\ \mathit{DefineOwnPropertyArray}(H', A, l, \mathit{ToString}(H', v_2), \mathit{newDesc}, \mathtt{strict}) = (H'', A', v)\end{array}}{(H, A, \mathit{tb}), \underline{x}[\underline{y}] = \underline{e} \rightarrow_{\underline{s}} (H'', A'), \mathtt{Normal}(v_3)}$$

`IRObject(IRId lhs, List<IRMember> members, Option<IRId> proto)`

11.1.5 Object Initialiser

$$\frac{\mathit{NewLoc}() = l \qquad H' = H[l \mapsto \mathit{NewObj}()] \qquad (H', A, \mathit{tb}), \underline{x} = l \rightarrow_{\underline{s}} (H'', A'), \mathit{ct}}{(H, A, \mathit{tb}), \underline{x} = \{\} \rightarrow_{\underline{s}} (H'', A'), \mathit{ct}}$$

$$\frac{\begin{array}{c}\mathit{NewLoc}() = l \qquad H_1 = H[l \mapsto \mathit{NewObj}()] \qquad (H_i, A, \mathit{tb}), \underline{m}_i \rightarrow_{\underline{m}} (H_{i+1}, \underline{y}_i, \mathit{ov}_i) \qquad 1 \leq i < j \leq |m^*| \\ (H_j, A, \mathit{tb}), \underline{m}_j \rightarrow_{\underline{m}} \mathit{err}\end{array}}{(H, A, \mathit{tb}), \underline{x} = \{(\underline{m},)^*\} \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}\mathit{NewLoc}() = l \qquad H_1 = H[l \mapsto \mathit{NewObj}()] \qquad A_1 = A \qquad (H_i, A_i, \mathit{tb}), \underline{m}_i \rightarrow_{\underline{m}} (H'_{i+1}, \underline{y}_i, \mathit{ov}_i) \qquad 1 \leq i \leq |m^*| = n \\ \mathit{DefineOwnProperty}(H'_{i+1}, A_i, l, \mathit{ToString}(H'_{i+1}, \underline{y}_i, \mathit{ov}_i, \mathtt{false}) = (H_{i+1}, A_{i+1}, v) \qquad (H_{n+1}, A_{n+1}, \mathit{tb}), \underline{x} = l \rightarrow_{\underline{s}} (H', A'), \mathit{ct}\end{array}}{(H, A, \mathit{tb}), \underline{x} = \{(\underline{m},)^*\} \rightarrow_{\underline{s}} (H', A'), \mathit{ct}}$$

`IRArray(IRId lhs, List<Option<IRExpr>> elements)`

11.1.4 Array Initialiser

$$\frac{\begin{array}{c}\mathit{NewLoc}() = l \qquad |(\underline{e})^*| = n \qquad H' = H[l \mapsto \mathit{NewArrObject}(n)] \qquad (H', A, \mathit{tb}), \underline{e_i} \rightarrow_{\underline{e}} v_i \qquad 0 \leq i < j < n \\ (H', A, \mathit{tb}), \underline{e_j} \rightarrow_{\underline{e}} \mathit{err}\end{array}}{(H, A, \mathit{tb}), \underline{x} = [(\underline{e},)^*] \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}\mathit{NewLoc}() = l \qquad |(\underline{y})^*| = n \qquad H' = H[l \mapsto \mathit{NewArrObject}(n)] \qquad (H', A, \mathit{tb}), \underline{y_i} \rightarrow_{\underline{e}} v_i \qquad 0 \leq i < n \\ \{[[\mathtt{Value}]] : v_i, [[\mathtt{Writable}]] : \mathtt{true}, [[\mathtt{Enumerable}]] : \mathtt{true}, [[\mathtt{Configurable}]] : \mathtt{true}\} = \mathit{dp}_i \\ H'_0 = H' \qquad A_0 = A \qquad \mathit{DefineOwnProperty}(H'_i, A_i, l, \mathit{ToString}(i), \mathit{dp}_i, \mathtt{false}) = (H'_{i+1}, A_{i+1}, v) \qquad H'' = H'_n \qquad A' = A_n \\ (H'', A', \mathit{tb}), \underline{x} = l \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}\end{array}}{(H, A, \mathit{tb}), \underline{x} = [(\underline{e},)^*] \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}}$$

`IRArgs(IRId lhs, List<Option<IRExpr>> elements)`

`IRCall(IRId lhs, IRId fun, IRId thisB, IRId args)`

11.2.3 Function Calls

11.2.4 Argument Lists

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} err}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(err)}$$

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} err}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(err)}$$

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} err}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(err)}$$

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \notin Loc}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad \neg IsCallable(H,v)}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \notin Loc}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H,A),\mathtt{Throw}(\mathtt{TypeError})}$$

10.4.3 Entering Function Code

$$\frac{\begin{array}{c}(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \in Loc\\ H(v).[[\mathsf{Code}]] = fv \qquad \{\} :: H(v).[[\mathsf{Scope}]] = A'\\ NewLoc() = l \qquad H' = H[l \mapsto NewArgObj(v, ParamsSize(fv), H(v_2))]\\ CreateBinding(H',A',arguments,\mathtt{eval}) = (H'',A'') \qquad SetBinding(H'',A'',arguments,l,\mathtt{strict}) = (H^f,A^f,err)\end{array}}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H^f,A),\mathtt{Throw}(err)}$$

$$\frac{\begin{array}{c}(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \in Loc\\ H(v).[[\mathsf{Code}]] = fv \qquad \{\} :: H(v).[[\mathsf{Scope}]] = A'\\ NewLoc() = l \qquad H' = H[l \mapsto NewArgObj(v, ParamsSize(fv), H(v_2))]\\ CreateBinding(H',A',arguments,\mathtt{eval}) = (H'',A'') \qquad SetBinding(H'',A'',arguments,l,\mathtt{strict}) = (H^f,A^f,v^f)\\ GetThis(H^f,v_1) = (H^t,l^t) \qquad (H^t,A^f,l^t),GetBody(fv) \to_{\underline{s}} (H^b,A^b),\mathtt{Throw}(ve)\end{array}}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H^b,A),\mathtt{Throw}(ve)}$$

$$\frac{\begin{array}{c}(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \in Loc\\ H(v).[[\mathsf{Code}]] = fv \qquad \{\} :: H(v).[[\mathsf{Scope}]] = A'\\ NewLoc() = l \qquad H' = H[l \mapsto NewArgObj(v, ParamsSize(fv), H(v_2))]\\ CreateBinding(H',A',arguments,\mathtt{eval}) = (H'',A'') \qquad SetBinding(H'',A'',arguments,l,\mathtt{strict}) = (H^f,A^f,v^f)\\ GetThis(H^f,v_1) = (H^t,l^t) \qquad (H^t,A^f,l^t),GetBody(fv) \to_{\underline{s}} (H^b,A^b),\mathtt{Return}(\mathtt{empty})\\ (H^b,A,tb),\underline{x} = \mathtt{undefined} \to_{\underline{s}} (H^r,A^r),ct\end{array}}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H^r,A^r),ct}$$

$$\frac{\begin{array}{c}(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \in Loc\\ H(v).[[\mathsf{Code}]] = fv \qquad \{\} :: H(v).[[\mathsf{Scope}]] = A'\\ NewLoc() = l \qquad H' = H[l \mapsto NewArgObj(v, ParamsSize(fv), H(v_2))]\\ CreateBinding(H',A',arguments,\mathtt{eval}) = (H'',A'') \qquad SetBinding(H'',A'',arguments,l,\mathtt{strict}) = (H^f,A^f,v^f)\\ GetThis(H^f,v_1) = (H^t,l^t) \qquad (H^t,A^f,l^t),GetBody(fv) \to_{\underline{s}} (H^b,A^b),\mathtt{Return}(v^b)\\ (H^b,A,tb),\underline{x} = v^b \to_{\underline{s}} (H^r,A^r),ct\end{array}}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H^r,A^r),ct}$$

$$\frac{\begin{array}{c}(H,A,tb),\underline{y} \to_{\underline{e}} v \qquad (H,A,tb),\underline{z_1} \to_{\underline{e}} v_1 \qquad (H,A,tb),\underline{z_2} \to_{\underline{e}} v_2 \qquad v \in Loc \qquad IsCallable(H,v) \qquad v_2 \in Loc\\ H(v).[[\mathsf{Code}]] = fv \qquad \{\} :: H(v).[[\mathsf{Scope}]] = A'\\ NewLoc() = l \qquad H' = H[l \mapsto NewArgObj(v, ParamsSize(fv), H(v_2))]\\ CreateBinding(H',A',arguments,\mathtt{eval}) = (H'',A'') \qquad SetBinding(H'',A'',arguments,l,\mathtt{strict}) = (H^f,A^f,v^f)\\ GetThis(H^f,v_1) = (H^t,l^t) \qquad (H^t,A^f,l^t),GetBody(fv) \to_{\underline{s}} (H^b,A^b),\mathtt{Normal}(\_)\\ (H^b,A,tb),\underline{x} = \mathtt{undefined} \to_{\underline{s}} (H^r,A^r),ct\end{array}}{(H,A,tb),\underline{x} = \underline{y}(\underline{z_1},\underline{z_2}) \to_{\underline{s}} (H^r,A^r),ct}$$

IRInternalCall(IRId lhs, IRId fun, IRExpr first, Option<IRId> second)

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toObject}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad \mathit{ToObject}(H,v)=(H',\mathit{err})}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toObject}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad \mathit{ToObject}(H,v)=(H',l)\qquad (H',A,\mathit{tb}),\underline{x}=l\rightarrow_{\underline{s}}(H'',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toObject}(e)\rightarrow_{\underline{s}}(H'',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{isObject}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad v\in\mathit{Loc}\qquad (H,A,\mathit{tb}),\underline{x}=\mathtt{true}\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{isObject}(e)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad v\notin\mathit{Loc}\qquad (H,A,\mathit{tb}),\underline{x}=\mathtt{false}\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{isObject}(e)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toString}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad (H,A,\mathit{tb}),\underline{x}=\mathit{ToString}(H,v)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toString}(e)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toNumber}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad (H,A,\mathit{tb}),\underline{x}=\mathit{ToNumber}(H,v)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toNumber}(e)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toBoolean}(e)\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\rightarrow_{\underline{e}}v\qquad (H,A,\mathit{tb}),\underline{x}=\mathit{ToBoolean}(v)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{toBoolean}(e)\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{\mathit{Lookup}(H,A,\underline{y},\mathtt{strict})=l\qquad (H,A,\mathit{tb}),\underline{x}=l\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{getBase}(\underline{y})\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{\mathit{Lookup}(H,A,\underline{y},\mathtt{strict})=\sigma\qquad (H,A,\mathit{tb}),\underline{x}=\#\mathtt{Global}\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{getBase}(\underline{y})\rightarrow_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{y}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{iteratorInit}(\underline{y})\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{y}\rightarrow_{\underline{e}}v\qquad v\notin\mathit{Loc}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{iteratorInit}(\underline{y})\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{y}\rightarrow_{\underline{e}}v\qquad v\in\mathit{Loc}\qquad \mathit{NewLoc}()=l\qquad H'=H[l\mapsto\mathit{IteratorInit}(\mathit{CollectProps}(H,v))]\\ (H',A,\mathit{tb}),\underline{x}=l\rightarrow_{\underline{s}}(H'',A'),\mathit{ct}\end{array}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{iteratorInit}(\underline{y})\rightarrow_{\underline{s}}(H'',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{y}\rightarrow_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{iteratorHasNext}(\underline{y},\underline{z})\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{y}\rightarrow_{\underline{e}}v_1\qquad v_1\notin\mathit{Loc}}{(H,A,\mathit{tb}),\underline{x}=\diamond\mathtt{iteratorHasNext}(\underline{y},\underline{z})\rightarrow_{\underline{s}}(H,A),\mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v \qquad v \in \mathit{Loc} \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorHasNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad v_1 \in \mathit{Loc} \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad v_2 \notin \mathit{Loc}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorHasNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{\begin{array}{c}(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad v_1 \in \mathit{Loc} \qquad v_2 \in \mathit{Loc} \\ \mathit{Next}(H, H(v_2), H(v_2).@\mathtt{property}(\text{“}@\mathtt{i}\text{”}), v_1) \in \mathit{Dom}(H(v_2)) \qquad (H, A, tb), \underline{x} = \mathtt{true} \rightarrow_{\underline{s}} (H', A'), \mathit{ct}\end{array}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorHasNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H', A'), \mathit{ct}}$$

$$\frac{\begin{array}{c}(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad v_1 \in \mathit{Loc} \qquad v_2 \in \mathit{Loc} \\ \mathit{Next}(H, H(v_2), H(v_2).@\mathtt{property}(\text{“}@\mathtt{i}\text{”}), v_1) \notin \mathit{Dom}(H(v_2)) \qquad (H, A, tb), \underline{x} = \mathtt{false} \rightarrow_{\underline{s}} (H', A'), \mathit{ct}\end{array}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorHasNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H', A'), \mathit{ct}}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad v_1 \notin \mathit{Loc}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v \qquad v \in \mathit{Loc} \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad v_1 \in \mathit{Loc} \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad v_2 \notin \mathit{Loc}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathtt{TypeError})}$$

$$\frac{\begin{array}{c}(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v_1 \qquad (H, A, tb), \underline{z} \rightarrow_{\underline{e}} v_2 \qquad v_1 \in \mathit{Loc} \qquad v_2 \in \mathit{Loc} \qquad \mathit{Next}(H, H(v_2), H(v_2).@\mathtt{property}(\text{“}@\mathtt{i}\text{”}), v_1) = i \\ H' = H[v_2 \mapsto H(v_2)[@\mathtt{property} \mapsto H(v_2).@\mathtt{property}[\text{“}@\mathtt{i}\text{”} \mapsto i + 1]]] \qquad (H', A, tb), \underline{x} = H(v_2).@\mathtt{property}(\text{“}\mathtt{i}\text{”}) \rightarrow_{\underline{s}} (H'', A')\end{array}}{(H, A, tb), \underline{x} = \diamond\mathtt{iteratorNext}(\underline{y}, \underline{z}) \rightarrow_{\underline{s}} (H'', A'), \mathit{ct}}$$

IRNew(IRId lhs, IRId fun, List<IRId> args)

11.2.2 The new Operator

$$\frac{(H, A, tb), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} = \mathsf{new}\ \underline{e}(\underline{e_1},\underline{e_2}) \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, tb), \underline{x} = \mathsf{new}\ \underline{e} \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} = \mathsf{new}\ \underline{e} \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

$$\frac{(H, A, tb), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, tb), \underline{x} = \mathsf{new}\ \underline{e} \rightarrow_{\underline{s}} (H, A), \mathtt{Throw}(\mathit{err})}$$

IRFunExpr(IRId lhs, IRFunctional ftn)

11.2.5 Function Expressions

$$\frac{\begin{array}{c}\mathit{CreateImmutableBinding}(H, \{\} :: A, \underline{f}, \mathtt{eval}) = (H', A') \qquad \mathit{NewFtnObject}(H', A', \mathsf{function}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H'', l) \\ \mathit{InitializeImmutableBinding}(H'', A', \underline{f}, l, \mathtt{strict}) = (H^f, A^{f'}, \mathit{err}) \qquad \mathit{er} :: A^f = A^{f'}\end{array}}{(H, A, tb), \underline{x} = \mathsf{function}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H^f, A^f), \mathtt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c}\mathit{CreateImmutableBinding}(H, \{\} :: A, \underline{f}, \mathtt{eval}) = (H', A') \qquad \mathit{NewFtnObject}(H', A', \mathsf{function}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H'', l) \\ \mathit{InitializeImmutableBinding}(H'', A', \underline{f}, l, \mathtt{strict}) = (H^f, A^{f'}, v^f) \qquad \mathit{er} :: A^f = A^{f'} \qquad (H^f, A^f, tb), \underline{x} = l \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}\end{array}}{(H, A, tb), \underline{x} = \mathsf{function}\ \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H''', A''), \mathit{ct}}$$

```
IRFunDecl(IRFunctional ftn)
IRFunctional(IRId name, List<IRId> params, List<IRStmt> args,
             List<IRFunDecl> fds, List<IRVarStmt> vds, List<IRStmt> body)
```

13 Function Definition

10.5 Declaration Binding Instantiation–Step 5

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \neg\mathit{HasBinding}(A, \underline{f}) \\ \mathit{CreateBinding}(H^f, A, \underline{f}, \texttt{eval}) = (H', A') \qquad \mathit{SetBinding}(H', A', \underline{f}, l, \texttt{strict}) = (H'', A'', \mathit{err}) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H'', A''), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \neg\mathit{HasBinding}(A, \underline{f}) \\ \mathit{CreateBinding}(H^f, A, \underline{f}, \texttt{eval}) = (H', A') \qquad \mathit{SetBinding}(H', A', \underline{f}, l, \texttt{strict}) = (H'', A'', v) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(\texttt{empty})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad p.[[\texttt{Configurable}]] \\ \{[[\texttt{Value}]] : \texttt{undefined}, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{eval}\} = \mathit{dp} \\ \mathit{DefineOwnProperty}(H^f, A, \#\texttt{Global}, \underline{f}, \mathit{dp}, \texttt{true}) = (H', A', \mathit{err}) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad p.[[\texttt{Configurable}]] \\ \{[[\texttt{Value}]] : \texttt{undefined}, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{eval}\} = \mathit{dp} \\ \mathit{DefineOwnProperty}(H^f, A, \#\texttt{Global}, \underline{f}, \mathit{dp}, \texttt{true}) = (H', A', v) \\ \mathit{SetBinding}(H', A', \underline{f}, l, \texttt{strict}) = (H'', A'', \mathit{err}) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H'', A''), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad p.[[\texttt{Configurable}]] \\ \{[[\texttt{Value}]] : \texttt{undefined}, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{eval}\} = \mathit{dp} \\ \mathit{DefineOwnProperty}(H^f, A, \#\texttt{Global}, \underline{f}, \mathit{dp}, \texttt{true}) = (H', A', v) \\ \mathit{SetBinding}(H', A', \underline{f}, l, \texttt{strict}) = (H'', A'', v') \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(\texttt{empty})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad \neg p.[[\texttt{Configurable}]] \wedge (p \in \mathit{AccessorProp} \vee (\neg p.[[\texttt{Writable}]] \vee \neg p.[[\texttt{Enumerable}]])) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H^f, A), \texttt{Throw}(\texttt{TypeError})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad \neg p.[[\texttt{Configurable}]] \wedge \neg(p \in \mathit{AccessorProp} \vee (\neg p.[[\texttt{Writable}]] \vee \neg p.[[\texttt{Enumerable}]])) \\ \mathit{SetBinding}(H^f, A, \underline{f}, l, \texttt{strict}) = (H', A', \mathit{err}) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \qquad \mathit{HasBinding}(A, \underline{f}) \qquad A = \#\texttt{Global} \\ \mathit{GetProperty}(H, \#\texttt{Global}, \underline{f}) = (p, \_) \qquad \neg p.[[\texttt{Configurable}]] \wedge \neg(p \in \mathit{AccessorProp} \vee (\neg p.[[\texttt{Writable}]] \vee \neg p.[[\texttt{Enumerable}]])) \\ \mathit{SetBinding}(H^f, A, \underline{f}, l, \texttt{strict}) = (H', A', v) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Normal}(\texttt{empty})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \\ \mathit{HasBinding}(A, \underline{f}) \qquad A \neq \#\texttt{Global} \qquad \mathit{SetBinding}(H^f, A, \underline{f}, l, \texttt{strict}) = (H', A', \mathit{err}) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Throw}(\mathit{err})}$$

$$\frac{\begin{array}{c} \mathit{NewFtnObject}(H, A, \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\}) = (H^f, l) \\ \mathit{HasBinding}(A, \underline{f}) \qquad A \neq \#\texttt{Global} \qquad \mathit{SetBinding}(H^f, A, \underline{f}, l, \texttt{strict}) = (H', A', v) \end{array}}{(H, A, \mathit{tb}), \text{function } \underline{f}(\underline{\mathit{this}},\underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Normal}(\texttt{empty})}$$

`IREval(IRId lhs, IRExpr arg)`

`IRBreak(IRId label)`

12.8 The `break` Statement

$(H, A, \mathit{tb}), \textsf{break}\ \underline{x} \rightarrow_{\underline{s}} (H, A), \texttt{Break}(\texttt{empty}, \underline{x})$

`IRReturn(Option<IRExpr> expr)`

12.9 The `return` Statement

$(H, A, \mathit{tb}), \textsf{return}\ \ \rightarrow_{\underline{s}} (H, A), \texttt{Return}(\texttt{undefined})$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \textsf{return}\ \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}), \textsf{return}\ \underline{e} \rightarrow_{\underline{s}} (H, A), \texttt{Return}(v)}$$

`IRWith(IRId id, IRStmt stmt)`

12.10 The `with` Statement

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \textsf{with}\ (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H, A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v \qquad \mathit{ToObject}(H, v) = (H', \mathit{err})}{(H, A, \mathit{tb}), \textsf{with}\ (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H', A), \texttt{Throw}(\mathit{err})}$$

$$\frac{(H, A, \mathit{tb}), \underline{x} \rightarrow_{\underline{e}} v \qquad \mathit{ToObject}(H, v) = (H', l) \qquad (H', l :: A, \mathit{tb}), \underline{s} \rightarrow_{\underline{s}} (H'', A'), \mathit{ct}}{(H, A, \mathit{tb}), \textsf{with}\ (\underline{x})\ \underline{s} \rightarrow_{\underline{s}} (H'', A), \mathit{ct}}$$

`IRLabelStmt(IRId label, IRStmt stmt)`

12.12 Labelled Statements

$$\frac{(H, A, \mathit{tb}), \underline{s} \rightarrow_{\underline{s}} (H', A'), \texttt{Break}(v, \underline{x})}{(H, A, \mathit{tb}), \underline{x}\colon \{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \texttt{Normal}(v)}$$

$$\frac{(H, A, \mathit{tb}), \underline{s} \rightarrow_{\underline{s}} (H', A'), \mathit{ct} \qquad \mathit{ct} \neq \texttt{Break}(v, \underline{x})}{(H, A, \mathit{tb}), \underline{x}\colon \{\underline{s}\} \rightarrow_{\underline{s}} (H', A'), \mathit{ct}}$$

`IRVarStmt(IRId lhs)`

10.5 Declaration Binding Instantiation–Step 8: $\texttt{CreateMutableBinding}(N, D)$
$\texttt{SetMutableBinding}(N, V, S)$

12.2 Variable Statement

$$\frac{\mathit{CreateBinding}(H, A, \underline{x}, \texttt{eval}) = (H', A') \qquad \mathit{SetBinding}(H', A', \underline{x}, \texttt{undefined}, \texttt{strict}) = (H'', A'', \mathit{err})}{(H, A, \mathit{tb}), \textsf{var}\ \underline{x} \rightarrow_{\underline{s}} (H'', A''), \texttt{Throw}(\mathit{err})}$$

$$\frac{\mathit{CreateBinding}(H, A, \underline{x}, \texttt{eval}) = (H', A') \qquad \mathit{SetBinding}(H', A', \underline{x}, \texttt{undefined}, \texttt{strict}) = (H'', A'', v)}{(H, A, \mathit{tb}), \textsf{var}\ \underline{x} \rightarrow_{\underline{s}} (H'', A''), \texttt{Normal}(\texttt{empty})}$$

`IRThrow(IRExpr expr)`

12.13 The `throw` Statement

$$\frac{(H,A,\mathit{tb}),\underline{e}\to_{\underline{e}}\mathit{ve}}{(H,A,\mathit{tb}),\textsf{throw }\underline{e}\to_{\underline{s}}(H,A),\texttt{Throw}(\mathit{ve})}$$

`IRSeq(List<IRStmt> stmts)`

12.1 Block

$(H,A,\mathit{tb}),\epsilon\to_{\underline{s}}(H,A),\texttt{Normal}(\texttt{empty})$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{ac}}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H',A'),\mathit{ac}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{nc}\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Throw}(\mathit{ve})}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Throw}(\mathit{ve})}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\texttt{Normal}(\mathit{vt})\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Break}(\texttt{empty},x)}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Break}(\mathit{vt},x)}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{nc}\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Break}(v,x)}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Break}(v,x)}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{nc}\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Normal}(\texttt{empty})}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\mathit{nc}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{nc}\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Normal}(v)}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Normal}(v)}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\texttt{Normal}(\mathit{vt})\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Return}(\texttt{empty})}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Return}(\mathit{vt})}$$

$$\frac{(H,A,\mathit{tb}),\underline{s}\to_{\underline{s}}(H',A'),\mathit{nc}\qquad(H',A',\mathit{tb}),\underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Return}(v)}{(H,A,\mathit{tb}),\underline{s}\ \underline{s}^*\to_{\underline{s}}(H'',A''),\texttt{Return}(v)}$$

`IRIf(IRExpr expr, IRStmt trueB, Option<IRStmt> falseB)`

12.5 The `if` Statement

$$\frac{(H,A,\mathit{tb}),\underline{e}\to_{\underline{e}}\mathit{err}}{(H,A,\mathit{tb}),\textsf{if }(\underline{e})\textsf{ then }\underline{s_1}\ (\textsf{else }\underline{s_2})^?\to_{\underline{s}}(H,A),\texttt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\to_{\underline{e}}v\qquad\mathit{ToBoolean}(v)=\texttt{true}\qquad(H,A,\mathit{tb}),s_1\to_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\textsf{if }(\underline{e})\textsf{ then }\underline{s_1}\ (\textsf{else }\underline{s_2})^?\to_{\underline{s}}(H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\to_{\underline{e}}v\qquad\mathit{ToBoolean}(v)=\texttt{false}}{(H,A,\mathit{tb}),\textsf{if }(\underline{e})\textsf{ then }\underline{s_1}\to_{\underline{s}}(H,A),\texttt{Normal}(\texttt{empty})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e}\to_{\underline{e}}v\qquad\mathit{ToBoolean}(v)=\texttt{false}\qquad(H,A,\mathit{tb}),s_2\to_{\underline{s}}(H',A'),\mathit{ct}}{(H,A,\mathit{tb}),\textsf{if }(\underline{e})\textsf{ then }\underline{s_1}\textsf{ else }\underline{s_2}\to_{\underline{s}}(H',A'),\mathit{ct}}$$

`IRWhile(IRExpr cond, IRStmt body)`

12.6.2 The `while` Statement

$$\frac{(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} \mathit{err}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H,A),\texttt{Throw}(\mathit{err})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} v \qquad \mathit{ToBoolean}(v) = \texttt{false}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H,A),\texttt{Normal}(\texttt{empty})}$$

$$\frac{(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} v \qquad \mathit{ToBoolean}(v) = \texttt{true} \qquad (H,A,\mathit{tb}),\underline{s} \to_{\underline{s}} (H',A'),\mathit{ac}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H',A'),\mathit{ac}}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} v \qquad \mathit{ToBoolean}(v) = \texttt{true} \qquad (H,A,\mathit{tb}),\underline{s} \to_{\underline{s}} (H',A'),\mathit{nc}\\ (H',A',\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\mathit{ac}\end{array}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\mathit{ac}}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} v \qquad \mathit{ToBoolean}(v) = \texttt{true} \qquad (H,A,\mathit{tb}),\underline{s} \to_{\underline{s}} (H',A'),\mathit{nc}\\ (H',A',\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\texttt{Normal}(\texttt{empty})\end{array}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\mathit{nc}}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{e} \to_{\underline{e}} v \qquad \mathit{ToBoolean}(v) = \texttt{true} \qquad (H,A,\mathit{tb}),\underline{s} \to_{\underline{s}} (H',A'),\mathit{nc}\\ (H',A',\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\texttt{Normal}(v)\end{array}}{(H,A,\mathit{tb}),\textsf{while }(\underline{e})\ \underline{s} \to_{\underline{s}} (H'',A''),\texttt{Normal}(v)}$$

`IRTry(IRStmt body, Option<IRId> name, Option<IRStmt> catchB,`
`Option<IRStmt> finallyB)`

12.14 The `try` Statement

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\texttt{Throw}(\mathit{ve}) \qquad \mathit{CreateBinding}(H',\{\} :: A',x,\texttt{eval}) = (H'',A'')\\ \mathit{SetBinding}(H'',A'',x,\mathit{ExnLoc}(\mathit{ve}),\texttt{false}) = (H^x,A^x,\mathit{err}') \qquad A^x = \mathit{er} :: A^f\end{array}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\} \to_{\underline{s}} (H^x,A^f),\texttt{Throw}(\mathit{err}')}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\texttt{Throw}(\mathit{ve}) \qquad \mathit{CreateBinding}(H',\{\} :: A',x,\texttt{eval}) = (H'',A'')\\ \mathit{SetBinding}(H'',A'',x,\mathit{ExnLoc}(\mathit{ve}),\texttt{false}) = (H^x,A^x,v) \qquad (H^x,A^x,\mathit{tb}),\underline{s_2} \to_{\underline{s}} (H^c,A^c),\mathit{ct} \qquad A^c = \mathit{er} :: A^f\end{array}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\} \to_{\underline{s}} (H^c,A^f),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\mathit{ct} \qquad \mathit{ct} \neq \texttt{Throw}(\mathit{ve})}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\} \to_{\underline{s}} (H',A'),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\mathit{ct} \qquad (H',A',\mathit{tb}),\underline{s_2} \to_{\underline{s}} (H'',A''),\mathit{nc}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ finally }\{\underline{s_2}\} \to_{\underline{s}} (H'',A''),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\mathit{ct} \qquad (H',A',\mathit{tb}),\underline{s_2} \to_{\underline{s}} (H'',A''),\mathit{ac}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ finally }\{\underline{s_2}\} \to_{\underline{s}} (H'',A''),\mathit{ac}}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\texttt{Throw}(\mathit{ve}) \qquad \mathit{CreateBinding}(H',\{\} :: A',x,\texttt{eval}) = (H'',A'')\\ \mathit{SetBinding}(H'',A'',x,\mathit{ExnLoc}(\mathit{ve}),\texttt{false}) = (H^x,A^x,\mathit{err}') \qquad A^x = \mathit{er} :: A^f\end{array}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\}\textsf{ finally }\{\underline{s_3}\} \to_{\underline{s}} (H^x,A^f),\texttt{Throw}(\mathit{err}')}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\texttt{Throw}(\mathit{ve}) \qquad \mathit{CreateBinding}(H',\{\} :: A',x,\texttt{eval}) = (H'',A'')\\ \mathit{SetBinding}(H'',A'',x,\mathit{ExnLoc}(\mathit{ve}),\texttt{false}) = (H^x,A^x,v) \qquad (H^x,A^x,\mathit{tb}),\underline{s_2} \to_{\underline{s}} (H^c,A^c),\mathit{ct}\\ A^c = \mathit{er} :: A^{c'} \qquad (H^c,A^{c'},\mathit{tb}),\underline{s_3} \to_{\underline{s}} (H^f,A^f),\mathit{nc}\end{array}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\}\textsf{ finally }\{\underline{s_3}\} \to_{\underline{s}} (H^f,A^f),\mathit{ct}}$$

$$\frac{\begin{array}{c}(H,A,\mathit{tb}),\underline{s_1} \to_{\underline{s}} (H',A'),\texttt{Throw}(\mathit{ve}) \qquad \mathit{CreateBinding}(H',\{\} :: A',x,\texttt{eval}) = (H'',A'')\\ \mathit{SetBinding}(H'',A'',x,\mathit{ExnLoc}(\mathit{ve}),\texttt{false}) = (H^x,A^x,v) \qquad (H^x,A^x,\mathit{tb}),\underline{s_2} \to_{\underline{s}} (H^c,A^c),\mathit{ct}\\ A^c = \mathit{er} :: A^{c'} \qquad (H^c,A^{c'},\mathit{tb}),\underline{s_3} \to_{\underline{s}} (H^f,A^f),\mathit{ac}\end{array}}{(H,A,\mathit{tb}),\textsf{try }\{\underline{s_1}\}\textsf{ catch }(\underline{x})\{\underline{s_2}\}\textsf{ finally }\{\underline{s_3}\} \to_{\underline{s}} (H^f,A^f),\mathit{ac}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s_1} \rightarrow_{\underline{s}} (H',A'),\mathit{ct} \qquad \mathit{ct} \neq \texttt{Throw}(\mathit{ve}) \qquad (H',A',\mathit{tb}),\underline{s_3} \rightarrow_{\underline{s}} (H'',A''),\mathit{nc}}{(H,A,\mathit{tb}),\textsf{try}\ \{\underline{s_1}\}\ \textsf{catch}\ (\underline{x})\{\underline{s_2}\}\ \textsf{finally}\ \{\underline{s_3}\} \rightarrow_{\underline{s}} (H'',A''),\mathit{ct}}$$

$$\frac{(H,A,\mathit{tb}),\underline{s_1} \rightarrow_{\underline{s}} (H',A'),\mathit{ct} \qquad \mathit{ct} \neq \texttt{Throw}(\mathit{ve}) \qquad (H',A',\mathit{tb}),\underline{s_3} \rightarrow_{\underline{s}} (H'',A''),\mathit{ac}}{(H,A,\mathit{tb}),\textsf{try}\ \{\underline{s_1}\}\ \textsf{catch}\ (\underline{x})\{\underline{s_2}\}\ \textsf{finally}\ \{\underline{s_3}\} \rightarrow_{\underline{s}} (H'',A''),\mathit{ac}}$$

```
IRStmtUnit(List<IRStmt> stmts)
```

5.4.3 Expressions

$$\boxed{(H,A,\mathit{tb}),\underline{e} \rightarrow_{\underline{e}} \mathit{ve}}$$

```
IRBin(IRExpr first, IROp op, IRExpr second)
```

11.8.6 The `instanceof` operator

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} \mathit{err}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \notin \mathit{Loc}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \texttt{TypeError}}$$

15.3.5.3 `[[HasInstance]](V)`

Instead of checking whether v_2 has a `[[HasInstance]]` internal method, we check whether v_2 is a function object.

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \in \mathit{Loc} \qquad \neg \mathit{IsCallable}(H,v_2)}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \texttt{TypeError}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \in \mathit{Loc} \qquad \mathit{IsCallable}(H,v_2) \qquad v_1 \notin \mathit{Loc}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \texttt{false}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \in \mathit{Loc} \qquad \mathit{IsCallable}(H,v_2) \qquad v_1 \in \mathit{Loc}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{instanceof}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{Inherit}(H,v_1,v_2)}$$

11.8.7 The `in` operator

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} \mathit{err}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{in}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{in}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \notin \mathit{Loc}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{in}\ \underline{e_2} \rightarrow_{\underline{e}} \texttt{TypeError}}$$

$$\frac{(H,A,\mathit{tb}),\underline{e_1} \rightarrow_{\underline{e}} v_1 \qquad (H,A,\mathit{tb}),\underline{e_2} \rightarrow_{\underline{e}} v_2 \qquad v_2 \in \mathit{Loc}}{(H,A,\mathit{tb}),\underline{e_1}\ \texttt{in}\ \underline{e_2} \rightarrow_{\underline{e}} \mathit{HasProperty}(H,v_2,\mathit{ToString}(H,v_1))}$$

`IRUn(IROp op, IRExpr expr)`

11.4.2 The `void` Operator

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \texttt{void}\ \underline{e} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}), \texttt{void}\ \underline{e} \rightarrow_{\underline{e}} \texttt{undefined}}$$

11.4.3 The `typeof` Operator

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}), \texttt{typeof}\ \underline{e} \rightarrow_{\underline{e}} \texttt{undefined}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}), \texttt{typeof}\ \underline{e} \rightarrow_{\underline{e}} \mathit{TypeTag}(H, v)}$$

11.4.6 Unary + Operator

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}),\ + \underline{e} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}),\ + \underline{e} \rightarrow_{\underline{e}} \mathit{ToNumber}(H, v)}$$

11.4.7 Unary − Operator

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}),\ - \underline{e} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}),\ - \underline{e} \rightarrow_{\underline{e}} \mathit{Negate}(\mathit{ToNumber}(H, v))}$$

11.4.8 Bitwise NOT Operator (∼)

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}),\ \sim \underline{e} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}),\ \sim \underline{e} \rightarrow_{\underline{e}} \sim (\mathit{ToInt32}(H, v))}$$

11.4.9 Logical NOT Operator (!)

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, \mathit{tb}),\ !\ \underline{e} \rightarrow_{\underline{e}} \mathit{err}}$$

$$\frac{(H, A, \mathit{tb}), \underline{e} \rightarrow_{\underline{e}} v}{(H, A, \mathit{tb}),\ !\ \underline{e} \rightarrow_{\underline{e}}\ \mathit{Negate}(\mathit{ToBoolean}(v))}$$

IRLoad(IRId obj, IRExpr index)

11.2.1 Property Accessors: IRLoad

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} err}{(H, A, tb),\ \underline{x}[\underline{e}] \rightarrow_{\underline{e}} err}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v \qquad (H, A, tb), \underline{e} \rightarrow_{\underline{e}} err}{(H, A, tb),\ \underline{x}[\underline{e}] \rightarrow_{\underline{e}} err}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, tb), \underline{e} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) = err}{(H, A, tb),\ \underline{x}[\underline{e}] \rightarrow_{\underline{e}} err}$$

$$\frac{(H, A, tb), \underline{x} \rightarrow_{\underline{e}} v_1 \qquad (H, A, tb), \underline{e} \rightarrow_{\underline{e}} v_2 \qquad \mathit{CheckObjectCoercible}(v_1) \neq err \qquad \mathit{ToObject}(H, v_1) = (H', l)}{(H, A, tb),\ \underline{x}[\underline{e}] \rightarrow_{\underline{e}} \mathit{Get}(H', l, \mathit{ToString}(H', v_2))}$$

IRUserId(String text)
IRTmpId(String text)

11.1.2 Identifier Reference
10.3.1 Identifier Resolution
10.2.2.1 GetIdentifierReference(lex,name,strict)
$(H, A, tb), \underline{x} \rightarrow_{\underline{e}} \mathit{GetBindingValue}(H, \mathit{Lookup}(H, A, \underline{x}, \texttt{strict}), \underline{x}, \texttt{strict})$

IRNumber(ignoreForEquals String text, Double num)

11.1.3 Literal Reference
7.8 Literals
$(H, A, tb), n \rightarrow_{\underline{e}} n$

IRString(String str)

$(H, A, tb), s \rightarrow_{\underline{e}} s$

IRBool(boolean bool)

$(H, A, tb), \mathsf{true} \rightarrow_{\underline{e}} \texttt{true}$
$(H, A, tb), \mathsf{false} \rightarrow_{\underline{e}} \texttt{false}$

IRUndef()

$(H, A, tb), \mathsf{undefined} \rightarrow_{\underline{e}} \texttt{undefined}$

IRNull()

$(H, A, tb), \mathsf{null} \rightarrow_{\underline{e}} \texttt{null}$

IRThis()

11.1.1 The this Keyword
$(H, A, tb), \mathsf{this} \rightarrow_{\underline{e}} tb$

5.4.4 Members

$\boxed{(H, A, tb), \underline{m} \rightarrow_{\underline{m}} (H, \underline{x}, ov) \text{ or } \mathit{err}}$

IRField(IRId prop, IRExpr expr)

11.1.5 Object Initialiser

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} \mathit{err}}{(H, A, tb), \underline{x} : \underline{y} \rightarrow_{\underline{m}} \mathit{err}}$$

$$\frac{(H, A, tb), \underline{y} \rightarrow_{\underline{e}} v \qquad \{[[\texttt{Value}]] : v, [[\texttt{Writable}]] : \texttt{true}, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\} = dp}{(H, A, tb), \underline{x} : \underline{y} \rightarrow_{\underline{m}} (H, \underline{x}, dp)}$$

IRGetProp(IRFunctional ftn)

$$\frac{\mathit{NewFtnObject}(H, A, \texttt{get}\ \underline{f}(\underline{\mathit{this}}, \underline{\mathit{arguments}})\{\underline{s}\}) = (H', l) \qquad \{[[\texttt{Get}]] : l, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\} = ap}{(H, A, tb), \texttt{get}\ \underline{f}(\underline{\mathit{this}}, \underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{m}} (H', \underline{f}, ap)}$$

IRSetProp(IRFunctional ftn)

$$\frac{\mathit{NewFtnObject}(H, A, \texttt{set}\ \underline{f}(\underline{\mathit{this}}, \underline{\mathit{arguments}})\{\underline{s}\}) = (H', l) \qquad \{[[\texttt{Set}]] : l, [[\texttt{Enumerable}]] : \texttt{true}, [[\texttt{Configurable}]] : \texttt{true}\} = ap}{(H, A, tb), \texttt{set}\ \underline{f}(\underline{\mathit{this}}, \underline{\mathit{arguments}})\{\underline{s}\} \rightarrow_{\underline{m}} (H', \underline{f}, ap)}$$

Chapter 6

CFG

6.1 Settings

.../jsaf/analysis/cfg/{package, CFG, CFGId}.scala

$$\begin{array}{rcllcl}
P & \in & \mathsf{Program} & = & \wp(\mathsf{FunctionId} \times \mathsf{ArgumentsName} \times \mathsf{ArgVars} \times \mathsf{LocalVars}) \times \mathsf{Graph}\\
fid & \in & \mathsf{FunctionId} & ::= & fid_{global} \mid fid_1 \mid \cdots \\
 & & \mathsf{VarKind} & ::= & \mathsf{GlobalVar} \mid \mathsf{PureLocalVar} \mid \mathsf{CapturedVar} \mid \mathsf{CapturedCatchVar}\\
 & & \mathsf{ArgVars}, \mathsf{LocalVars} & ::= & x^*\\
 & & \mathsf{ArgumentsName} & = & \mathsf{String}\\
G, \langle \mathbb{C}, \hookrightarrow, \overset{\mathsf{exc}}{\hookrightarrow}, \mathbb{A}\rangle & \in & \mathsf{Graph} & = & \wp(\mathsf{Node}) \times \wp(\mathsf{Edge}) \times \wp(\mathsf{Edge}) \times \wp(\mathsf{Call})\\
n & \in & \mathsf{Node} & = & \mathsf{FunctionId}\times\mathsf{Label}\\
 & & \mathsf{Edge},\mathsf{Call} & = & \mathsf{Node}\times\mathsf{Node}\\
 & & \mathsf{Label} & ::= & \mathsf{ENTRY} \mid \mathsf{EXIT} \mid \mathsf{EXIT\text{-}EXC} \mid c_1 \mid \cdots\\
 & & \texttt{Label} & = & \texttt{LEntry} \mid \texttt{LExit} \mid \texttt{LExitExc} \mid \texttt{LBlock(id : BlockId)}
\end{array}$$

A call expression splits into a pair of call and after-call nodes in this flow graph, and there is no edge between the pair. In order to treat them as a call-site and a return-site of the call, the pair $(cp_{\mathit{call}}, cp_{\mathit{after\text{-}call}})$ must be recorded in $\mathbb{A}$ as an element.

6.2 Helper Functions

.../jsaf/analysis/cfg/CFG.scala

$$\begin{array}{lcl}
\underline{\mathsf{getCmd}}_P & : & \mathsf{Node} \rightarrow \mathsf{Command}\\
\underline{\mathsf{getArgVars}}_P & : & \mathsf{FunctionId} \rightarrow \mathsf{ArgVars}\\
\underline{\mathsf{getLocalVars}}_P & : & \mathsf{FunctionId} \rightarrow \mathsf{LocalVars}\\
\underline{\mathsf{getCallFromAftercall}}_P & : & \mathsf{Node} \rightarrow \mathsf{Node}\\
\underline{\mathsf{getAftercallFromCall}}_P & : & \mathsf{Node} \rightarrow \mathsf{Node}\\
\underline{\mathsf{getExcSucc}}_P & : & \mathsf{Node} \rightarrow \mathsf{Node}\\
\underline{\mathsf{getArgumentsName}}_P & : & \mathsf{FunctionId} \rightarrow \mathsf{String}\\
\underline{\mathsf{getReturnVar}}_P & : & \mathsf{Node} \rightarrow \mathsf{String}\\
\underline{\mathsf{getVarKind}}_P & : & \mathsf{String} \rightarrow \mathsf{VarKind}\\
\underline{\mathsf{isUserFunction}}_P & : & \mathsf{FunctionId} \rightarrow \mathsf{Boolean}
\end{array}$$

6.3 Syntax of Command

.../jsaf/analysis/cfg/{CFG, CFGInst, CFGExpr}.scala

$c \in$ Command	$::=$	$\texttt{entry}$	Entry	*entry node*
	$\mid$	$\texttt{exit}$	Exit	*exit node*
	$\mid$	$\texttt{exit-exc}$	ExitExc	*exit node for exception*
	$\mid$	i^+	Block	*basic block*
$i \in$ Instruction	$::=$	$x \texttt{ := alloc(} e^? \texttt{)}$	CFGAlloc	
	$\mid$	$x \texttt{ := allocArray(} n \texttt{)}$	CFGAllocArray	
	$\mid$	$x \texttt{ := allocArg(} n \texttt{)}$	CFGAllocArg	
	$\mid$	$x \texttt{ := } e$	CFGExprStmt	
	$\mid$	$x \texttt{ := delete(} e \texttt{)}$	CFGDelete	
	$\mid$	$x \texttt{ := delete(} e_1, e_2 \texttt{)}$	CFGDeleteProp	
	$\mid$	$e \texttt{[} e \texttt{] := } e$	CFGStore	
	$\mid$	$x_1 \texttt{ := function } x_2^? \texttt{(} fid \texttt{)}$	CFGFunExpr	
	$\mid$	$\texttt{construct(} e_1, e_2, e_3 \texttt{)}$	CFGConstruct	
	$\mid$	$\texttt{call(} e_1, e_2, e_3 \texttt{)}$	CFGCall	
	$\mid$	$\texttt{assert(} e \otimes e \texttt{)}$	CFGAssert	
	$\mid$	$\texttt{catch(} x \texttt{)}$	CFGCatch	
	$\mid$	$\texttt{return(} e^? \texttt{)}$	CFGReturn	
	$\mid$	$\texttt{throw(} e \texttt{)}$	CFGThrow	
	$\mid$	$x \texttt{ := } \diamond x \texttt{(} x^* \texttt{)}$	CFGInternalCall	
	$\mid$	$\texttt{noop}$	CFGNoOp	
$e \in$ Expression	$::=$	x	CFGVarRef	
	$\mid$	$e \otimes e$	CFGBin	
	$\mid$	$\ominus e$	CFGUn	
	$\mid$	$e \texttt{[} e \texttt{]}$	CFGLoad	
	$\mid$	n	CFGNumber	*Number, double*
	$\mid$	“s”	CFGString	*String*
	$\mid$	$\texttt{true, false}$	CFGBool	*Boolean*
	$\mid$	$\texttt{null}$	CFGNull	
	$\mid$	$\texttt{this}$	CFGThis	
$\ominus$	$::=$	$\texttt{void} \mid \texttt{typeof} \mid \texttt{+} \mid \texttt{-} \mid \texttt{\~{}} \mid \texttt{!}$		
$\otimes$	$::=$	$\texttt{instanceof} \mid \texttt{in} \mid \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \mid \texttt{+} \mid \texttt{-} \mid \star \mid \texttt{/} \mid \texttt{\%} \mid \texttt{==} \mid \texttt{!=} \mid \texttt{===}$		
	$\mid$	$\texttt{!==} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=}$		

Chapter 7

IR to CFG

7.1 Constraints

- There is no instruction after `call` or `return` in a node. There is no instruction before `catch` in a node.
  - $\forall n \in \textit{nodes}.\ (i_k \in n \wedge (i_k = \texttt{call} \vee i_k = \texttt{return})) \rightarrow \neg(\exists i_{k'} \in n.\ k < k')$
  - $\forall n \in \textit{nodes}.\ (i_k \in n \wedge (i_k = \texttt{catch}) \rightarrow \neg(\exists i_{k'} \in n.\ k > k')$
- An `entry` node has no predecessor, `exit` and `exit-exc` nodes have no successor.
  - $\forall (n_1, n_2) \in \hookrightarrow .n_1 \neq \texttt{exit} \wedge n_1 \neq \texttt{exit-exc} \wedge n_2 \neq \texttt{entry}$
- A call expression splits into a pair of call and after-call nodes in this flow graph, and there is no edge between the pair. In order to treat them as a call-site and a return-site of the call, the pair $(cp_{call}, cp_{after\text{-}call})$ must be recorded in $\mathbb{A}$ as an element.
  - $\begin{array}{l} \forall n \in \mathbb{C}.((\textit{LastInstOf}(n) = \texttt{call}) \rightarrow \\ \qquad \exists n' \in \mathbb{C}.((n, n') \in \mathbb{A} \wedge n \not\hookrightarrow n')) \end{array}$
  - $\forall (n_1, n_2), (n_1', n_2') \in \mathbb{A}.\ n_1 = n_1' \Leftrightarrow n_2 = n_2'$

7.2 Translation

.../jsaf/analysis/cfg/CFG.scala

7.2.1 Data Type

$$
G \in \mathsf{CFG} \quad : \quad \left\{
\begin{array}{ccc}
\mathit{nodes} & : & \mathtt{Node\ list} \\
\mathit{succMap} & : & \mathtt{Node} \mapsto \mathtt{Node\ set} \\
\mathit{predMap} & : & \mathtt{Node} \mapsto \mathtt{Node\ set} \\
\mathit{excSuccMap} & : & \mathtt{Node} \mapsto \mathtt{Node} \\
\mathit{excPredMap} & : & \mathtt{Node} \mapsto \mathtt{Node\ set} \\
\mathit{callFromAftercallMap} & : & \mathtt{Node} \mapsto \mathtt{Node} \\
\mathit{aftercallFromCallMap} & : & \mathtt{Node} \mapsto \mathtt{Node} \\
\mathit{callFromAftercatchMap} & : & \mathtt{Node} \mapsto \mathtt{Node} \\
\mathit{aftercatchFromCallMap} & : & \mathtt{Node} \mapsto \mathtt{Node} \\
\mathit{cmdMap} & : & \mathtt{Node} \mapsto \mathtt{Cmd} \\
\mathit{funcMap} & : & \mathtt{FunctionId} \mapsto \mathtt{ArgumentsName} \times \mathtt{ArgVars} \times \mathtt{LocalVars} \\
\mathit{returnVarMap} & : & \mathtt{Node} \mapsto \mathtt{CFGId} \\
 & & \\
\mathit{NewFunction} & : & \mathtt{ArgumentsName} \times \mathtt{ArgVars} \times \mathtt{LocalVars} \to \mathtt{FunctionId} \\
\mathit{NewBlock} & : & \mathtt{FunctionId} \to \mathtt{BlockNode} \\
\mathit{NewAfterCallBlock} & : & \mathtt{FunctionId} \times \mathtt{CFGId} \to \mathtt{BlockNode} \\
\mathit{NewAfterCatchBlock} & : & \mathtt{FunctionId} \to \mathtt{BlockNode} \\
\mathit{AddInst} & : & \mathtt{BlockNode} \times \mathtt{CFGInst} \to \mathtt{Unit} \\
\mathit{AddEdge} & : & \mathtt{Node} \times \mathtt{Node} \to \mathtt{Unit} \\
\mathit{AddEdge} & : & \mathtt{Node\ list} \times \mathtt{Node} \to \mathtt{Unit} \\
\mathit{AddExcEdge} & : & \mathtt{Node} \times \mathtt{Node} \to \mathtt{Unit} \\
\mathit{AddExcEdge} & : & \mathtt{Node\ list} \times \mathtt{Node} \to \mathtt{Unit} \\
\mathit{AddCall} & : & \mathtt{Node} \times \mathtt{Node} \times \mathtt{Node} \to \mathtt{Unit}
\end{array}
\right\}
$$

$$
\begin{array}{rll}
 & \mathtt{Node} & = \mathtt{FunctionId} \times \mathit{Label} \\
fid \in & \mathtt{FunctionId} & = \mathtt{Int} \\
\#name \in & \mathit{Label} & = \{\mathsf{LEntry}, \mathsf{LExit}, \mathsf{LExitExc}\} \cup \mathtt{LabelBlock} \\
 & \mathtt{BlockNode} & = \mathtt{FunctionId} \times \mathtt{LabelBlock} \\
 & \mathtt{LabelBlock} & = \mathtt{Int} \\
 & \mathtt{Cmd} & = \{\mathsf{Entry}, \mathsf{Exit}, \mathsf{ExitExc}\} \cup \mathtt{Block} \\
 & \mathtt{Block} & = \mathtt{CFGInst\ list} \\
 & \mathtt{ArgumentsName} & = \mathtt{String} \\
 & \mathtt{ArgVars} & = \mathtt{CFGId\ list} \\
 & \mathtt{LocalVars} & = \mathtt{CFGId\ list}
\end{array}
$$

7.2.2 CFG Methods

$$
\begin{array}{ll}
\mathit{NewFunction}(\mathit{argsName}, \mathit{argVars}, \mathit{localVars}) = & fid \overset{let}{=} \mathit{newFunctionId}() \\
 & \mathit{funcMap} \leftarrow \mathit{funcMap}[fid \mapsto (\mathit{argsName}, \mathit{argVars}, \mathit{localVars})] \\
 & nodes \leftarrow (fid, \mathsf{LEntry}) :: nodes \\
 & \mathit{cmdMap} \leftarrow \mathit{cmdMap}[(fid, \mathsf{LEntry}) \mapsto \mathsf{Entry}] \\
 & nodes \leftarrow (fid, \mathsf{LExit}) :: nodes \\
 & \mathit{cmdMap} \leftarrow \mathit{cmdMap}[(fid, \mathsf{LExit}) \mapsto \mathsf{Exit}] \\
 & nodes \leftarrow (fid, \mathsf{LExitExc}) :: nodes \\
 & \mathit{cmdMap} \leftarrow \mathit{cmdMap}[(fid, \mathsf{LExitExc}) \mapsto \mathsf{ExitExc}] \\
 & fid \\
 & \\
\mathit{SetGlobalFId}(\mathit{fid}) & = globalFId \leftarrow fid
\end{array}
$$

```
$\mathit{NewBlock}(fid)$ $= \mathit{bid} \overset{\mathit{let}}{=} \mathit{newBlockId}()$
    $\mathit{blockNode} \overset{\mathit{let}}{=} (fid, bid)$
    $nodes \leftarrow \mathit{blockNode} :: nodes$
    $\mathit{cmdMap} \leftarrow \mathit{cmdMap}[\mathit{blockNode} \mapsto [\,]]$
    $blockNode$

$\mathit{NewAfterCallBlock}(fid, x)$ $= \mathit{blockNode} \overset{\mathit{let}}{=} NewBlock(fid)$
    $\mathit{returnVarMap} \leftarrow \mathit{returnVarMap}[\mathit{blockNode} \mapsto x]$
    $blockNode$

$\mathit{NewAfterCatchBlock}(fid)$ $= NewBlock(fid)$

$\mathit{AddInst}(\mathit{blockNode}, \mathit{inst})$ $= \mathit{block} \overset{\mathit{let}}{=} \mathit{cmdMap}(\mathit{blockNode})$
    $\mathit{cmdMap} \leftarrow \mathit{cmdMap}[\mathit{blockNode} \mapsto \mathit{block}@[\mathit{inst}]]$

$\mathit{AddEdge}(n_1, n_2)$ $=$ if $(\mathit{succMap}(n_1) \neq \texttt{null})$ then
        $\mathit{succMap} \leftarrow \mathit{succMap}[n_1 \mapsto \{n_2\} \cup \mathit{succMap}(n_1)]$
        $predMap \leftarrow predMap[n_2 \mapsto \{n_1\} \cup predMap(n_2))$
    else $\mathit{succMap} \leftarrow \mathit{succMap}[n_1 \mapsto \{n_2\}]$
        $predMap \leftarrow predMap[n_2 \mapsto \{n_1\}))$
$\mathit{AddEdge}(N, n_2)$ $= \mathit{Iter}(N)(\lambda\, n \Rightarrow \mathit{AddEdge}(n, n_2))$

$\mathit{AddExcEdge}(n_1, n_2)$ $=$ if $(\mathit{excSuccMap}(n_1) \neq \texttt{null})$ then
        $\mathit{excSuccMap} \leftarrow \mathit{excSuccMap}[n_1 \mapsto \{n_2\} \cup \mathit{excSuccMap}(n_1)]$
        $excPredMap \leftarrow excPredMap[n_2 \mapsto \{n_1\} \cup excPredMap(n_2))$
    else $\mathit{excSuccMap} \leftarrow \mathit{excSuccMap}[n_1 \mapsto \{n_2\}]$
        $excPredMap \leftarrow excPredMap[n_2 \mapsto \{n_1\}))$
$\mathit{AddExcEdge}(N, n_2)$ $= \mathit{Iter}(N)(\lambda\, n \Rightarrow \mathit{AddExcEdge}(n, n_2))$

$\mathit{AddCall}(n_1, n_2)$ $=$ if $(\mathit{aftercallFromCallMap}(n_1) \neq \texttt{null})$ then
        $\mathit{aftercallFromCallMap} \leftarrow \mathit{aftercallFromCallMap}[n_1 \mapsto \{n_2\} \cup \mathit{aftercallFromCallMap}(n_1)]$
        $\mathit{callFromAftercallMap} \leftarrow \mathit{callFromAftercallMap}[n_1 \mapsto \{n_2\} \cup \mathit{callFromAftercallMap}(n_1)]$
    if $(\mathit{aftercatchFromCallMap}(n_1) \neq \texttt{null})$ then
        $\mathit{aftercatchFromCallMap} \leftarrow \mathit{aftercatchFromCallMap}[n_1 \mapsto \{n_3\} \cup \mathit{aftercatchFromCallMap}(n_1)]$
        $\mathit{callFromAftercatchMap} \leftarrow \mathit{callFromAftercatchMap}[n_1 \mapsto \{n_3\} \cup \mathit{callFromAftercatchMap}(n_1)]$
```

7.2.3 Helper Functions

```
$\textit{Fold}(A)(b)(f)$        : $\texttt{Any list} \times \texttt{Any}' \times (\texttt{Any} \times \texttt{Any}' \rightarrow \texttt{Any}') \rightarrow \texttt{Any}'$
                        = if ($\textit{Length}(A) = 0$) then $b$
                          else $\textit{Fold}(\textit{TailOf}(A))(f(\textit{HeadOf}(A), b))(f)$

$\textit{Iter}(A)(f)$           : $\texttt{Any list} \times (\texttt{Any} \rightarrow \texttt{Unit}) \rightarrow \texttt{Unit}$
                        = if ($\textit{Length}(A) = 0$) then $\texttt{unit}$
                          else $f(\textit{HeadOf}(A))$
                               $\textit{Iter}(\textit{TailOf}(A))(f)$

$\textit{GetTail}(G, N)(fid)$: $\mathsf{CFG} \times \texttt{Node list} \times \texttt{FunctionId} \rightarrow \texttt{BlockNode}$
                        = if ($\textit{Length}(N) = 1$) then
                               $\textit{HeadOf}(N)$
                          else if ($\textit{Length}(N) = 0$) then
                               $n \overset{let}{=} G.\textit{NewBlock}(fid)$
                               $n$
                          else $n \overset{let}{=} G.\textit{NewBlock}(fid)$
                               $G.\textit{AddEdge}(N, n)$
                               $n$

$\textit{ToString}(l)$          : $\texttt{Label} \rightarrow \texttt{String}$
                        = $l.getId().getText()$
```

7.2.4 Translation Rules

$$
\begin{array}{lll}
L & \in & \texttt{LabelMap} : \texttt{String} \mapsto \texttt{Node set} \\
 & & [\![-]\!]_{\textit{root}} \quad : \texttt{IRRoot} \rightarrow \mathsf{CFG} \\
 & & [\![-]\!]_{\textit{fdvars}} \quad : \texttt{IRFunDecl list} \rightarrow \texttt{LocalVars} \\
 & & [\![-]\!]_{\textit{vds}} \quad : \texttt{IRVarStmt list} \rightarrow \texttt{LocalVars} \\
 & & [\![-]\!]_{\textit{args}} \quad : \texttt{IRStmt list} \rightarrow \texttt{ArgVars} \\
 & & [\![-]\!]_{\textit{fd}} \quad : \texttt{IRFunDecl} \times \mathsf{CFG} \times \texttt{Node list} \times \texttt{FunctionId} \rightarrow \texttt{Node list} \\
 & & [\![-]\!]_{\textit{stmt}} \quad : \texttt{IRStmt} \times \mathsf{CFG} \times \texttt{Node list} \times \texttt{LabelMap} \times \texttt{FunctionId} \rightarrow \texttt{Node list} \times \texttt{LabelMap} \\
 & & [\![-]\!]_{\textit{mem}} \quad : \texttt{IRField} \times \mathsf{CFG} \times \texttt{Node} \times \texttt{IRId} \rightarrow \texttt{Unit} \\
 & & [\![-]\!]_{\textit{elem}} \quad : \texttt{IRExpr} \times \mathsf{CFG} \times \texttt{Node} \times \texttt{IRId} \times \texttt{Int} \rightarrow \mathbf{\texttt{Int}}
\end{array}
$$

$$
\begin{aligned}
[\![\texttt{IRRoot}(fd^*, vd^*, stmt^*)]\!]_{root} = {} & G \leftarrow \texttt{new}\ \mathsf{CFG}() \\
& argVars \overset{let}{=} [\,] \\
& localVars \overset{let}{=} [\![fd^*]\!]_{fdvars} \mathbin{@} [\![vd^*]\!]_{vds} \\
& fid_{global} \overset{let}{=} G.NewFunction(\text{“”}, argVars, localVars) \\
& G.SetGlobalFId(fid_{global}) \\
& n_{start} \overset{let}{=} G.NewBlock(fid_{global}) \\
& G.AddEdge((fid_{global}, \mathsf{LEntry}), n_{start}) \\
& N_1 \overset{let}{=} [\![fd^*]\!]_{fd^*}(G, [n_{start}])(fid_{global}) \\
& L \overset{let}{=} [\ \#return \mapsto [\,], \#throw \mapsto [\,], \#throw\_end \mapsto [\,]\,, \#after\_catch \mapsto [\,]\,] \\
& (N_2, L_1) \overset{let}{=} [\![stmt^*]\!]_{stmt*}(G, N_1, L)(fid_{global}) \\
& G.AddEdge(N_2, (fid_{global}, \mathsf{LExit})) \\
& G.AddEdge(L_1(\#return), (fid_{global}, \mathsf{LExit})) \\
& G.AddExcEdge(L_1(\#throw), (fid_{global}, \mathsf{LExitExc})) \\
& G.AddEdge(L_1(\#throw\_end), (fid_{global}, \mathsf{LExitExc})) \\
& G.AddEdge(L_1(\#after\_catch), (fid_{global}, \mathsf{LExitExc})) \\
& G
\end{aligned}
$$

$$
[\![arg^*]\!]_{args} = Fold(arg^*)([\,])(\lambda(args, x = arguments[i]) \Rightarrow args@[x])
$$

$$
[\![fd^*]\!]_{fdvars} = Fold(fd^*)([\,])(\lambda(vars, \substack{\mathsf{function}\ f\,(this, args) \\ \{s^*_{arg}, fd^*, vd^*, s^*_{body}\}}) \Rightarrow vars@[f])
$$

$$
[\![vd^*]\!]_{vds} = Fold(vd^*)([\,])(\lambda(vars, \mathsf{var}\ x) \Rightarrow vars@[x])
$$

$$
\begin{aligned}
[\![fd^*]\!]_{fd^*}(G, N)(fid) = {} & \text{if } (Length(fd^*) = 0) \text{ then } N \\
& \text{else } [\![TailOf(fd^*)]\!]_{fd*}(G, [\![HeadOf(fd^*)]\!]_{fd}(G, N)(fid))(fid)
\end{aligned}
$$

$$
\begin{aligned}
[\![\substack{\mathsf{function}\ f\,(this, args) \\ \{s^*_{arg}, fd^*, vd^*, s^*_{body}\}}]\!]_{fd}(G, N)(fid) = {} & argVars \overset{let}{=} [\![s^*_{arg}]\!]_{args} \\
& localVars \overset{let}{=} [\![fd^*]\!]_{fdvars} \mathbin{@} [\![vd^*]\!]_{vds} - argVars \\
& fid_{new} \overset{let}{=} G.NewFunction(args, argVars, localVars) \\
& n_{start} \overset{let}{=} G.NewBlock(fid_{new}) \\
& G.AddEdge((fid_{new}, \mathsf{LEntry}), n_{start}) \\
& L \overset{let}{=} [\ \#return \mapsto [\,], \#throw \mapsto [\,]\,, \#throw\_end \mapsto [\,]\,, \#after\_catch \mapsto [\,]\,] \\
& N_1 \overset{let}{=} [\![fd^*]\!]_{fd^*}(G, [n_{start}])(fid_{new}) \\
& (N_2, L_1) \overset{let}{=} [\![stmt^*]\!]_{stmt*}(G, N_1, L)(fid_{new}) \\
& G.AddEdge(N_2, (fid_{new}, \mathsf{LExit})) \\
& G.AddEdge(L_1(\#return), (fid_{new}, \mathsf{LExit})) \\
& G.AddExcEdge(L_1(\#throw), (fid_{new}, \mathsf{LExitExc})) \\
& G.AddEdge(L_1(\#throw\_end), (fid_{new}, \mathsf{LExitExc})) \\
& G.AddEdge(L_1(\#after\_catch), (fid_{new}, \mathsf{LExitExc})) \\
& n_{tail} \overset{let}{=} GetTail(G, N)(fid) \\
& G.AddInst(n_{tail}, f := \texttt{function}\ (fid_{new})_{loc_1, loc_2}) \\
& [n_{tail}]
\end{aligned}
$$

$$\begin{aligned}
\llbracket \textit{stmt}^* \rrbracket_{\textit{stmt}*}(G,N,L)(fid) &= \text{if } (\textit{Length}(\textit{stmt}^*) = 0) \text{ then } (N,L)\\
&\quad \text{else } (N_1,L_1) \overset{\textit{let}}{=} \llbracket \textit{HeadOf}(\textit{stmt}^*) \rrbracket_{\textit{stmt}}(G,N,L)(fid)\\
&\qquad \llbracket \textit{TailOf}(\textit{stmt}^*) \rrbracket_{\textit{stmt}*}(G,N_1,L_1)(fid)
\end{aligned}$$

$$\llbracket \textsf{IRSeq}(\textit{stmt}^*) \rrbracket_{\textit{stmt}}(G,N,L)(fid) = \llbracket \textit{stmt}^* \rrbracket_{\textit{stmt}*}(G,N,L)(fid)$$

$$\llbracket \textsf{IRStmtUnit}(\textit{stmt}^*) \rrbracket_{\textit{stmt}}(G,N,L)(fid) = \llbracket \textit{stmt}^* \rrbracket_{\textit{stmt}*}(G,N,L)(fid)$$

$$\begin{aligned}
\llbracket \begin{smallmatrix} x=\textsf{function}\ f\,(\textit{this},\textit{args}) \\ \{s^*_{\textit{arg}},\textit{fd}^*,\textit{vd}^*,s^*_{\textit{body}}\} \end{smallmatrix} \rrbracket_{\textit{stmt}}(G,N,L)(fid) &= \textit{argVars} \overset{\textit{let}}{=} \llbracket s^*_{\textit{arg}} \rrbracket_{\textit{args}}\\
&\quad \textit{localVars} \overset{\textit{let}}{=} \llbracket \textit{fd}^* \rrbracket_{\textit{fdvars}} \mathbin{@} \llbracket \textit{vd}^* \rrbracket_{\textit{vds}} - \textit{argVars}\\
&\quad fid_{\textit{new}} \overset{\textit{let}}{=} G.\textit{NewFunction}(\textit{args}, \textit{argVars}, \textit{localVars})\\
&\quad n_{\textit{start}} \overset{\textit{let}}{=} G.\textit{NewBlock}(fid_{\textit{new}})\\
&\quad G.\textit{AddEdge}((fid_{\textit{new}}, \textsf{LEntry}), n_{\textit{start}})\\
&\quad L_{\textit{new}} \overset{\textit{let}}{=} [\,\#return \mapsto [\,], \#throw \mapsto [\,]\,, \#throw\_end \mapsto [\,]\,, \#after\_catch \mapsto [\,]\,]\\
&\quad N_1 \overset{\textit{let}}{=} \llbracket \textit{fd}^* \rrbracket_{\textit{fd}*}(G, [n_{\textit{start}}])(fid_{\textit{new}})\\
&\quad (N_2, L_1) \overset{\textit{let}}{=} \llbracket \textit{stmt}^* \rrbracket_{\textit{stmt}*}(G, N_1, L_{\textit{new}})(fid_{\textit{new}})\\
&\quad G.\textit{AddEdge}(N_2, (fid_{\textit{new}}, \textsf{LExit}))\\
&\quad G.\textit{AddEdge}(L_1(\#return), (fid_{\textit{new}}, \textsf{LExit}))\\
&\quad G.\textit{AddExcEdge}(L_1(\#throw), (fid_{\textit{new}}, \textsf{LExitExc}))\\
&\quad G.\textit{AddEdge}(L_1(\#throw\_end), (fid_{\textit{new}}, \textsf{LExitExc}))\\
&\quad G.\textit{AddEdge}(L_1(\#after\_catch), (fid_{\textit{new}}, \textsf{LExitExc}))\\
&\quad n_{\textit{tail}} \overset{\textit{let}}{=} \textit{GetTail}(G,N)(fid)\\
&\quad if\ (\underline{\textsf{getVarKind}}_P(f) = \textsf{CapturedVar})\\
&\qquad G.\textit{AddInst}(n_{\textit{tail}}, x := \texttt{function}\ f\ (fid_{\textit{new}})_{loc_1,loc_2,loc_3})\\
&\quad else\\
&\qquad G.\textit{AddInst}(n_{\textit{tail}}, x := \texttt{function}\ (fid_{\textit{new}})_{loc_1,loc_2})\\
&\quad ([n_{\textit{tail}}], L)
\end{aligned}$$

$$\begin{aligned}
\llbracket x = \{\textit{member}^*, \textit{proto}^?\} \rrbracket_{\textit{stmt}}(G,N,L)(fid) &= n_{\textit{tail}} \overset{\textit{let}}{=} \textit{GetTail}(G,N)(fid)\\
&\quad G.\textit{AddInst}(n_{\textit{tail}}, x := \texttt{alloc}(\textit{proto}^?)_{loc})\\
&\quad \textit{Iter}(\textit{memeber}^*)(\lambda(m) \Rightarrow \llbracket m \rrbracket_{\textit{mem}}(G, n_{tail})(x))\\
&\quad ([n_{tail}], L[\#throw \mapsto n_{tail} :: L(\#throw)])
\end{aligned}$$

$$\llbracket y : z \rrbracket_{\textit{mem}}(G,n)(x) = G.\textit{AddInst}(n, x[\text{“}y\text{”}] := z)$$

$$\begin{aligned}
\llbracket x = [\textit{elem}^*] \rrbracket_{\textit{stmt}}(G,N,L)(fid) &= n_{\textit{tail}} \overset{\textit{let}}{=} \textit{GetTail}(G,N)(fid)\\
&\quad G.\textit{AddInst}(n_{\textit{tail}}, x := \texttt{allocArray}(\textit{Length}(\textit{elem}^*))_{loc})\\
&\quad \_ \overset{\textit{let}}{=} \textit{Fold}(\textit{elem}^*)(0)(\lambda(e,k) \Rightarrow \llbracket y \rrbracket_{\textit{elem}}(G, n_{tail})(x,k))\\
&\quad ([n_{tail}], L[\#throw \mapsto n_{tail} :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = [\textit{elem}^*] \rrbracket_{\textit{stmt}}(G,N,L)(fid) &= n_{\textit{tail}} \overset{\textit{let}}{=} \textit{GetTail}(G,N)(fid)\\
(arguments) &\quad G.\textit{AddInst}(n_{\textit{tail}}, x := \texttt{allocArg}(\textit{Length}(\textit{elem}^*))_{loc})\\
&\quad \_ \overset{\textit{let}}{=} \textit{Fold}(\textit{elem}^*)(0)(\lambda(e,k) \Rightarrow \llbracket y \rrbracket_{\textit{elem}}(G, n_{tail})(x,k))\\
&\quad ([n_{tail}], L[\#throw \mapsto n_{tail} :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket y \rrbracket_{\textit{elem}}(G,n)(x,k) &= G.\textit{AddInst}(n, x[\text{“}k\text{”}] := y))\\
&\quad k+1
\end{aligned}$$

$$
\begin{array}{ll}
[\![x = f(\mathit{this}, \mathit{args}^{?})]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) = & n_1 \stackrel{\mathit{let}}{=} \mathit{GetTail}(G,N)(\mathit{fid}) \\
& \mathit{if}\ (f = \diamond \mathit{toObject}) \\
& \quad G.\mathit{AddInst}(n_1, x = \diamond f_1([\mathit{this},\mathit{args}])_{loc}) \\
& \quad ([n_1], L[\#\mathit{throw} \mapsto n_1 :: L(\#\mathit{throw})]) \\
& \mathit{else\ if}\ (f = \diamond \mathit{toBoolean} \lor f = \diamond \mathit{toNumber} \\
& \qquad\quad \lor\ f = \diamond \mathit{toString} \lor f = \diamond \mathit{isObject} \lor f = \diamond \mathit{iteratorInit} \\
& \qquad\quad \lor\ f = \diamond \mathit{iteratorHasNext} \lor f = \diamond \mathit{iteratorNext} \lor f = \diamond \mathit{getBase}) \\
& \quad G.\mathit{AddInst}(n_1, x = \diamond f_1([\mathit{this},\mathit{args}])) \\
& \quad ([n_1], L) \\
& \mathit{else} \\
& \quad G.\mathit{AddInst}(n_1, \mathtt{call}(f, \mathit{this}, \mathit{args})_{loc}) \\
& \quad n_2 \stackrel{\mathit{let}}{=} G.\mathit{NewAfterCallBlock}(\mathit{fid}, x) \\
& \quad n_3 \stackrel{\mathit{let}}{=} G.\mathit{NewAfterCatchBlock}(\mathit{fid}) \\
& \quad G.\mathit{AddCall}(n_1, n_2, n_3) \\
& \quad ([n_2], L[\#\mathit{throw} \mapsto n_1 :: L(\#\mathit{throw}), \#\mathit{after\_catch} \mapsto n_3 :: L(\#\mathit{after\_catch})])
\end{array}
$$

$$
\begin{array}{ll}
[\![x = c(\mathit{args})]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) & = n_1 \stackrel{\mathit{let}}{=} \mathit{GetTail}(G,N)(\mathit{fid}) \\
(\mathit{irnew}) & \quad G.\mathit{AddInst}(n_1, \mathtt{construct}(c, \mathit{args.hd}, \mathit{args.tl.hd})_{loc}) \\
& \quad n_2 \stackrel{\mathit{let}}{=} G.\mathit{NewAfterCallBlock}(\mathit{fid}, x) \\
& \quad n_3 \stackrel{\mathit{let}}{=} G.\mathit{NewAfterCatchBlock}(\mathit{fid}) \\
& \quad G.\mathit{AddCall}(n_1, n_2, n_3) \\
& \quad ([n_2], (L[\#\mathit{throw} \mapsto n_1 :: L(\#\mathit{throw}), \#\mathit{after\_catch} \mapsto n_3 :: L(\#\mathit{after\_catch})])
\end{array}
$$

$$
[\![\mathsf{with}(x)\ s]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) = ???
$$

$$
\begin{array}{ll}
[\![l : \{s\}]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) & = n \stackrel{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& \quad (N_1, L_1) \stackrel{\mathit{let}}{=} [\![s]\!]_{\mathit{stmt}}(G, N, L[l \mapsto [\,]])(\mathit{fid}) \\
& \quad G.\mathit{AddEdge}(N_1, n) \\
& \quad G.\mathit{AddEdge}(L_1(l), n) \\
& \quad L_2 \stackrel{\mathit{let}}{=} L_1 - l \\
& \quad ([n], L_2)
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathsf{if}(e)\ s_{\mathit{true}}\ \mathsf{else}\ s_{\mathit{false}}]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) = & n_1 \stackrel{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& n_2 \stackrel{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& G.\mathit{AddEdge}(N, n_1) \\
& G.\mathit{AddEdge}(N, n_2) \\
& G.\mathit{AddInst}(n_1, \mathtt{assert}(e)) \\
& G.\mathit{AddInst}(n_2, \mathtt{assert}(\neg e)) \\
& (N_1, L_1) \stackrel{\mathit{let}}{=} [\![s_{\mathit{true}}]\!]_{\mathit{stmt}}(G, [n_1], L)(\mathit{fid}) \\
& (N_2, L_2) \stackrel{\mathit{let}}{=} [\![s_{\mathit{false}}]\!]_{\mathit{stmt}}(G, [n_2], L_1)(\mathit{fid}) \\
& (N_1 @ N_2, L_2[\#\mathit{throw} \mapsto n_1 :: n_2 :: L_2(\#\mathit{throw})])
\end{array}
$$

$$
\begin{array}{ll}
[\![\mathsf{if}(e)\ s_{\mathit{true}}]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) & = n_1 \stackrel{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& \quad n_2 \stackrel{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& \quad G.\mathit{AddEdge}(N, n_1) \\
& \quad G.\mathit{AddEdge}(N, n_2) \\
& \quad G.\mathit{AddInst}(n_1, \mathtt{assert}(e)) \\
& \quad G.\mathit{AddInst}(n_2, \mathtt{assert}(\neg e)) \\
& \quad (N_1, L_1) \stackrel{\mathit{let}}{=} [\![s_{\mathit{true}}]\!]_{\mathit{stmt}}(G, [n_1], L)(\mathit{fid}) \\
& \quad (N_1 @ [n_2], L_1[\#\mathit{throw} \mapsto n_1 :: n_2 :: L_1(\#\mathit{throw})])
\end{array}
$$

$$\begin{aligned}
\llbracket \textsf{while}(e)\ s \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n_1 \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad n_{head} \overset{\mathit{let}}{=} G.\mathit{NewBlock}(fid)\\
&\quad n_2 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(fid)\\
&\quad n_3 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(fid)\\
&\quad G.\mathit{AddEdge}(n_1, n_{head})\\
&\quad G.\mathit{AddEdge}(n_{head}, n_2)\\
&\quad G.\mathit{AddEdge}(n_{head}, n_3)\\
&\quad G.AddInst(n_2, \texttt{assert}(e))\\
&\quad G.AddInst(n_3, \texttt{assert}(\neg e))\\
&\quad (N_1, L_1) \overset{\mathit{let}}{=} \llbracket s \rrbracket_{\mathit{stmt}}(G, [n_2], L)(fid)\\
&\quad G.\mathit{AddEdge}(N_1, n_{head})\\
&\quad ([n_3], L_1[\#throw \mapsto n_2 :: n_3 :: L_1(\#throw))
\end{aligned}$$

$$\begin{aligned}
\llbracket \textsf{throw}\ x \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, \texttt{throw}(x))\\
&\quad ([\,], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket \textsf{return}\ x \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, \texttt{return}(x))\\
&\quad ([\,], L[\#return \mapsto n :: L(\#return)])
\end{aligned}$$

$$\begin{aligned}
\llbracket \textsf{return}\ \underline{x}^{?} \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad ([\,], L[\#return \mapsto n :: L(\#return)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = e \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := e)\\
&\quad if\ (e = \texttt{IRId})\\
&\qquad ([n], L[\#throw \mapsto n :: L(\#throw)])\\
&\quad else\\
&\qquad ([n], L)
\end{aligned}$$

$$\begin{aligned}
\llbracket x = \textsf{delete}\ y \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := \texttt{delete}(y))\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = \textsf{delete}\ y[z] \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := \texttt{delete}(y, z))\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = \ominus\ y \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := \ominus y)\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = y \otimes z \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := y \otimes z)\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x[y] = z \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x[y] := z)\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$\begin{aligned}
\llbracket x = y[e] \rrbracket_{\mathit{stmt}}(G, N, L)(fid) &= n \overset{\mathit{let}}{=} \mathit{GetTail}(G, N)(fid)\\
&\quad G.\mathit{AddInst}(n, x := y[e])\\
&\quad ([n], L[\#throw \mapsto n :: L(\#throw)])
\end{aligned}$$

$$
\begin{array}{ll}
[\![\text{try}\{s\}\ \text{catch}(x)\ \{s_c\}]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) = & n_1 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& G.\mathit{AddEdge}(N, n_1) \\
& n_3 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& G.\mathit{AddInst}(n_3, \texttt{catch}(x)) \\
& L_{try} \overset{\mathit{let}}{=} [\ \#return \mapsto [\,], \#throw \mapsto [\,]\,, \#throw\_end \mapsto [\,]\,, \#after\_catch \mapsto [\,]\,] \\
& (N_1, L_1) = [\![s]\!]_{\mathit{stmt}}(G, [n_2], L_{try})(\mathit{fid}) \\
& G.\mathit{AddExcEdge}(L_1(\#throw), n_2) \\
& G.\mathit{AddEdge}(L_1(\#throw\_end), n_2) \\
& G.\mathit{AddEdge}(L_1(\#catch), n_2) \\
& (N_2, L_2) = [\![s_c]\!]_{\mathit{stmt}}(G, [n_2], L_1[\#throw \mapsto [\,]\,, \#throw\_end \mapsto [\,]\,, \\
& \qquad\qquad\qquad\qquad\qquad\qquad \#after\_catch \mapsto [\,]\,])(\mathit{fid}) \\
& L_3 \overset{\mathit{let}}{=} \mathit{Fold}(L_2)(L)(\lambda((l, N'), L') \Rightarrow \\
& \qquad if\ (L'.contains(l)) \\
& \qquad\qquad L'[l \mapsto L'(l)@N'] \\
& \qquad else \\
& \qquad\qquad L'[l \mapsto N'] \\
& (N_1@N_2, L_3)
\end{array}
$$

$$
\begin{array}{ll}
[\![\text{try}\{s\}\ \text{finally}\ \{s_f\}]\!]_{\mathit{stmt}}(G,N,L)(\mathit{fid}) = & n_1 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& G.\mathit{AddEdge}(N, n_1) \\
& n_2 \overset{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& L_{try} \overset{\mathit{let}}{=} [\ \#return \mapsto [\,], \#throw \mapsto [\,]\,, \#throw\_end \mapsto [\,]\,, \#after\_catch \mapsto [\,]\,] \\
& (N_1, L_1) \overset{\mathit{let}}{=} [\![s]\!]_{\mathit{stmt}}(G, [n_1], L_{try})(\mathit{fid}) \\
& (N_2, L_2) \overset{\mathit{let}}{=} [\![s_f]\!]_{\mathit{stmt}}(G, [n_2], L)(\mathit{fid}) \\
& G.\mathit{AddEdge}(N_1, n_2) \\
& L_3 \overset{\mathit{let}}{=} \mathit{Fold}(L_1[\ \#after\_catch \mapsto [\,]\,])(L_2)(\lambda((l, N'), L') \Rightarrow \\
& \qquad if\ (N' \neq Nil) \\
& \qquad\qquad n_{dup} \overset{\mathit{let}}{=} G.\mathit{NewBlock}(\mathit{fid}) \\
& \qquad\qquad (N'', L'') \overset{\mathit{let}}{=} [\![s_f]\!]_{\mathit{stmt}}(G, [n_{dup}], L')(\mathit{fid}) \\
& \qquad\qquad if\ (l = \#throw) \\
& \qquad\qquad\quad G.\mathit{AddEdge}(L_1(\#after\_catch), n_{dup}) \\
& \qquad\qquad\quad G.\mathit{AddExcEdge}(N', n_{dup});\ L''[\#throw\_end \mapsto L''(\#throw\_end)@N''] \\
& \qquad\qquad else \\
& \qquad\qquad\quad G.\mathit{AddEdge}(N', n_{dup});\ L''[l \mapsto L''(l)@N'']) \\
& (N_2, L_3)
\end{array}
$$

```
$[\![ \begin{smallmatrix}\mathsf{try}\{s\}\ \mathsf{catch}(x)\ \{s_c\}\\ \mathsf{finally}\ \{s_f\}\end{smallmatrix} ]\!]_{\mathit{stmt}}(G, N, L)(fid) = n_1 \overset{let}{=} G.\mathit{NewBlock}(fid)$
    $G.\mathit{AddEdge}(N, n_1)$
    $n_2 \overset{let}{=} G.\mathit{NewBlock}(fid)$
    $G.\mathit{AddInst}(n_2, \texttt{catch}(x))$
    $n_3 \overset{let}{=} G.\mathit{NewBlock}(fid)$
    $L_{try} \overset{let}{=} [\ \#return \mapsto [\,], \#throw \mapsto [\,]\ , \#throw\_end \mapsto [\,]\ , \#after\_catch \mapsto [\,]\,]$
    $(N_1, L_1) \overset{let}{=} [\![s]\!]_{\mathit{stmt}}(G, [n_1], L_{try})(fid)$
    $G.\mathit{AddExcEdge}(L_1(\#throw), n_2)$
    $G.\mathit{AddEdge}(L_1(\#throw\_end), n_2)$
    $G.\mathit{AddEdge}(L_1(\#after\_catch), n_2)$
    $(N_2, L_2) \overset{let}{=} [\![s_c]\!]_{\mathit{stmt}}(G, [n_2], L_1[\#throw \mapsto [\,]\ , \#throw\_end \mapsto [\,]\ ,$
                                $\#after\_catch \mapsto [\,]\,])(fid)$
    $(N_3, L_3) \overset{let}{=} [\![s_f]\!]_{\mathit{stmt}}(G, [n_3], L)(fid)$
    $G.\mathit{AddEdge}(N_1 @ N_2, n_3)$
    $L_4 \overset{let}{=} \mathit{Fold}(L_2[\ \#after\_catch \mapsto [\,]\,])(L_3)(\lambda((l, N'), L') \Rightarrow$
        $if\ (N' \neq Nil)$
            $n_{dup} \overset{let}{=} G.\mathit{NewBlock}(fid)$
            $(N'', L'') \overset{let}{=} [\![s_f]\!]_{\mathit{stmt}}(G, [n_{dup}], L')(fid)$
            $if\ (l = \#throw)$
                $G.\mathit{AddEdge}(L_2(\#after\_catch), n_{dup})$
                $G.\mathit{AddExcEdge}(N', n_{dup});\ L''[\#throw\_end \mapsto L''(\#throw\_end) @ N'']$
            $else$
                $G.\mathit{AddEdge}(N', n_{dup});\ L''[l \mapsto L''(l) @ N'']$
    $(N_3, L_4)$
```

Chapter 8

CFG Collecting Semantics

Assumptions and limitations are as follows:

- All the variables declared by '`var`' are included in the set $\mathsf{LocalVars}$.
- Followings are not yet supported: regular expression, with, getter, setter, eval.
- Runtime exception is omitted. `throw` is the only way to make an exception.
- Semantics of operators are omitted.
- Semantics for helper functions is not written using denotational semantics(they are not compositional).
- Try-catch clause in a finally block can disturb a flow of a previous throwed value.

8.1 Settings

$$
\begin{array}{rlcl}
x \in & \mathsf{Prop} & = & \mathsf{String} \cup \left\{ \begin{array}{l} @return,\ @exception,\ @exception\_all,\ @this,\ @up,\ @outer \\ @proto,\ @scope,\ @class,\ @function,\ @extensible,\ @construct \end{array} \right\} \\
l \in & \mathsf{Loc} & ::= & \#Global \mid \#ObjProto \mid \#ArrayProto \mid \#RefErrProto \\
 & & \mid & \#RangeErrProto \mid \#TypeErrProto \\
 & & \mid & l_1 \mid \cdots \\
cp \in & \mathsf{ControlPoint} & = & \mathsf{Node} \\
H \in & \mathsf{Heap} & = & \mathsf{Loc} \xrightarrow{\mathsf{fin}} \mathsf{Obj} \\
o \in & \mathsf{Obj} & = & \mathsf{Prop} \xrightarrow{\mathsf{fin}} \mathsf{PropValue} \\
A \in & \mathsf{Env} & = & \mathsf{Loc\ list} \\
(H, A), \mathsf{stuck} \in & \mathsf{State} & = & \mathsf{Heap} \times \mathsf{Env} \\
 & \mathsf{PropValue} & = & \mathsf{ObjectValue} \cup \mathsf{Value} \cup \mathsf{FunctionId} \cup \mathsf{Env} \\
 & & & \textit{`}\mathsf{Value} \cup \mathsf{FunctionId} \cup \mathsf{Env}\textit{' is for internal property.} \\
v \in & \mathsf{Value} & = & \mathsf{Loc} \cup \mathsf{PValue} \\
ov \in & \mathsf{ObjectValue} & = & \left\{ \begin{array}{l} value : \mathsf{Value}; \\ writable : \mathsf{Bool}; \\ enumerable : \mathsf{Bool}; \\ configurable : \mathsf{Bool}; \end{array} \right\} \\
pv \in & \mathsf{PValue} & = & \mathsf{Number} \cup \mathsf{String} \cup \mathsf{Bool} \cup \{\ \mathsf{undefined},\ \mathsf{null}\ \} \\
n \in & \mathsf{Number} & ::= & \mathsf{NaN} \mid \mathsf{Inf} \mid -\mathsf{Inf} \mid 0 \mid 1 \mid -1 \mid 2 \mid \cdots \\
s \in & \mathsf{String} & & \\
b \in & \mathsf{Bool} & & \\
exc \in & \mathsf{Exception} & ::= & \mathsf{ReferenceError} \mid \mathsf{RangeError} \mid \mathsf{TypeError}
\end{array}
$$

8.2 Helper Functions

$$\underline{\mathsf{PushStack}}(l_o^*, l_n) = l_n :: l_o^*$$
$$\underline{\mathsf{TopStack}}(l_n :: l_o^*) = l_n$$

$$\underline{\mathsf{Dom}}(H) = \{\ l \mid l \mapsto o \in H\ \}$$
$$\underline{\mathsf{Dom}}(o) = \{\ x \mid x \mapsto v \in o\ \}$$

$$\underline{\mathsf{IsArray}}(H, l) = \begin{cases} \mathsf{true} & \text{if } H(l)(@class) = \text{“}Array\text{”} \\ \mathsf{false} & \text{otherwise} \end{cases}$$
$$\underline{\mathsf{IsObject}}(H, l) = \begin{cases} \mathsf{true} & \text{if } H(l)(@class) = \text{“}Object\text{”} \lor H(l)(@class) = \text{“}Function\text{”} \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$\underline{\mathsf{IsArrayIndex}}$ $: \mathsf{Value} \to \mathsf{Bool}$

$$\underline{\mathsf{IsArrayIndex}}(v) = \begin{cases} \mathsf{true} & \text{if } \underline{\mathsf{toString}}(\underline{\mathsf{ToUint32}}(\underline{\mathsf{toString}}(v))) = \underline{\mathsf{toString}}(v) \\ & \quad \land \underline{\mathsf{ToUint32}}(\underline{\mathsf{toString}}(v)) \neq 2^{32} - 1 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$\underline{\mathsf{VarStore}}$ $: \mathsf{Heap} \times \mathsf{Env} \times \mathsf{Prop} \times \mathsf{Value} \times \mathsf{Bool} \to \mathsf{Heap}$

$$\underline{\mathsf{VarStore}}(H, [\#Global], x, v, b) = \underline{\mathsf{PropStore}}(H, \#Global, x, v)$$
$$\quad \text{if } x \notin \underline{\mathsf{Dom}}(H(\#Global))$$
$$\underline{\mathsf{VarStore}}(H, l_{hd} :: l_{tl}^*, x, v, b) = H[l_{hd} \mapsto H(l_{hd})[x \mapsto \{\ H(l_{hd})(x) \text{ with } value = v; writable = b\ \}]]$$
$$\quad \text{if } x \in \underline{\mathsf{Dom}}(H(l_{hd}))$$
$$\underline{\mathsf{VarStore}}(H, l_{hd} :: l_{tl}^*, x, v, b) = \underline{\mathsf{VarStore}}(H, l_{tl}^*, x, v, b)$$
$$\quad \text{if } x \notin \underline{\mathsf{Dom}}(H(l_{hd}))$$

$\underline{\mathsf{VarStoreE}}$ $: \mathsf{Heap} \times \mathsf{Env} \times \mathsf{Prop} \times \mathsf{Value} \times \mathsf{Bool} \to \mathsf{Heap}$

$$\underline{\mathsf{VarStoreE}}(H, A, t, v, b) = \underline{\mathsf{VarStore}}(H, A, t, v, b)$$
$$\underline{\mathsf{VarStoreE}}(H, A, x, v, b) = \begin{cases} \underline{\mathsf{VarStore}}(H, A, x, v, b) & \text{if } \underline{\mathsf{CanPutVar}}(H, A, x) \\ H & \text{otherwise} \end{cases}$$

$\underline{\mathsf{PropStore}}$ $: \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \times \mathsf{Value} \to \mathsf{Heap}$

$$\underline{\mathsf{PropStore}}(H, l, x, v) = H\left[l \mapsto H(l) \left[x \mapsto \left\{ \begin{array}{l} value = v; \\ enumerable = \mathsf{true}; \\ configurable = \mathsf{true}; \\ writable = \mathsf{true} \end{array} \right\} \right] \right]$$
$$\quad \text{if } x \notin \underline{\mathsf{Dom}}(H(l))$$
$$\underline{\mathsf{PropStore}}(H, l, x, v) = H\ [l \mapsto H(l)\ [x \mapsto \{\ H(l)(x) \text{ with } value = v\ \}]]$$
$$\quad \text{if } x \in \underline{\mathsf{Dom}}(H(l))$$

$\underline{\mathsf{Delete}}$ $: \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \to \mathsf{Heap} \times \mathsf{Bool}$

$$\underline{\mathsf{Delete}}(H, l, x) = (H, \mathsf{true})$$
$$\quad \text{if } \neg\underline{\mathsf{HasOwnProperty}}(H, l, x)$$
$$\underline{\mathsf{Delete}}(H, l, x) = (H, \mathsf{false})$$
$$\quad \text{if } \underline{\mathsf{HasOwnProperty}}(H, l, x) \land H(l)(x).configurable = \mathsf{false}$$
$$\underline{\mathsf{Delete}}(H, l, x) = (H[l \mapsto H(l) - x], \mathsf{true})$$
$$\quad \text{if } \underline{\mathsf{HasOwnProperty}}(H, l, x) \land H(l)(x).configurable = \mathsf{true}$$

$\underline{\mathsf{Lookup}}$ $: \mathsf{Heap} \times \mathsf{Env} \times \mathsf{Prop} \to \mathsf{Value} \cup \mathsf{Exception}$

$$\begin{array}{ll} \underline{\mathsf{Lookup}}(H, [\#Global], x) = \mathsf{ReferenceError} & \text{if } \neg\underline{\mathsf{HasProperty}}(H, \#Global, x) \\ \underline{\mathsf{Lookup}}(H, [\#Global], x) = \underline{\mathsf{Proto}}(H, \#Global, x) & \text{if } \underline{\mathsf{HasProperty}}(H, \#Global, x) \\ \underline{\mathsf{Lookup}}(H, l_{hd} :: l_{tl}^*, x) = H(l_{hd})(x).value & \text{if } x \in \underline{\mathsf{Dom}}(H(l_{hd})) \\ \underline{\mathsf{Lookup}}(H, l_{hd} :: l_{tl}^*, x) = \underline{\mathsf{Lookup}}(H, l_{tl}^*, x) & \text{if } x \notin \underline{\mathsf{Dom}}(H(l_{hd})) \end{array}$$

$$\begin{array}{ll}
\underline{\mathsf{TypeTag}} & : \mathsf{Heap} \times \mathsf{Value} \rightarrow \left\{ \begin{array}{l} \text{“}number\text{”}, \text{“}string\text{”}, \text{“}boolean\text{”}, \text{“}object\text{”}, \\ \text{“}function\text{”}, \text{“}null\text{”}, \text{“}undefined\text{”} \end{array} \right\} \\
& \underline{\mathsf{TypeTag}}(H, v) = \begin{cases} \text{“}number\text{”} & \text{if } v \in \mathsf{Number} \\ \text{“}boolean\text{”} & \text{if } v \in \mathsf{Boolean} \\ \text{“}string\text{”} & \text{if } v \in \mathsf{String} \\ \text{“}object\text{”} & \text{if } v \in \mathsf{Loc} \wedge \neg \underline{\mathsf{IsCallable}}(H, v) \\ \text{“}function\text{”} & \text{if } v \in \mathsf{Loc} \wedge \underline{\mathsf{IsCallable}}(H, v) \\ \text{“}object\text{”} & \text{if } v = \mathsf{null} \\ \text{“}undefined\text{”} & \text{if } v = \mathsf{undefined} \end{cases}
\end{array}$$

$$\begin{array}{ll}
\underline{\mathsf{CanPut}} & : \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \rightarrow \mathsf{Bool} \\
& \quad \underline{\mathsf{CanPut}}(H, l, x) = \underline{\mathsf{CanPutHelp}}(H, l, x, l)
\end{array}$$

$$\begin{array}{ll}
\underline{\mathsf{CanPutHelp}} & : \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \times \mathsf{Loc} \rightarrow \mathsf{Bool} \\
& \quad \underline{\mathsf{CanPutHelp}}(H, l_1, x, l_2) = \underline{\mathsf{CanPutHelp}}(H, H(l_1)(@proto).value, x, l_2) \\
& \qquad \text{if } x \notin \underline{\mathsf{Dom}}(H(l_1)) \wedge H(l_1)(@proto).value \neq \mathsf{null} \\
& \quad \underline{\mathsf{CanPutHelp}}(H, l_1, x, l_2) = H(l)(@extensible) \\
& \qquad \text{if } x \notin \underline{\mathsf{Dom}}(H(l_1)) \wedge H(l_1)(@proto).value = \mathsf{null} \\
& \quad \underline{\mathsf{CanPutHelp}}(H, l_1, x, l_2) = H(l_1)(x).writable \\
& \qquad \text{if } x \in \underline{\mathsf{Dom}}(H(l_1))
\end{array}$$

$$\begin{array}{lll}
\underline{\mathsf{CanPutVar}} & : \mathsf{Heap} \times \mathsf{Env} \times \mathsf{Prop} \rightarrow \mathsf{Bool} & \\
& \quad \underline{\mathsf{CanPutVar}}(H, [\#Global], x) = \underline{\mathsf{CanPut}}(H, \#Global, x) & \\
& \quad \underline{\mathsf{CanPutVar}}(H, l_{hd} :: l^*_{tl}, x) = H(l_{hd})(x).writable & \text{if } x \in \underline{\mathsf{Dom}}(H(l_{hd})) \\
& \quad \underline{\mathsf{CanPutVar}}(H, l_{hd} :: l^*_{tl}, x) = \underline{\mathsf{CanPutVar}}(H, l^*_{tl}, x) & \text{if } x \notin \underline{\mathsf{Dom}}(H(l_{hd}))
\end{array}$$

$$\begin{array}{ll}
\underline{\mathsf{HasProperty}} & : \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \rightarrow \mathsf{Bool} \\
& \underline{\mathsf{HasProperty}}(H, l, x) = \mathsf{true} \\
& \quad \text{if } \underline{\mathsf{HasOwnProperty}}(H, l, x) \\
& \underline{\mathsf{HasProperty}}(H, l, x) = \mathsf{false} \\
& \quad \text{if } \neg \underline{\mathsf{HasOwnProperty}}(H, l, x) \wedge H(l_1)(@proto).value = \mathsf{null} \\
& \underline{\mathsf{HasProperty}}(H, l, x) = \underline{\mathsf{HasProperty}}(H, H(l)(@proto).value, x) \\
& \quad \text{if } \neg \underline{\mathsf{HasOwnProperty}}(H, l, x) \wedge H(l_1)(@proto).value \neq \mathsf{null}
\end{array}$$

$$\begin{array}{lll}
\underline{\mathsf{HasOwnProperty}} & : \mathsf{Heap} \times \mathsf{Loc} \times \mathsf{Prop} \rightarrow \mathsf{Bool} & \\
& \quad \underline{\mathsf{HasOwnProperty}}(H, l, x) = \mathsf{false} & \text{if } x \notin \underline{\mathsf{Dom}}(H(l)) \\
& \quad \underline{\mathsf{HasOwnProperty}}(H, l, x) = \mathsf{true} & \text{if } x \in \underline{\mathsf{Dom}}(H(l))
\end{array}$$

$$\begin{array}{lll}
\underline{\mathsf{LookupBase}} & : \mathsf{Heap} \times \mathsf{Env} \times \mathsf{Prop} \rightarrow \mathsf{Loc} & \\
& \quad \underline{\mathsf{LookupBase}}(H, [\#Global], x) = \underline{\mathsf{ProtoBase}}(H, \#Global, x) & \\
& \quad \underline{\mathsf{LookupBase}}(H, l_{hd} :: l^*_{tl}, x) = l_{hd} & \text{if } x \in \underline{\mathsf{Dom}}(H(l_{hd})) \\
& \quad \underline{\mathsf{LookupBase}}(H, l_{hd} :: l^*_{tl}, x) = \underline{\mathsf{LookupBase}}(H, l^*_{tl}, x) & \text{if } x \notin \underline{\mathsf{Dom}}(H(l_{hd}))
\end{array}$$

$\underline{\mathsf{ProtoBase}}$: Heap × Loc × Prop → Loc

$$\begin{array}{l}
\underline{\mathsf{ProtoBase}}(H, l, x) = l \\
\quad \text{if } \ x \in \underline{\mathsf{Dom}}(H(l)) \\
\underline{\mathsf{ProtoBase}}(H, l, x) = \{\} \\
\quad \text{if } \ x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value \ = \mathsf{null} \\
\underline{\mathsf{ProtoBase}}(H, l, x) = \underline{\mathsf{ProtoBase}}(H, H(l)(@proto).value, x) \\
\quad \text{if } \ x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value \ \neq \mathsf{null}
\end{array}$$

$\underline{\mathsf{Proto}}$: Heap × Loc × Prop → Value

$$\begin{array}{l}
\underline{\mathsf{Proto}}(H, l, x) = H(l)(x).value \\
\quad \text{if } \ x \in \underline{\mathsf{Dom}}(H(l)) \\
\underline{\mathsf{Proto}}(H, l, x) = \underline{\mathsf{Proto}}(H, H(l)(@proto).value, x) \\
\quad \text{if } \ x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value \ \neq \mathsf{null} \\
\underline{\mathsf{Proto}}(H, l, x) = \mathsf{undefined} \\
\quad \text{if } \ x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value \ = \mathsf{null}
\end{array}$$

$\underline{\mathsf{NewObject}}$: Loc → Obj

$$\underline{\mathsf{NewObject}}(l) = \left\{ \begin{array}{l}
@class \mapsto {''}Object{''}, \\
@proto \mapsto \left\{ \begin{array}{l} value = l; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\
@extensible \mapsto \mathsf{true}
\end{array} \right\}$$

$\underline{\mathsf{NewFunctionObject}}$: FunctionId × Env × Loc × Number → Obj

$$\underline{\mathsf{NewFunctionObject}}(fid, A, l, n) = \left\{ \begin{array}{l}
@class \mapsto ``Function", \\
@function \mapsto fid, \\
@construct \mapsto fid, \\
@scope \mapsto A, \\
@proto \mapsto \left\{ \begin{array}{l} value = \#FunctionProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\
``prototype" \mapsto \left\{ \begin{array}{l} value : l; \\ writable : \mathsf{true}; \\ enumerable : \mathsf{false}; \\ configurable : \mathsf{false} \end{array} \right\}, \\
``length" \mapsto \left\{ \begin{array}{l} value : n; \\ writable : \mathsf{false}; \\ enumerable : \mathsf{false}; \\ configurable : \mathsf{false} \end{array} \right\}, \\
@extensible \mapsto \mathsf{true}
\end{array} \right\}$$

$\underline{\mathsf{NewArrayObject}}$: Number → Obj

$$\underline{\mathsf{NewArrayObject}}(n) = \left\{ \begin{array}{l}
@class \mapsto ``Array", \\
@proto \mapsto \left\{ \begin{array}{l} value = \#ArrayProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\
``length" \mapsto \left\{ \begin{array}{l} value : n; \\ writable : \mathsf{true}; \\ enumerable : \mathsf{false}; \\ configurable : \mathsf{false} \end{array} \right\}, \\
@extensible \mapsto \mathsf{true}
\end{array} \right\}$$

$\underline{\mathsf{NewArgObject}}$: Number → Obj

$$\underline{\mathsf{NewArgObject}}(n) = \left\{ \begin{array}{l} @class \mapsto \text{“}Arguments\text{”}, \\ @proto \mapsto \left\{ \begin{array}{l} value = \#ObjProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ \text{“}length\text{”} \mapsto \left\{ \begin{array}{l} value : n; \\ writable : \mathsf{true}; \\ enumerable : \mathsf{false}; \\ configurable : \mathsf{true} \end{array} \right\}, \\ @extensible \mapsto \mathsf{true} \end{array} \right\}$$

$\underline{\mathsf{NewBoolean}}$: Bool → Obj

$$\underline{\mathsf{NewBoolean}}(v) = \left\{ \begin{array}{l} @class \mapsto \text{“}Boolean\text{”}, \\ @proto \mapsto \left\{ \begin{array}{l} value = \#BoolProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @extensible \mapsto \mathsf{true}, \\ @primitive \mapsto v \end{array} \right\}$$

$\underline{\mathsf{NewNumber}}$: Number → Obj

$$\underline{\mathsf{NewNumber}}(v) = \left\{ \begin{array}{l} @class \mapsto \text{“}Number\text{”}, \\ @proto \mapsto \left\{ \begin{array}{l} value = \#NumProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @extensible \mapsto \mathsf{true}, \\ @primitive \mapsto v \end{array} \right\}$$

$\underline{\mathsf{NewString}}$: String → Obj

$$\begin{array}{l} \underline{\mathsf{NewString}}(s) = o_1 \cup o_2 \\ \quad \text{where } v_{len} = length(s) \\ \quad \wedge\ o_1 = \left\{ \begin{array}{l} @class \mapsto \text{“}String\text{”}, \\ @proto \mapsto \left\{ \begin{array}{l} value = \#StrProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @extensible \mapsto \mathsf{true}, \\ @primitive \mapsto s, \\ \text{“}length\text{”} \mapsto \left\{ \begin{array}{l} value = v_{len}; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\} \end{array} \right\} \\ \quad \wedge\ o_2 = \left\{ \text{“}i\text{”} \mapsto \left\{ \begin{array}{l} value = v_{char}; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{true}; \\ configurable = \mathsf{false} \end{array} \right\} \;\middle|\; 0 \leq i < v_{len} \wedge\ v_{char} = charAt(s,i) \right\} \end{array}$$

$\underline{\mathsf{IsCallable}}$: Heap × Loc → Bool

$$\underline{\mathsf{IsCallable}}(H,l) = \begin{cases} \mathsf{true} & \text{if } @function \in \underline{\mathsf{Dom}}(H(l)) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$\underline{\mathsf{HasConstruct}}$

$$: \mathsf{Heap} \times \mathsf{Loc} \to \mathsf{Bool}$$

$$\underline{\mathsf{HasConstruct}}(H, l) = \begin{cases} \mathsf{true} & \text{if } @construct \in \underline{\mathsf{Dom}}(H(l)) \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$\underline{\mathsf{newLocation}}$

$$: \mathsf{Unit} \to \mathsf{Loc}$$

$$\underline{\mathsf{newLocation}}() = l_{new}$$

$\underline{\mathsf{toNumber}}$

$$: \mathsf{PValue} \to \mathsf{Number}$$

$$\underline{\mathsf{toNumber}}(pv) = \begin{cases} \mathsf{NaN} & \text{if } pv = \mathsf{undefined} \\ 0 & \text{if } pv = \mathsf{null} \vee pv = \mathsf{false} \\ 1 & \text{if } pv = \mathsf{true} \\ pv & \text{if } pv \in \mathsf{Number} \\ \underline{\mathsf{Str2Num}(pv)} & \text{if } pv \in \mathsf{String} \end{cases}$$

$\underline{\mathsf{toString}}$

$$: \mathsf{PValue} \to \mathsf{String}$$

$$\underline{\mathsf{toString}}(pv) = \begin{cases} \text{“}undefined\text{”} & \text{if } pv = \mathsf{undefined} \\ \text{“}null\text{”} & \text{if } pv = \mathsf{null} \\ \text{“}pv\text{”} & \text{if } pv \in \mathsf{Boolean} \\ \text{“}pv\text{”} & \text{if } pv \in \mathsf{Number} \\ pv & \text{if } pv \in \mathsf{String} \end{cases}$$

$\underline{\mathsf{toBoolean}}$

$$: \mathsf{Value} \to \mathsf{Bool}$$

$$\underline{\mathsf{toBoolean}}(v) = \begin{cases} \mathsf{false} & \text{if } v = \mathsf{undefined} \\ \mathsf{false} & \text{if } v = \mathsf{null} \\ v & \text{if } v \in \mathsf{Boolean} \\ \mathsf{false} & \text{if } v \in \mathsf{Number} \wedge v \in \{\ 0, \mathsf{NaN}\ \} \\ \mathsf{true} & \text{if } v \in \mathsf{Number} \wedge v \notin \{\ 0, \mathsf{NaN}\ \} \\ \mathsf{false} & \text{if } v \in \mathsf{String} \wedge v = \text{“”} \\ \mathsf{true} & \text{if } v \in \mathsf{String} \wedge v \neq \text{“”} \\ \mathsf{true} & \text{if } v \in \mathsf{Loc} \end{cases}$$

$\underline{\mathsf{toPrimitive}}$

$$: \mathsf{Value} \to \mathsf{PValue}$$

$$\underline{\mathsf{toPrimitive}}(v) = \begin{cases} v & \text{if } v \notin \mathsf{Loc} \\ \underline{\mathsf{Obj2Str}(v)} & \text{if } v \in \mathsf{Loc} \end{cases}$$

$\underline{\mathsf{toObject}}$

$$: \mathsf{Heap} \times \mathsf{Value} \to \mathsf{Heap} \times \mathsf{Value} \cup \mathsf{Exception}$$

$$\underline{\mathsf{toObject}}(H, l) = (H, l)$$

$$\underline{\mathsf{toObject}}(H, v) = (H, \mathsf{TypeError}) \quad \text{if } v \in \{\ \mathsf{undefined}, \mathsf{null}\ \}$$

$$\underline{\mathsf{toObject}}(H, v) = (H_1, l_{new})$$

$$\text{where } o = \begin{cases} \underline{\mathsf{NewString}}(v) & \text{if } v \in \mathsf{String} \\ \underline{\mathsf{NewNumber}}(v) & \text{if } v \in \mathsf{Number} \\ \underline{\mathsf{NewBoolean}}(v) & \text{if } v \in \mathsf{Bool} \end{cases}$$

$$H_1 = H[l_{new} \mapsto o]$$

$$l_{new} = \underline{\mathsf{newLocation}}()$$

$\underline{\mathsf{getThis}}$: Value → Loc

$$\underline{\mathsf{getThis}}(v) = \begin{cases} \#Global & \text{if } v = \mathsf{undefined} \\ \#Global & \text{if } v = \mathsf{null} \\ \#Global & \text{if } v \in \mathsf{Loc} \wedge \neg\underline{\mathsf{IsObject}}(v) \\ v & \text{if } v \in \mathsf{Loc} \wedge \underline{\mathsf{IsObject}}(v) \end{cases}$$

$\underline{\mathsf{inherit}}$: Heap × Loc × Loc → Value

$$\underline{\mathsf{inherit}}(H, l_1, l_2) = \begin{cases} \mathsf{true} & \text{if } l_1 = l_2 \\ \mathsf{false} & \text{if } l_1 \neq l_2 \wedge H(l_1)(@proto).value\ = \mathsf{null} \\ \underline{\mathsf{inherit}}(H, H(l_1)(@proto).value, l_2) & \text{if } l_1 \neq l_2 \wedge H(l_1)(@proto).value\ \neq \mathsf{null} \end{cases}$$

$\underline{\mathsf{iteratorInit}}$: Obj × $\wp$(Prop) × Number → Obj

$$\underline{\mathsf{iteratorInit}}(o, P, n) = \begin{cases} \{@i \mapsto 0\} \\ \quad \text{if } P = \emptyset \\ \underline{\mathsf{iteratorInit}}(o, P - x, n + 1)[n \mapsto x] \\ \quad \text{if } x \in P \end{cases}$$

$\underline{\mathsf{collectProps}}$: Heap × Loc → $\wp$(Loc)

$$\underline{\mathsf{collectProps}}(H, l) = \begin{cases} \underline{\mathsf{Dom}}(H(l)) \cup\ \underline{\mathsf{CollectProps}}(H, H(l)(@proto).value) & \text{if } H(l)(@proto).value \neq \mathsf{null} \\ \{\} & \text{if } H(l)(@proto).value = \mathsf{null} \end{cases}$$

$\underline{\mathsf{isEnumerable}}$: Heap × Loc × Prop → Bool

$$\underline{\mathsf{isEnumerable}}(H, l, x) = \begin{cases} H(l)(x).enumerable \\ \quad \text{if } x \in \underline{\mathsf{Dom}}(H(l)) \\ \underline{\mathsf{isEnumerable}}(H, H(l)(@proto).value, x) \\ \quad \text{if } x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value \neq \mathsf{null} \\ \mathsf{false} \\ \quad \text{if } x \notin \underline{\mathsf{Dom}}(H(l)) \wedge H(l)(@proto).value = \mathsf{null} \end{cases}$$

$\underline{\mathsf{next}}$: Heap × Obj × Number × Loc → Number

$$\underline{\mathsf{next}}(H, o_{iter}, n, l) = \begin{cases} n & \text{if } n \notin \underline{\mathsf{Dom}}(o_{iter}) \\ n & \text{if } n \in \underline{\mathsf{Dom}}(o_{iter}) \wedge\ \underline{\mathsf{isEnumerable}}(H, l, o_{iter}(n)) \\ \underline{\mathsf{next}}(H, o_{iter}, n + 1, l) & \text{otherwise} \end{cases}$$

$\underline{\mathsf{NewExceptionObject}}$: Exception → Obj

$$\underline{\mathsf{NewExceptionObject}}(exc) = \underline{\mathsf{NewObject}}(l)$$
$$\quad \text{where } \ l = \begin{cases} \#RefErrProto & \text{if } exc = \mathsf{ReferenceError} \\ \#RangeErrProto & \text{if } exc = \mathsf{RangeError} \\ \#TypeErrProto & \text{if } exc = \mathsf{TypeError} \end{cases}$$

$\underline{\mathsf{RaiseException}}$: Heap × Exception ∪ Value → Heap

$$\begin{aligned} &\underline{\mathsf{RaiseException}}(H, v) = H_1 \\ &\quad \text{where } \ H_1 = H[\#temp \mapsto H(\#temp)[@exception \mapsto v]] \\ &\underline{\mathsf{RaiseException}}(H, exc) = H_2 \\ &\quad \text{where } \ l_e = \underline{\mathsf{newLocation}}() \\ &\qquad\qquad H_1 = H[l_e \mapsto \underline{\mathsf{NewExceptionObject}}(exc)] \\ &\qquad\qquad H_2 = H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]] \end{aligned}$$

8.3 Semantics

$$\begin{array}{lcl}
\mathcal{C} & \in & \mathsf{ControlPoint} \to \mathsf{Command} \to \wp(\mathsf{State}) \to \wp(\mathsf{State}) \\
\mathcal{I} & \in & \mathsf{ControlPoint} \to \mathsf{Instruction} \to \mathsf{State} \to \mathsf{State} \\
\mathcal{V} & \in & \mathsf{Expression} \to \mathsf{State} \to \mathsf{Value} \cup \mathsf{Exception} \\
\mathcal{B} & \in & \mathsf{Expression} \to \mathsf{State} \to \mathsf{State}
\end{array}$$

$$\begin{array}{l}
\mathcal{C}_{cp}\llbracket \mathsf{entry} \rrbracket S = \bigcup \{\ (H_1, A) \mid (H, A) \in S\ \} \\
\quad \text{where } (fid_{this}, \mathsf{ENTRY}) = cp \wedge l = \underline{\mathsf{TopStack}}(A) \\
\qquad\quad \wedge\ x^*_{argvar} = \underline{\mathsf{getArgVars}}_P(fid_{this}) \ \wedge\ x^*_{localvar} = \underline{\mathsf{getLocalVars}}_P(fid_{this}) \\
\qquad\quad \wedge\ H_1 = H\left[l \mapsto H(l) \left[\begin{array}{l} \left(x_{argvar} \mapsto \left\{ \begin{array}{l} value = \mathsf{undefined}; writable = \mathsf{true}; \\ enumerable = \mathsf{false}; configurable = \mathsf{false} \end{array} \right\}\right)^*, \\ \left(x_{localvar} \mapsto \left\{ \begin{array}{l} value = \mathsf{undefined}; writable = \mathsf{true}; \\ enumerable = \mathsf{true}; configurable = \mathsf{false} \end{array} \right\}\right)^* \end{array} \right]\right]
\end{array}$$

$$\mathcal{C}_{cp}\llbracket \mathsf{exit} \rrbracket S = S$$

$$\mathcal{C}_{cp}\llbracket \mathsf{exit\text{-}exc} \rrbracket S = S$$

$$\mathcal{C}_{cp}\llbracket i^+ \rrbracket S = \bigcup \{\ (\mathcal{I}_{cp}\llbracket i \rrbracket(H, A))^+ \mid (H, A) \in S\ \}$$

$$\mathcal{I}_{cp}\llbracket i \rrbracket(H, A) = (H, A) \quad \text{if } \ \underline{\mathsf{HasProperty}}(H, \#temp, @exception) \vee (H, A) = \mathsf{stuck}$$

** if e is None, v is considered like a value which is not an element of* Loc*.*

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\texttt{:=alloc(}e^?\texttt{)} \rrbracket(H, A) = (H_2, A) \quad \text{if } \ v = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } l_{new} = \underline{\mathsf{newLocation}}() \\
\qquad\qquad l_p = \left\{ \begin{array}{ll} v & \text{if } \ v \in \mathsf{Loc} \\ \#ObjProto & \text{otherwise} \end{array} \right. \\
\qquad\qquad H_1 = H[l_{new} \mapsto \underline{\mathsf{NewObject}(l_p)}] \\
\qquad\qquad H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x, l_{new}, \mathsf{true}) \\
\mathcal{I}_{cp}\llbracket x\texttt{:=alloc(}e^?\texttt{)} \rrbracket(H, A) = (H_1, A) \quad \text{if } \ exc = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\texttt{:=allocArray(n)} \rrbracket(H, A) = (H_2, A) \\
\quad \text{where } l_{new} = \underline{\mathsf{newLocation}}() \\
\qquad\qquad n = \mathcal{V}\llbracket \mathrm{n} \rrbracket(H, A) \\
\qquad\qquad H_1 = H[l_{new} \mapsto \underline{\mathsf{NewArrayObject}(n)}] \\
\qquad\qquad H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x, l_{new}, \mathsf{true})
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\texttt{:=allocArg(n)} \rrbracket(H, A) = (H_2, A) \\
\quad \text{where } l_{new} = \underline{\mathsf{newLocation}}() \\
\qquad\qquad n = \mathcal{V}\llbracket \mathrm{n} \rrbracket(H, A) \\
\qquad\qquad H_1 = H[l_{new} \mapsto \underline{\mathsf{NewArgObject}(n)}] \\
\qquad\qquad H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x, l_{new}, \mathsf{true})
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\texttt{:=}e \rrbracket(H, A) = (H_1, A) \quad \text{if } \ v = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } H_1 = \underline{\mathsf{VarStoreE}}(H, A, x, v, \mathsf{true}) \\
\mathcal{I}_{cp}\llbracket x\texttt{:=}e \rrbracket(H, A) = (H_1, A) \quad \text{if } \ exc = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x_1\texttt{:=delete(}x_2\texttt{)} \rrbracket(H, A) = (H_2, A) \\
\quad \text{where } l_{base} = \underline{\mathsf{LookupBase}}(H, A, x_2) \\
\qquad\qquad (H_1, b) = \underline{\mathsf{Delete}}(H, l_{base}, x_2) \\
\qquad\qquad H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x_1, b, \mathsf{true}) \\
\mathcal{I}_{cp}\llbracket x\texttt{:=delete(}e\texttt{)} \rrbracket(H, A) = (H_1, A) \quad \text{if } \ v = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } H_1 = \underline{\mathsf{VarStoreE}}(H, A, x, \mathsf{true}, \mathsf{true}) \\
\mathcal{I}_{cp}\llbracket x\texttt{:=delete(}e\texttt{)} \rrbracket(H, A) = (H_1, A) \quad \text{if } \ exc = \mathcal{V}\llbracket e \rrbracket(H, A) \\
\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\texttt{:=delete(}e_1\texttt{,}e_2\texttt{)} \rrbracket(H, A) = (H_2, A) \\
\quad \text{where } l = \mathcal{V}\llbracket e_1 \rrbracket(H, A) \ \wedge\ s = \mathcal{V}\llbracket e_2 \rrbracket(H, A) \\
\qquad\qquad (H_1, b) = \underline{\mathsf{Delete}}(H, l, s) \\
\qquad\qquad H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x, b, \mathsf{true})
\end{array}$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,e_2\,\texttt{]}\,\texttt{=}e_3\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; exc = \mathcal{V}\llbracket e_3\rrbracket(H,A) \\
\quad \text{where} \;\; H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,e_2\,\texttt{]}\,\texttt{=}e_3\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; \neg\mathsf{IsArray}(H,l) \;\wedge\; \underline{\mathsf{CanPut}}(H,l,x) \\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \\
\qquad\qquad x = \mathcal{V}\llbracket e_2\rrbracket(H,A) \\
\qquad\qquad v = \mathcal{V}\llbracket e_3\rrbracket(H,A) \\
\qquad\qquad H_1 = \underline{\mathsf{PropStore}}(H,l,x,v)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,e_2\,\texttt{]}\,\texttt{=}e_3\rrbracket(H,A) = (H_2,A) \quad \text{if} \;\; \mathsf{IsArray}(H,l) \;\wedge\; \underline{\mathsf{CanPut}}(H,l,v_{idx}) \;\wedge\; \underline{\mathsf{IsArrayIndex}}(v_{idx}) \\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \\
\qquad\qquad v_{idx} = \mathcal{V}\llbracket e_2\rrbracket(H,A) \\
\qquad\qquad v = \mathcal{V}\llbracket e_3\rrbracket(H,A) \\
\qquad\qquad n_{oldLen} = \underline{\mathsf{Proto}}(H,l,\text{“}length\text{”}) \\
\qquad\qquad H_1 = H[l \mapsto H(l)[v_{idx} \mapsto \{\; value = v; writable = \mathsf{true}; enumerable = \mathsf{true}; configurable = \mathsf{true} \;\}]] \\
\qquad\qquad H_2 = \begin{cases} H_1[l \mapsto H_1(l)[\text{“}length\text{”} \mapsto H(l)(\text{“}length\text{”})\ \mathsf{with}\ value = v_{idx}+1]] & \text{if} \;\; n_{oldLen} \leq v_{idx} \\ H_1 & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,\text{“}length\text{”}\,\texttt{]}\,\texttt{=}e_2\rrbracket(H,A) = (H_2,A) \quad \text{if} \;\; \mathsf{IsArray}(H,l) \;\wedge\; \underline{\mathsf{CanPut}}(H,l,\text{“}length\text{”}) \;\wedge\; n_{newLen} \geq 0 \\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \\
\qquad\qquad n_{oldLen} = \underline{\mathsf{Proto}}(H,l,\text{“}length\text{”}) \\
\qquad\qquad n_{newLen} = \underline{\mathsf{toNumber}}(\mathcal{V}\llbracket e_2\rrbracket(H,A)) \\
\qquad\qquad H_1 = H[l \mapsto H(l)[\text{“}length\text{”} \mapsto H(l)(\text{“}length\text{”})\ \mathsf{with}\ value = n_{newLen}]] \\
\qquad\qquad H_2 = \begin{cases} \bigsqcup_{x=n_{oldLen}-1\ \mathsf{to}\ n_{newLen}} \underline{\mathsf{Delete}}(H_1,l,x) & \text{if} \;\; n_{newLen} < n_{oldLen} \\ H_1 & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,\text{“}length\text{”}\,\texttt{]}\,\texttt{=}e_2\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; \mathsf{IsArray}(H,l) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge\; \big(n_{newLen} < 0 \vee n_{newLen} \in \{\; \mathsf{NaN}, \mathsf{Inf}, -\mathsf{Inf} \;\}\big) \\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \\
\qquad\qquad n_{newLen} = \underline{\mathsf{toNumber}}(\mathcal{V}\llbracket e_2\rrbracket(H,A)) \\
\qquad\qquad H_1 = \underline{\mathsf{RaiseException}}(H, \mathsf{RangeError})
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket e_1\,\texttt{[}\,e_2\,\texttt{]}\,\texttt{=}e_3\rrbracket(H,A) = (H,A) \quad \text{if} \;\; \neg\underline{\mathsf{CanPut}}(H,l,x) \;\wedge\; v = \mathcal{V}\llbracket e_3\rrbracket(H,A) \\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \\
\qquad\qquad x = \mathcal{V}\llbracket e_2\rrbracket(H,A)
\end{array}
$$

$$
\mathcal{I}_{cp}\llbracket x_1\,\texttt{:=function}\; x_2^{?}\,\texttt{(}fid\texttt{)}\,\rrbracket(H,A) = \left(H_1 \left[\begin{array}{l} l_{new1} \mapsto \underline{\mathsf{NewFunctionObject}}(fid, A_1, l_{new2}, n), \\ l_{new2} \mapsto o_{new} \left[\text{“}constructor\text{”} \mapsto \left\{ \begin{array}{l} value = l_{new1}; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{true} \end{array} \right\} \right] \end{array} \right], A_1 \right)
$$

$$
\begin{array}{l}
\quad \text{where} \;\; l = \underline{\mathsf{TopStack}}(A) \;\wedge\; l_{new1} = \underline{\mathsf{newLocation}}() \wedge l_{new2} = \underline{\mathsf{newLocation}}() \wedge l_{new3} = \underline{\mathsf{newLocation}}() \\
\qquad\qquad H_1 = \underline{\mathsf{VarStoreE}}(H, A, x_1, l_{new1}, \mathsf{true}) \\
\qquad\qquad \wedge\, o_{new} = \underline{\mathsf{NewObject}}(\#ObjProto) \\
\qquad\qquad \wedge\, l_{new3} = \{\;\, x_2 \mapsto \{\;\, value = l_{new1}; writable = \mathsf{false}; enumerable = \mathsf{false}; configurable = \mathsf{false} \;\, \} \;\, \} \\
\qquad\qquad \wedge\, A_1 = \underline{\mathsf{PushStack}}(A, l_{new3}) \\
\qquad\qquad \wedge\, n = |\underline{\mathsf{getArgVars}}_P(fid)|
\end{array}
$$

$$\mathcal{I}_{cp}[\![\texttt{construct}\,(e_1,e_2,e_3)\,]\!](H,A) = \left(H \left[\begin{array}{l} l_{arg} \mapsto H(l_{arg}) \left[callee \mapsto \left\{ \begin{array}{l} value = l_{fun}; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{true} \end{array} \right\} \right], \\ l_{new} \mapsto \left\{ \begin{array}{l} arguments \mapsto \left\{ \begin{array}{l} value = v_{arg}; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @this \mapsto value = l_{this}, \\ @up \mapsto A, \\ @return \mapsto H(\#temp)(@return), \end{array} \right\} \end{array} \right], A_1 \right)$$

where $\underline{\mathsf{HasConstruct}}(H, \mathcal{V}[\![e_1]\!](H,A)) \wedge A_1 = \underline{\mathsf{PushStack}}(\mathcal{V}[\![e_1\,\texttt{[}@scope\texttt{]}\,]\!](H,A), l_{new})$
$\wedge\, l = \mathcal{V}[\![e_1]\!](H,A)$
$\wedge\, arguments = \underline{\mathsf{getArgumentsName}}_P(fid_{callee})$
$\wedge\, v_{arg} = \mathcal{V}[\![e_3]\!](H,A)$
$\wedge\, l_{this} = \underline{\mathsf{getThis}}(\mathcal{V}[\![e_2]\!](H,A))$
$\wedge\, l_{new} = \underline{\mathsf{newLocation}}() \;\wedge\, fid_{callee} = \mathcal{V}[\![e_1\,\texttt{[}@construct\texttt{]}\,]\!](H,A)$
$\wedge\, cp_{\textit{after-call}} = \underline{\mathsf{getAftercallFromCall}}_P(cp)$
$\wedge \hookrightarrow\, := \hookrightarrow \cup \left\{\; (cp, (fid_{callee}, \mathsf{ENTRY})), ((fid_{callee}, \mathsf{EXIT}), cp_{\textit{after-call}}) \;\right\}$
$\wedge \overset{\mathsf{exc}}{\hookrightarrow} := \overset{\mathsf{exc}}{\hookrightarrow} \cup \left\{\; ((fid_{callee}, \mathsf{EXIT\text{-}EXC}), cp_{\textit{after-call}}) \;\right\}$
$\wedge\, \underline{\mathsf{BelongsTo}} := \underline{\mathsf{BelongsTo}} \cup \left\{\; (l_{new}, cp) \;\right\}$

$\mathcal{I}_{cp}[\![\texttt{construct}\,(e_1,e_2,e_3)\,]\!](H,A) = (H_1, A)$ if $\neg\underline{\mathsf{HasConstruct}}(H, v) \vee v \notin \mathsf{Loc}$
where $v = \mathcal{V}[\![e_1]\!](H,A)$
$H_1 = \underline{\mathsf{RaiseException}}(H, \mathsf{TypeError})$
$\mathcal{I}_{cp}[\![\texttt{construct}\,(e_1,e_2,e_3)\,]\!](H,A) = (H_1, A)$ if $exc = \mathcal{V}[\![e_1]\!](H,A)$
where $H_1 = \underline{\mathsf{RaiseException}}(H, exc)$

$$\mathcal{I}_{cp}[\![\texttt{call}\,(e_1,e_2,e_3)\,]\!](H,A) = \left(H \left[\begin{array}{l} l_{arg} \mapsto H(l_{arg}) \left[callee \mapsto \left\{ \begin{array}{l} value = l_{fun}; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{true} \end{array} \right\} \right], \\ l_{new} \mapsto \left\{ \begin{array}{l} arguments \mapsto \left\{ \begin{array}{l} value = l_{arg}; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @this \mapsto value = l_{this}, \\ @up \mapsto A, \\ @return \mapsto H(\#temp)(@return), \end{array} \right\} \end{array} \right], A_1 \right)$$

where $l_{fun} = \mathcal{V}[\![e_1]\!](H,A) \wedge \underline{\mathsf{IsCallable}}(H, l_{fun}) \wedge A_1 = \underline{\mathsf{PushStack}}(\mathcal{V}[\![e_1\,\texttt{[}@scope\texttt{]}\,]\!](H,A), l_{new})$
$\wedge\, arguments = \underline{\mathsf{getArgumentsName}}_P(fid_{callee})$
$\wedge\, l_{arg} = \mathcal{V}[\![e_3]\!](H,A)$
$\wedge\, l_{this} = \underline{\mathsf{getThis}}(\mathcal{V}[\![e_2]\!](H,A))$
$\wedge\, l_{new} = \underline{\mathsf{newLocation}}() \;\wedge\, fid_{callee} = \mathcal{V}[\![e_1\,\texttt{[}@function\texttt{]}\,]\!](H,A)$
$\wedge\, cp_{\textit{after-call}} = \underline{\mathsf{getAftercallFromCall}}_P(cp)$
$\wedge \hookrightarrow\, := \hookrightarrow \cup \left\{\; (cp, (fid_{callee}, \mathsf{ENTRY})), ((fid_{callee}, \mathsf{EXIT}), cp_{\textit{after-call}}) \;\right\}$
$\wedge \overset{\mathsf{exc}}{\hookrightarrow} := \overset{\mathsf{exc}}{\hookrightarrow} \cup \left\{\; ((fid_{callee}, \mathsf{EXIT\text{-}EXC}), cp_{\textit{after-call}}) \;\right\}$
$\wedge\, \underline{\mathsf{BelongsTo}} := \underline{\mathsf{BelongsTo}} \cup \left\{\; (l_{new}, cp) \;\right\}$

$\mathcal{I}_{cp}[\![\texttt{call}\,(e_1,e_2,e_3)\,]\!](H,A) = (H_1, A)$ if $\neg\underline{\mathsf{IsCallable}}(H, v) \vee v \notin \mathsf{Loc}$
where $v = \mathcal{V}[\![e_1]\!](H,A)$
$H_1 = \underline{\mathsf{RaiseException}}(H, \mathsf{TypeError})$
$\mathcal{I}_{cp}[\![\texttt{call}\,(e_1,e_2,e_3)\,]\!](H,A) = (H_1, A)$ if $exc = \mathcal{V}[\![e_1]\!](H,A)$
where $H_1 = \underline{\mathsf{RaiseException}}(H, exc)$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\texttt{after-call(}x\texttt{)}]\!](H,A) = \mathsf{stuck} \quad \text{if } \neg\underline{\mathsf{BelongsTo}}(l, cp_{call})\\
&\quad \text{where } l = \underline{\mathsf{TopStack}}(A)\\
&\qquad\qquad cp_{call} = \underline{\mathsf{getCallFromAfterCall}}_P(cp)\\
&\mathcal{I}_{cp}[\![\texttt{after-call(}x\texttt{)}]\!](H,A) = (H_2, A_1) \quad \text{if } \underline{\mathsf{BelongsTo}}(l, cp_{call})\\
&\quad \text{where } l = \underline{\mathsf{TopStack}}(A)\\
&\qquad\qquad cp_{call} = \underline{\mathsf{getCallFromAfterCall}}_P(cp)\\
&\qquad\qquad A_1 = H(l)(@up)\\
&\qquad\qquad H_1 = \underline{\mathsf{VarStoreE}}(H, A_1, x, H(\#temp)(@return), \mathsf{true})\\
&\qquad\qquad H_2 = H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto H(l)(@return)]]
\end{aligned}
$$

$$
\mathcal{I}_{cp}[\![\texttt{assert(}e_1 \otimes e_2\texttt{)}]\!](H,A) = \mathcal{B}[\![e_1 \otimes e_2]\!](H,A)
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\texttt{catch(}x\texttt{)}]\!](H,A) = (H_2, A)\\
&\quad \text{where } H_1 = \underline{\mathsf{VarStore}}(H, A, x, H(\#temp)(@exception), \mathsf{true}),\\
&\qquad\qquad H_2 = \underline{\mathsf{Delete}}(H_1, \#temp, @exception)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\texttt{return(}e\texttt{)}]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A) \quad \text{if } v = \mathcal{V}[\![e]\!](H,A)\\
&\mathcal{I}_{cp}[\![\texttt{return(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } exc = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{aligned}
$$

$$
\mathcal{I}_{cp}[\![\texttt{return()}]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \mathsf{undefined}]], A)
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\texttt{throw(}e\texttt{)}]\!](H,A) = (H_2, A)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, \mathcal{V}[\![e]\!](H,A)),\\
&\qquad\qquad \wedge\ H_2 = H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto \mathsf{undefined}]]
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{toObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_2, A) \quad \text{if } v = \mathcal{V}[\![e]\!](H,A)\ \wedge\ (H_1, l_{new}) = \underline{\mathsf{toObject}}(H, v)\\
&\quad \text{where } H_2 = \underline{\mathsf{VarStoreE}}(H_1, A, x, l_{new}, \mathsf{true})\\
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{toObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } v = \mathcal{V}[\![e]\!](H,A)\ \wedge\ (\_, exc) = \underline{\mathsf{toObject}}(H, v)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)\\
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{toObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } exc = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{isObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } l = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{VarStoreE}}(H, A, x, \mathsf{true}, \mathsf{true})\\
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{isObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } pv = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{VarStoreE}}(H, A, x, \mathsf{false}, \mathsf{true})\\
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{isObject}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } exc = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{toString}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } v = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } pv = \underline{\mathsf{toPrimitive}}(v)\\
&\qquad\qquad H_1 = \underline{\mathsf{VarStoreE}}(H, A, x, \underline{\mathsf{toString}}(pv), \mathsf{true})\\
&\mathcal{I}_{cp}[\![x\texttt{:}\underline{=\diamond\mathsf{toString}}\texttt{(}e\texttt{)}]\!](H,A) = (H_1, A) \quad \text{if } exc = \mathcal{V}[\![e]\!](H,A)\\
&\quad \text{where } H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{aligned}
$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{toNumber}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; l = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\quad \text{where} \;\; pv = \underline{\mathsf{toPrimitive}}(l)\\
\qquad\qquad\; H_1 = \underline{\mathsf{VarStoreE}}(H,A,x,\underline{\mathsf{toNumber}}(pv),\mathsf{true})\\
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{toNumber}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; pv = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\quad \text{where} \;\; H_1 = \underline{\mathsf{VarStoreE}}(H,A,x,\underline{\mathsf{toNumber}}(pv),\mathsf{true})\\
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{toNumber}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; exc = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\quad \text{where} \;\; H_1 = \underline{\mathsf{RaiseException}}(H,exc)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{toBoolean}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; v = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\quad \text{where} \;\; H_1 = \underline{\mathsf{VarStoreE}}(H,A,x,\underline{\mathsf{toBoolean}}(v),\mathsf{true})\\
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{toBoolean}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_1,A) \quad \text{if} \;\; exc = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\quad \text{where} \;\; H_1 = \underline{\mathsf{RaiseException}}(H,exc)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x_1\,\texttt{:=}\diamond\underline{\mathsf{getBase}}\,\texttt{(}x_2\texttt{)}\,\rrbracket(H,A) = (H_1,A)\\
\quad \text{where} \;\; l_{base} = \underline{\mathsf{LookupBase}}(H,A,x_2)\\
\qquad\qquad\; H_1 = \underline{\mathsf{VarStoreE}}(H,A,x_1,l_{base},\mathsf{true})
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x\,\texttt{:=}\diamond\underline{\mathsf{iteratorInit}}\,\texttt{(}e\texttt{)}\,\rrbracket(H,A) = (H_2,A)\\
\quad \text{where} \;\; l = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\qquad\qquad\; P = \underline{\mathsf{collectProps}}(H,l)\\
\qquad\qquad\; o_{new} = \underline{\mathsf{iteratorInit}}(H(l),P,0)\\
\qquad\qquad\; l_{new} = \underline{\mathsf{newLocation}}()\\
\qquad\qquad\; H_1 = H[l_{new} \mapsto o_{new}]\\
\qquad\qquad\; H_2 = \underline{\mathsf{VarStoreE}}(H_1,A,x,l_{new},\mathsf{true})
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x_1\,\texttt{:=}\diamond\underline{\mathsf{iteratorHasNext}}\,\texttt{(}e,x_2\texttt{)}\,\rrbracket(H,A) = (H_1,A)\\
\quad \text{where} \;\; l_1 = \mathcal{V}\llbracket x_2\rrbracket(H,A)\\
\qquad\qquad\; l_2 = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\qquad\qquad\; i = \underline{\mathsf{next}}(H,H(l_1),H(l_1)(@i),l_2)\\
\qquad\qquad\; b = \begin{cases} \mathsf{true} & \text{if} \;\; i \in \underline{\mathsf{Dom}}(H(l_1))\\ \mathsf{false} & \text{otherwise} \end{cases}\\
\qquad\qquad\; H_1 = \underline{\mathsf{VarStoreE}}(H,A,x_1,b,\mathsf{true})
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket x_1\,\texttt{:=}\diamond\underline{\mathsf{iteratorNext}}\,\texttt{(}e,x_2\texttt{)}\,\rrbracket(H,A) = (H_2,A)\\
\quad \text{where} \;\; l_1 = \mathcal{V}\llbracket x_2\rrbracket(H,A)\\
\qquad\qquad\; l_2 = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\qquad\qquad\; i = \underline{\mathsf{next}}(H,H(l_1),H(l_1)(@i),l_2)\\
\qquad\qquad\; v = H(l_1)(\underline{\mathsf{toString}}(i)))\\
\qquad\qquad\; H_1 = H[l_1 \mapsto H(l_1)[@i \mapsto i+1]]\\
\qquad\qquad\; H_2 = \underline{\mathsf{VarStoreE}}(H_1,A,x_1,v,\mathsf{true})
\end{array}$$

$$\mathcal{V}\llbracket x\rrbracket(H,A) = \underline{\mathsf{Lookup}}(H,A,x)$$

$$\begin{array}{l}
\mathcal{V}\llbracket e_1 \otimes e_2\rrbracket(H,A) = v_1 \otimes v_2 \quad \text{if} \;\; v_1 = \mathcal{V}\llbracket e_1\rrbracket(H,A) \wedge\; v_2 = \mathcal{V}\llbracket e_2\rrbracket(H,A)\\
\mathcal{V}\llbracket e_1 \otimes e_2\rrbracket(H,A) = exc \quad \text{if} \;\; exc = \mathcal{V}\llbracket e_1\rrbracket(H,A)\\
\mathcal{V}\llbracket e_1 \otimes e_2\rrbracket(H,A) = exc \quad \text{if} \;\; v = \mathcal{V}\llbracket e_1\rrbracket(H,A) \wedge\; exc = \mathcal{V}\llbracket e_2\rrbracket(H,A)
\end{array}$$

$$\begin{array}{l}
\mathcal{V}\llbracket \ominus e\rrbracket(H,A) = \ominus v \quad \text{if} \;\; v = \mathcal{V}\llbracket e\rrbracket(H,A)\\
\mathcal{V}\llbracket \ominus e\rrbracket(H,A) = exc \quad \text{if} \;\; exc = \mathcal{V}\llbracket e\rrbracket(H,A)
\end{array}$$

$$\mathcal{V}\llbracket e_1\,\texttt{[}e_2\texttt{]}\,\rrbracket(H,A) = v \quad \text{where} \;\; l = \mathcal{V}\llbracket e_1\rrbracket(H,A) \wedge s = \mathcal{V}\llbracket e_2\rrbracket(H,A) \wedge v = \underline{\mathsf{Proto}}(H,l,s)$$

$$\mathcal{V}\llbracket \mathrm{n}\rrbracket(H,A) = n$$

$$\mathcal{V}\llbracket \text{“s”}\rrbracket(H,A) = s$$

$$\mathcal{V}\llbracket\texttt{true}\rrbracket(H,A) = \mathsf{true}$$

$$\mathcal{V}\llbracket\texttt{false}\rrbracket(H,A) = \mathsf{false}$$

$$\mathcal{V}\llbracket\texttt{null}\rrbracket(H,A) = \mathsf{null}$$

$$\mathcal{V}\llbracket\texttt{this}\rrbracket(H,A) = l_{this} \quad \text{where } \ l_{this} = H(l)(@\mathit{this}) \land l = \underline{\mathsf{TopStack}}(A)$$

$$\begin{aligned}
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \underline{\mathsf{inherit}}(H, H(l_1)(@\mathit{proto}), l_3)\\
&\quad \text{where } \ \underline{\mathsf{HasConstruct}}(H,l_2)\\
&\qquad\qquad \land\, l_1 = \mathcal{V}\llbracket e_1\rrbracket(H,A) \land\ l_2 = \mathcal{V}\llbracket e_2\rrbracket(H,A)\\
&\qquad\qquad \land\, l_3 = \underline{\mathsf{Proto}}(H, l_2, \text{“}\mathit{prototype}\text{”})\\
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \mathsf{false}\\
&\quad \text{where } \ \underline{\mathsf{HasConstruct}}(H,l_2)\\
&\qquad\qquad \land\, \mathcal{V}\llbracket e_1\rrbracket(H,A) \in \mathsf{PValue} \land\ l_2 = \mathcal{V}\llbracket e_2\rrbracket(H,A)\\
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \mathsf{TypeError}\\
&\quad \text{where } \ \mathcal{V}\llbracket e_2\rrbracket(H,A) \in \mathsf{PValue} \lor\ (\neg\underline{\mathsf{HasConstruct}}(H,l_2) \land l_2 = \mathcal{V}\llbracket e_2\rrbracket(H,A))\\
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \mathsf{TypeError}\\
&\quad \text{where } \ \underline{\mathsf{HasConstruct}}(H,l_2) \land\ l_2 = \mathcal{V}\llbracket e_2\rrbracket(H,A) \land\ \underline{\mathsf{Proto}}(H, l_2, \text{“}\mathit{prototype}\text{”}) \in \mathsf{PValue}\\
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \mathit{exc}\\
&\quad \text{where } \ \mathit{exc} = \mathcal{V}\llbracket e_1\rrbracket(H,A) \in \mathsf{Exception}\\
&\mathcal{V}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(H,A) = \mathit{exc}\\
&\quad \text{where } \ v = \mathcal{V}\llbracket e_1\rrbracket(H,A)\ \land\ \mathit{exc} = \mathcal{V}\llbracket e_2\rrbracket(H,A) \in \mathsf{Exception}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{V}\llbracket e_1\,\texttt{in}\,e_2\rrbracket(H,A) = \underline{\mathsf{HasProperty}}(H,l,x)\\
&\quad \text{where } \ v = \mathcal{V}\llbracket e_1\rrbracket(H,A) \land\ l = \mathcal{V}\llbracket e_2\rrbracket(H,A)\\
&\qquad\qquad \land\, x = \underline{\mathsf{toString}}(\underline{\mathsf{toPrimitive}}(v))\\
&\mathcal{V}\llbracket e_1\,\texttt{in}\,e_2\rrbracket(H,A) = \mathsf{TypeError}\\
&\quad \text{where } \ \mathcal{V}\llbracket e_1\rrbracket(H,A) \in \mathsf{Value} \land\ \mathcal{V}\llbracket e_2\rrbracket(H,A) \in \mathsf{PValue}\\
&\mathcal{V}\llbracket e_1\,\texttt{in}\,e_2\rrbracket(H,A) = \mathcal{V}\llbracket e_1\rrbracket(H,A)\\
&\quad \text{where } \ \mathcal{V}\llbracket e_1\rrbracket(H,A) \in \mathsf{Exception}\\
&\mathcal{V}\llbracket e_1\,\texttt{in}\,e_2\rrbracket(H,A) = \mathit{exc}\\
&\quad \text{where } \ v = \mathcal{V}\llbracket e_1\rrbracket(H,A)\ \land\ \mathit{exc} = \mathcal{V}\llbracket e_2\rrbracket(H,A)
\end{aligned}$$

$$\begin{aligned}
&\mathcal{V}\llbracket\texttt{typeof}\,e\rrbracket(H,A) = \underline{\mathsf{TypeTag}}(H,v) \quad \text{if } \ v = \mathcal{V}\llbracket e\rrbracket(H,A)\\
&\mathcal{V}\llbracket\texttt{typeof}\,e\rrbracket(H,A) = \mathit{exc} \quad \text{if } \ \mathit{exc} = \mathcal{V}\llbracket e\rrbracket(H,A)
\end{aligned}$$

$$\begin{aligned}
&\mathcal{B}\llbracket e\rrbracket(H,A) = S\\
&\quad \text{where } \ \mathcal{V}\llbracket e\rrbracket(H,A) = v\\
&\qquad\qquad S = \begin{cases} (H,A) & \underline{\mathsf{toBoolean}}(v) = \mathsf{true}\\ \mathsf{stuck} & \underline{\mathsf{toBoolean}}(v) = \mathsf{false}\end{cases}\\
&\mathcal{B}\llbracket e\rrbracket(H,A) = (H_1[\#\mathit{temp} \mapsto H_1(\#\mathit{temp})[@\mathit{exception} \mapsto l_e]], A)\\
&\quad \text{where } \ \mathcal{V}\llbracket e\rrbracket(H,A) = \mathit{exc}\\
&\qquad\qquad l_e = \underline{\mathsf{newLocation}}()\\
&\qquad\qquad H_1 = H[l_e \mapsto \underline{\mathsf{NewExceptionObject}}(\mathit{exc})]
\end{aligned}$$

$$\begin{aligned}
\ominus \ &::= \ \mathsf{void} \mid \texttt{+} \mid \texttt{-} \mid \texttt{\~{}} \mid \texttt{!}\\
\otimes \ &::= \ \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>}\\
&\quad\ \mid \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \mid \texttt{==} \mid \texttt{!=} \mid \texttt{===} \mid \texttt{!==} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=}
\end{aligned}$$

We consider the collecting semantics of program P that is characterized by an invariant $\llbracket P\rrbracket \in \mathbb{C} \to \wp(\mathsf{State})$, collecting a set of reachable states at each control point. The collecting semantics is defined by the least fixpoint of composition of semantic functions $F_{\textit{control-flow}}, F_{\textit{exception-flow}} \in (\mathbb{C} \to \wp(\mathsf{State})) \to (\mathbb{C} \to \wp(\mathsf{State}))$ such that,

$$\begin{aligned}
\underline{\mathsf{ExcFlow}}(S) &= \{\ (H,A) \mid \mathsf{stuck} \neq (H,A) \in S \land \underline{\mathsf{HasProperty}}(H, \#\mathit{temp}, @\mathit{exception})\ \}\\
\underline{\mathsf{NormalFlow}}(S) &= \{\ (H,A) \mid \mathsf{stuck} \neq (H,A) \in S \land \neg\underline{\mathsf{HasProperty}}(H, \#\mathit{temp}, @\mathit{exception})\ \}\\
f_{cp} &= \mathcal{C}_{cp}(\underline{\mathsf{getCmd}}_P(cp))\\
F_{\textit{control-flow}}(X) &= \lambda cp \in \mathbb{C}. \textstyle\bigcup_{cp' \hookrightarrow cp} f_{cp'}\ (\underline{\mathsf{NormalFlow}}(X(cp')))\ .\\
F_{\textit{exception-flow}}(X) &= \lambda cp \in \mathbb{C}. \textstyle\bigcup_{cp' \overset{\mathsf{exc}}{\hookrightarrow} cp} f_{cp'}\ (\underline{\mathsf{ExcFlow}}(X(cp')))\ .\\
F &= F_{\textit{control-flow}} \circ F_{\textit{exception-flow}}
\end{aligned}$$

Chapter 9

CFG Abstract Semantics

`.../jsaf/analysis/typing/{package, Config}.scala`

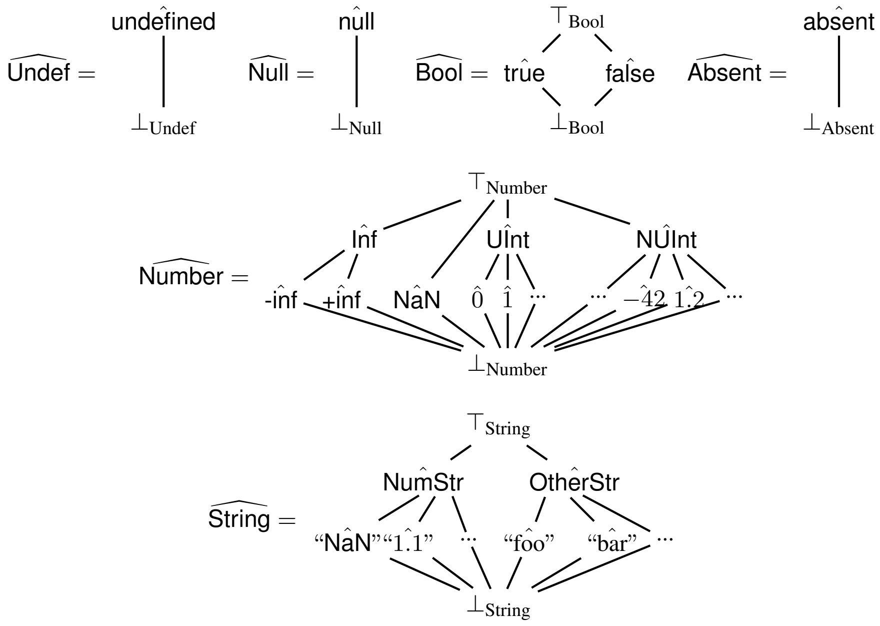
Assumptions and limitations are as follows:

- When a value is updated, a part of $\widehat{\mathsf{PropValue}}$ type value is directly used instead of $\widehat{\mathsf{PropValue}} \times \widehat{\mathsf{Absent}}$. In this case, the $\widehat{\mathsf{Absent}}$ value and the rest parts of $\widehat{\mathsf{PropValue}}$ is considered as $\bot$.
  e.g.) $x \mapsto \langle \hat{v}, \hat{\mathsf{false}}, \hat{\mathsf{false}}\, \hat{\mathsf{false}} \rangle$ means $x \mapsto \langle\langle\langle \hat{v}, \hat{\mathsf{false}}, \hat{\mathsf{false}}\, \hat{\mathsf{false}} \rangle, \bot_{Value}, \bot_{FunctionId} \rangle, \bot_{Absent} \rangle$.
- Semantics for helper functions is not written using denotational semantics(they are not compositional).
- For appropriate type conversion, a subscript is used. In this case, all the implicit values can be considered as $\bot$.
- We maintain mutable inter-procedural edge set ($\overset{\mathsf{ip}}{\hookrightarrow}$) throughout semantics.

9.1 Settings

`.../jsaf/analysis/typing/domain/{package, AbsDomain, DomainPrinter}.scala`[1]

$$
\begin{array}{rlcl}
\hat{cp} \in & \widehat{\mathsf{ControlPoint}} & = & \mathsf{Node} \times \widehat{\mathsf{CallContext}} \\
\hat{cc} \in & \widehat{\mathsf{CallContext}} & & \textit{Parameterized context-sensitivity. See Section 9.4} \\
\hat{S}, (\hat{H}, \hat{C}) \in & \widehat{\mathsf{State}} & = & \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \\
\hat{H} \in & \widehat{\mathsf{Heap}} & = & \widehat{\mathsf{Loc}} \xrightarrow{\text{fin}} \widehat{\mathsf{Obj}} \\
\hat{C} \in & \widehat{\mathsf{Context}} & = & \wp(\widehat{\mathsf{Loc}}) \times \wp(\widehat{\mathsf{Loc}}) \times \wp(\widehat{\mathsf{Address}}) \times \wp(\widehat{\mathsf{Address}}) \\
 & & & \textit{variable environment, this (moved to \#PureLocal), may old, must old} \\
\hat{l}_R, \hat{l}_O, \hat{l} \in & \widehat{\mathsf{Loc}} & = & \widehat{\mathsf{Address}} \times \widehat{\mathsf{RecencyTag}} \\
\hat{a} \in & \widehat{\mathsf{Address}} & ::= & \#\widehat{Global} \mid \#\widehat{StringProto} \mid \#\widehat{BooleanProto} \mid \#\widehat{FunctionProto} \\
 & & & \mid \#\widehat{RefErr} \mid \#\widehat{RangeErr} \mid \#\widehat{TypeErr} \mid \#\widehat{RefErrProto} \\
 & & & \mid \#\widehat{RangeErrProto} \mid \#\widehat{ArrayProto} \mid \#\widehat{TypeErrProto} \\
 & & & \mid \#\widehat{ObjProto} \mid \#\widehat{PureLocal} \mid \#\widehat{GlobalCallsite} \mid \#\widehat{Collapsed} \\
 & & & \mid \hat{a}_1 \mid \cdots \\
 & \widehat{\mathsf{RecencyTag}} & ::= & \widehat{Recent} \mid \widehat{Old} \\
\hat{o} \in & \widehat{\mathsf{Obj}} & = & \mathsf{Prop} \xrightarrow{\text{fin}} \widehat{\mathsf{PropValue}} \times \widehat{\mathsf{Absent}} \\
\hat{propv} \in & \widehat{\mathsf{PropValue}} & = & \widehat{\mathsf{ObjectValue}} \times \widehat{\mathsf{Value}} \times \wp(\mathsf{FunctionId}) \\
\hat{ov} \in & \widehat{\mathsf{ObjectValue}} & = & \widehat{\mathsf{Value}} \times \widehat{\mathsf{Bool}} \times \widehat{\mathsf{Bool}} \times \widehat{\mathsf{Bool}} \\
 & & & value,\ writable,\ enumerable,\ configurable \\
\hat{v} \in & \widehat{\mathsf{Value}} & = & \widehat{\mathsf{PValue}} \times \wp(\widehat{\mathsf{Loc}}) \\
\hat{pv} \in & \widehat{\mathsf{PValue}} & = & \widehat{\mathsf{Undef}} \times \widehat{\mathsf{Null}} \times \widehat{\mathsf{Bool}} \times \widehat{\mathsf{Number}} \times \widehat{\mathsf{String}} \\
\hat{exc} \in & \widehat{\mathsf{Exception}} & ::= & \widehat{\mathsf{Error}} \mid \widehat{\mathsf{EvalError}} \mid \widehat{\mathsf{RangeError}} \mid \widehat{\mathsf{ReferenceError}} \mid \widehat{\mathsf{SyntaxError}} \mid \widehat{\mathsf{TypeError}} \mid \widehat{\mathsf{URIError}} \\
 & \widehat{\mathsf{IPEdge}} & = & \widehat{\mathsf{ControlPoint}} \times \widehat{\mathsf{ControlPoint}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Obj}} \\
\overset{\mathsf{ip}}{\hookrightarrow} \in & \wp(\widehat{\mathsf{IPEdge}}) & & \\
pe \in & \mathsf{PrunExpression} & = & \{x, e_1[e_2]\} \\
re \in & \mathsf{RelExpr} & = & \mathsf{Expression} \ \S\ \mathsf{Expression} \\
\S \in & \mathsf{IROP} & = & \mathsf{IRRelOP} \cup \mathsf{IRObjOP} \\
 & \mathsf{IRRelOP} & = & == \ \mid\ != \ \mid\ === \ \mid\ !== \ \mid\ > \ \mid\ >= \ \mid\ < \ \mid\ <= \ \mid \\
 & \mathsf{IRObjOP} & = & \texttt{in} \mid \texttt{notIn} \mid \texttt{instanceof} \mid \texttt{notInstanceof}
\end{array}
$$



[1] Among the $\widehat{\mathsf{IROP}}$ operators, we handle only $==$, $!=$, $===$, and $!==$ for now as the $\widehat{\underline{\mathsf{K}}}$ function describes in Section 6.2.

9.2 Domain Operators

Heap Order $: \widehat{\textsf{Heap}} \times \widehat{\textsf{Heap}} \rightarrow \textsf{Boolean}$

$$\hat{H}_1 \sqsubseteq \hat{H}_2 \overset{\text{def}}{=} dom(\hat{H}_1) \subseteq dom(\hat{H}_2) \wedge \forall \hat{l} \in dom(\hat{H}_1) : \hat{H}_1(\hat{l}) \sqsubseteq \hat{H}_2(\hat{l})$$

Heap Join $: \widehat{\textsf{Heap}} \times \widehat{\textsf{Heap}} \rightarrow \widehat{\textsf{Heap}}$

$$\hat{H}_1 \sqcup \hat{H}_2 \overset{\text{def}}{=} \forall \hat{l} \in dom(\hat{H}_1) \cup dom(\hat{H}_2) : \begin{cases} \left[\hat{l} \mapsto \hat{H}_1(\hat{l}) \sqcup \hat{H}_2(\hat{l})\right] & \text{if } \hat{l} \in dom(\hat{H}_1) \wedge \hat{l} \in dom(\hat{H}_2) \\ \left[\hat{l} \mapsto \hat{H}_2(\hat{l})\right] & \text{if } \hat{l} \notin dom(\hat{H}_1) \wedge \hat{l} \in dom(\hat{H}_2) \\ \left[\hat{l} \mapsto \hat{H}_1(\hat{l})\right] & \text{if } \hat{l} \in dom(\hat{H}_1) \wedge \hat{l} \notin dom(\hat{H}_2) \end{cases}$$

Heap Domain In $: \widehat{\textsf{Heap}} \times \widehat{\textsf{Loc}} \rightarrow \textsf{Boolean}$

$$\hat{l} \in dom(\hat{H}) \overset{\text{def}}{=} \begin{cases} \textsf{true} & \text{if } \hat{l} \in \{\hat{l}' \mid (\hat{l}', \hat{o}) \in \hat{H}\} \\ \textsf{false} & \text{otherwise} \end{cases}$$

Although $\bot_{Obj}$ is returned for non-existent locations, heap is still partial function.

Heap Lookup $: \widehat{\textsf{Heap}} \times \widehat{\textsf{Loc}} \rightarrow \widehat{\textsf{Obj}}$

$$\hat{H}(\hat{l}) \overset{\text{def}}{=} \begin{cases} \hat{o} & \text{if } (\hat{l}, \hat{o}) \in \hat{H} \\ \bot_{Obj} & \text{otherwise} \end{cases}$$

Heap Update $: \widehat{\textsf{Heap}} \times \widehat{\textsf{Loc}} \times \widehat{\textsf{Obj}} \rightarrow \widehat{\textsf{Heap}}$

$$\hat{H}[\hat{l} \mapsto \hat{o}] \overset{\text{def}}{=} \begin{cases} \{(\hat{l}, \hat{o})\} \cup (\hat{H} - \hat{l}) & \text{if } \hat{l} = \hat{l}_R \wedge \hat{o} \neq \bot_{Obj} \\ \bot_{Heap} & \text{if } \hat{l} = \hat{l}_R \wedge \hat{o} = \bot_{Obj} \\ \{(\hat{l}, \hat{H}(\hat{l}) \sqcup \hat{o})\} \cup (\hat{H} - \hat{l}) & \text{if } \hat{l} = \hat{l}_O \wedge \hat{H}(\hat{l}) \sqcup \hat{o} \neq \bot_{Obj} \\ \bot_{Heap} & \text{if } \hat{l} = \hat{l}_O \wedge \hat{H}(\hat{l}) \sqcup \hat{o} = \bot_{Obj} \end{cases}$$

Context Order $: \widehat{\textsf{Context}} \times \widehat{\textsf{Context}} \rightarrow \textsf{Boolean}$

$$\begin{array}{ll} \hat{C}_1 \sqsubseteq \hat{C}_2 \overset{\text{def}}{=} & \hat{C}_1.3 \subseteq \hat{C}_2.3 \ \wedge \\ & \hat{C}_1.4 \supseteq \hat{C}_2.4 \qquad \textit{order is opposite for must old set} \end{array}$$

Context Join $: \widehat{\textsf{Context}} \times \widehat{\textsf{Context}} \rightarrow \widehat{\textsf{Context}}$

$$\hat{C}_1 \sqcup \hat{C}_2 \overset{\text{def}}{=} \langle \{\}, \{\},\ \hat{C}_1.3 \cup \hat{C}_2.3,\ \hat{C}_1.4 \cap \hat{C}_2.4 \rangle$$

Obj Order $: \widehat{\textsf{Obj}} \times \widehat{\textsf{Obj}} \rightarrow \textsf{Boolean}$

$$\hat{o}_1 \sqsubseteq \hat{o}_2 \overset{\text{def}}{=} \forall x \in dom(\hat{o}_1) \cup dom(\hat{o}_2) : \hat{o}_1(x) \sqsubseteq \hat{o}_2(x)$$

Obj Join $: \widehat{\textsf{Obj}} \times \widehat{\textsf{Obj}} \rightarrow \widehat{\textsf{Obj}}$

$$\hat{o}_1 \sqcup \hat{o}_2 \overset{\text{def}}{=} \forall x \in dom(\hat{o}_1) \cup dom(\hat{o}_2) : [x \mapsto \hat{o}_1(x) \sqcup \hat{o}_2(x)]$$

Obj Domain In $: \widehat{\textsf{Obj}} \times \widehat{\textsf{String}} \rightarrow \widehat{\textsf{Bool}}$

$$\hat{s} \dot{\in} dom(\hat{o}) \overset{\text{def}}{=} \begin{cases} x \dot{\in} dom(\hat{o}) & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\textsf{NumStrSingle}}(x) \\ x \dot{\in} dom(\hat{o}) & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\textsf{OtherStrSingle}}(x) \\ \hat{b}_1 & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\textsf{NumStr}} \\ \hat{b}_2 & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\textsf{OtherStr}} \\ \hat{b}_3 & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \top_{String} \\ \bot_{Bool} & \text{if } \hat{o} = \bot_{Obj} \vee \hat{s} = \bot_{String} \end{cases}$$

$$\text{where } \hat{b}_1 = \begin{cases} \top_{Bool} & \text{if } \hat{o}(@\textit{default\_number}).1.1.1 \not\sqsubseteq \bot_{Value} \\ \top_{Bool} & \text{if } \hat{o}(@\textit{default\_number}).1.1.1 \sqsubseteq \bot_{Value} \\ & \quad \wedge \exists x \in dom(\hat{o}) : x \in \textsf{String} \wedge \text{``}\hat{x}\text{''} \sqsubseteq \widehat{\textsf{NumStr}} \\ \widehat{\textsf{false}} & \text{otherwise} \end{cases}$$

$$\hat{b}_2 = \begin{cases} \top_{Bool} & \text{if } \hat{o}(@\textit{default\_other}).1.1.1 \not\sqsubseteq \bot_{Value} \\ \top_{Bool} & \text{if } \hat{o}(@\textit{default\_other}).1.1.1 \sqsubseteq \bot_{Value} \\ & \quad \wedge \exists x \in dom(\hat{o}) : x \in \textsf{String} \wedge \text{``}\hat{x}\text{''} \sqsubseteq \widehat{\textsf{OtherStr}} \\ \widehat{\textsf{false}} & \text{otherwise} \end{cases}$$

$$\hat{b}_3 = \begin{cases} \top_{Bool} & \text{if } \hat{o}(@\textit{default\_number}).1.1.1 \not\sqsubseteq \bot_{Value} \vee \hat{o}(@\textit{default\_other}).1.1.1 \not\sqsubseteq \bot_{Value} \\ \top_{Bool} & \text{if } \hat{o}(@\textit{default\_number}).1.1.1 \sqsubseteq \bot_{Value} \wedge \hat{o}(@\textit{default\_other}).1.1.1 \sqsubseteq \bot_{Value} \\ & \quad \wedge \exists x \in dom(\hat{o}) : x \in \textsf{String} \\ \widehat{\textsf{false}} & \text{otherwise} \end{cases}$$

Obj Domain In $: \widehat{\mathsf{Obj}} \times \mathsf{Prop} \to \widehat{\mathsf{Bool}}$

$$x \dot{\in} dom(\hat{o}) \overset{\text{def}}{=} \begin{cases} \hat{b} & \text{if } \hat{o} \neq \bot_{Obj} \\ \bot_{Bool} & \text{if } \hat{o} = \bot_{Obj} \end{cases}$$

$$\text{where } \hat{b} = \begin{cases} \widehat{\mathsf{true}} & \text{if } \hat{o}(x) \not\sqsubseteq \bot \wedge \widehat{\mathsf{absent}} \not\sqsubseteq \hat{o}(x).2 \\ \top_{Bool} & \text{if } \hat{o}(x) \not\sqsubseteq \bot \wedge \widehat{\mathsf{absent}} \sqsubseteq \hat{o}(x).2 \\ \top_{Bool} & \text{if } \hat{o}(x) \sqsubseteq \bot \wedge x \in \mathsf{String} \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{NumStr}} \\ & \quad \wedge \hat{o}(@\textit{default\_number}).1.1.1 \not\sqsubseteq \bot_{Value} \\ \top_{Bool} & \text{if } \hat{o}(x) \sqsubseteq \bot \wedge x \in \mathsf{String} \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{OtherStr}} \\ & \quad \wedge \hat{o}(@\textit{default\_other}).1.1.1 \not\sqsubseteq \bot_{Value} \\ \widehat{\mathsf{false}} & \text{if } \hat{o}(x) \sqsubseteq \bot \wedge x \in \mathsf{String} \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{NumStr}} \\ & \quad \wedge \hat{o}(@\textit{default\_number}).1.1.1 \sqsubseteq \bot_{Value} \\ \widehat{\mathsf{false}} & \text{if } \hat{o}(x) \sqsubseteq \bot \wedge x \in \mathsf{String} \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{OtherStr}} \\ & \quad \wedge \hat{o}(@\textit{default\_other}).1.1.1 \sqsubseteq \bot_{Value} \\ \widehat{\mathsf{false}} & \text{if } \hat{o}(x) \sqsubseteq \bot \wedge x \notin \mathsf{String} \\ \widehat{\mathsf{false}} & \text{otherwise} \end{cases}$$

Obj Lookup $: \widehat{\mathsf{Obj}} \times \widehat{\mathsf{String}} \to \widehat{\mathsf{PropValue}} \times \widehat{\mathsf{Absent}}$

$$\hat{o}(\hat{s}) \overset{\text{def}}{=} \begin{cases} \hat{o}(x) & \text{if } \hat{s} = \widehat{\mathsf{NumStrSingle}}(x) \\ \hat{o}(x) & \text{if } \hat{s} = \widehat{\mathsf{OtherStrSingle}}(x) \\ \langle (\bigsqcup_{x \in P_1} \hat{o}(x)).1 \sqcup \hat{o}(@\textit{default\_number}).1, \top_{Absent} \rangle & \text{if } \hat{s} = \widehat{\mathsf{NumStr}} \\ \langle (\bigsqcup_{x \in P_2} \hat{o}(x)).1 \sqcup \hat{o}(@\textit{default\_other}).1, \top_{Absent} \rangle & \text{if } \hat{s} = \widehat{\mathsf{OtherStr}} \\ \left\langle \begin{array}{c} (\bigsqcup_{x \in P_3} \hat{o}(x)).1 \sqcup \hat{o}(@\textit{default\_number}).1 \\ \sqcup \hat{o}(@\textit{default\_other}).1 \end{array}, \top_{Absent} \right\rangle & \text{if } \hat{s} = \top_{String} \\ \bot_{PropValue \times Absent} & \text{if } \hat{s} = \bot_{String} \end{cases}$$

$$\begin{aligned} \text{where } P_1 &= \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{NumStr}}\} \\ P_2 &= \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{OtherStr}}\} \\ P_3 &= \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String}\} \end{aligned}$$

Obj Lookup $: \widehat{\mathsf{Obj}} \times \mathsf{Prop} \to \widehat{\mathsf{PropValue}} \times \widehat{\mathsf{Absent}}$

$$\hat{o}(x) \overset{\text{def}}{=} \begin{cases} \langle pr\hat{o}pv, a\hat{b}s \rangle & \text{if } x \to \langle pr\hat{o}pv, a\hat{b}s \rangle \in \hat{o} \\ \langle \bot_{PropValue}, \bot_{Absent} \rangle & \text{if } x \to \langle pr\hat{o}pv, a\hat{b}s \rangle \notin \hat{o} \wedge x \notin \mathsf{String} \\ \langle pr\hat{o}pv_2, a\hat{b}s_2 \rangle & \text{if } \begin{array}{l} x \to \langle pr\hat{o}pv_1, a\hat{b}s_1 \rangle \notin \hat{o} \wedge x \in \mathsf{String} \\ \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{NumStr}} \wedge @\textit{default\_number} \to \langle pr\hat{o}pv_2, a\hat{b}s_2 \rangle \in \hat{o} \end{array} \\ \langle pr\hat{o}pv_3, a\hat{b}s_3 \rangle & \text{if } \begin{array}{l} x \to \langle pr\hat{o}pv_1, a\hat{b}s_1 \rangle \notin \hat{o} \wedge x \in \mathsf{String} \\ \wedge \alpha(x) \sqsubseteq \widehat{\mathsf{OtherStr}} \wedge @\textit{default\_other} \to \langle pr\hat{o}pv_3, a\hat{b}s_3 \rangle \in \hat{o} \end{array} \end{cases}$$

$$\textit{Obj Update} \quad : \widehat{\mathsf{Obj}} \times \widehat{\mathsf{String}} \times \widehat{\mathsf{PropValue}} \to \widehat{\mathsf{Obj}}$$

$$\hat{o}[\hat{s} \mapsto pr\hat{o}pv] \overset{\mathsf{def}}{=} \begin{cases} \hat{o}[x \mapsto pr\hat{o}pv] & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{NumStrSingle}}(x) \\ \hat{o}[x \mapsto pr\hat{o}pv] & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{OtherStrSingle}}(x) \\ \hat{o}\begin{bmatrix} \forall x \in P_1 : x \mapsto \hat{o}(x) \sqcup pr\hat{o}pv, \\ \textit{@default\_number} \mapsto \hat{o}(\textit{@default\_number}) \sqcup pr\hat{o}pv \end{bmatrix} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{NumStr}} \\ \hat{o}\begin{bmatrix} \forall x \in P_2 : x \mapsto \hat{o}(x) \sqcup pr\hat{o}pv, \\ \textit{@default\_other} \mapsto \hat{o}(\textit{@default\_other}) \sqcup pr\hat{o}pv \end{bmatrix} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{OtherStr}} \\ \hat{o}\begin{bmatrix} \forall x \in P_3 : x \mapsto \hat{o}(x) \sqcup pr\hat{o}pv, \\ \textit{@default\_number} \mapsto \hat{o}(\textit{@default\_number}) \sqcup pr\hat{o}pv, \\ \textit{@default\_other} \mapsto \hat{o}(\textit{@default\_other}) \sqcup pr\hat{o}pv \end{bmatrix} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \top_{String} \\ \bot_{Obj} & \text{if } \hat{o} = \bot_{Obj} \vee \hat{s} = \bot_{String} \end{cases}$$

$$\begin{aligned} \text{where } \; & P_1 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{NumStr}}\} \\ & P_2 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{OtherStr}}\} \\ & P_3 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String}\} \end{aligned}$$

$$\textit{Obj Update} \quad : \widehat{\mathsf{Obj}} \times \mathsf{Prop} \times \widehat{\mathsf{PropValue}} \to \widehat{\mathsf{Obj}}$$

$$\hat{o}[x \mapsto pr\hat{o}pv] \overset{\mathsf{def}}{=} \begin{cases} \{(x, \langle pr\hat{o}pv, \bot_{Absent}\rangle)\} \cup (\hat{o} \setminus \{(x, \langle pr\hat{o}pv', \hat{abs}'\rangle)\}) & \text{if } \hat{o} \neq \bot_{Obj} \\ \bot_{Obj} & \text{if } \hat{o} = \bot_{Obj} \end{cases}$$

$$\textit{Obj Update} \quad : \widehat{\mathsf{Obj}} \times \mathsf{Prop} \times \widehat{\mathsf{PropValue}} \times \widehat{\mathsf{Absent}} \to \widehat{\mathsf{Obj}}$$

$$\hat{o}[x \mapsto \langle pr\hat{o}pv, \hat{abs}\rangle] \overset{\mathsf{def}}{=} \begin{cases} \{(x, \langle pr\hat{o}pv, \hat{abs}\rangle)\} \cup (\hat{o} \setminus \{(x, \langle pr\hat{o}pv', \hat{abs}'\rangle)\}) & \text{if } \hat{o} \neq \bot_{Obj} \\ \bot_{Obj} & \text{if } \hat{o} = \bot_{Obj} \end{cases}$$

$$\textit{Obj Remove} \quad : \widehat{\mathsf{Obj}} \times \widehat{\mathsf{String}} \to \widehat{\mathsf{Obj}}$$

$$\hat{o} - \hat{s} \overset{\mathsf{def}}{=} \begin{cases} \hat{o} - x & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{NumStrSingle}}(x) \\ \hat{o} - x & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{OtherStrSingle}}(x) \\ \hat{o} \sqcup \bigsqcup_{x \in P_1} \{(y, \langle pr\hat{o}pv, \hat{abs}\rangle) \mid (y, \langle pr\hat{o}pv, \hat{abs}\rangle) \in \hat{o} \wedge y \neq x\} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{NumStr}} \\ \hat{o} \sqcup \bigsqcup_{x \in P_2} \{(y, \langle pr\hat{o}pv, \hat{abs}\rangle) \mid (y, \langle pr\hat{o}pv, \hat{abs}\rangle) \in \hat{o} \wedge y \neq x\} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \widehat{\mathsf{OtherStr}} \\ \hat{o} \sqcup \bigsqcup_{x \in P_3} \{(y, \langle pr\hat{o}pv, \hat{abs}\rangle) \mid (y, \langle pr\hat{o}pv, \hat{abs}\rangle) \in \hat{o} \wedge y \neq x\} & \text{if } \hat{o} \neq \bot_{Obj} \wedge \hat{s} = \top_{String} \\ \bot_{Obj} & \text{if } \hat{o} = \bot_{Obj} \vee \hat{s} = \bot_{String} \end{cases}$$

$$\begin{aligned} \text{where } \; & P_1 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{o}(x).1.1.4 \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{NumStr}}\} \\ & P_2 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{o}(x).1.1.4 \wedge \text{“}\hat{x}\text{”} \sqsubseteq \widehat{\mathsf{OtherStr}}\} \\ & P_3 = \{x \mid x \in dom(\hat{o}) \wedge x \in \mathsf{String} \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{o}(x).1.1.4\} \end{aligned}$$

$$\textit{Obj Remove} \quad : \widehat{\mathsf{Obj}} \times \mathsf{Prop} \to \widehat{\mathsf{Obj}}$$

$$\hat{o} - x \overset{\mathsf{def}}{=} \begin{cases} \{(y, \langle pr\hat{o}pv, \hat{abs}\rangle) \mid (y, \langle pr\hat{o}pv, \hat{abs}\rangle) \in \hat{o} \wedge y \neq x\} & \text{if } \hat{o} \neq \bot_{Obj} \\ \bot_{Obj} & \text{if } \hat{o} = \bot_{Obj} \end{cases}$$

$$\S^t \quad : \mathsf{IRRelOP} \to \mathsf{IRRelOP}$$

$$\S^t = \begin{cases} < & \text{if } \S = > \\ <= & \text{if } \S = >= \\ > & \text{if } \S = < \\ >= & \text{if } \S = <= \\ \S & \text{otherwise} \end{cases}$$

9.3 Helper Functions

.../jsaf/analysis/typing/Helper.scala

$\underline{\widehat{\mathsf{CanPut}}}$ $\quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{CanPut}}}(\hat{H}, \hat{l}, \hat{s}) = \underline{\widehat{\mathsf{CanPutHelp}}}(\hat{H}, \hat{l}, \hat{s}, \hat{l})$$

Cycle in prototype chain is detected at implementation level.

$\underline{\widehat{\mathsf{CanPutHelp}}}$ $\quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}}$

$$\begin{aligned}
&\underline{\widehat{\mathsf{CanPutHelp}}}(\hat{H}, \hat{l}_1, \hat{s}, \hat{l}_2) = \hat{b}_1 \sqcup \hat{b}_2 \\
&\quad \text{where } \hat{b}_1 = \begin{cases} \hat{H}(\hat{l}_1)(\hat{s}).1.1.2 \ \textit{// writable attribute} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
&\qquad\quad \hat{L}_{proto} = \hat{H}(\hat{l}_1)(@proto).1.1.1.2 \ \textit{// } \wp(\widehat{\mathsf{Loc}}) \textit{ type} \\
&\qquad\quad \hat{b}_2 = \begin{cases} \hat{b}_3 \sqcup \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{CanPutHelp}}}(\hat{H}, \hat{l}_{proto}, \hat{s}, \hat{l}_2) & \text{if } \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
&\qquad\quad \hat{b}_3 = \begin{cases} \hat{H}(\hat{l}_2)(@extensible).1.2.1.3 & \text{if } \hat{H}(\hat{l}_1)(@proto).1.1.1.1.2 \not\sqsubseteq \bot_{Null} \\ \bot_{Bool} & \text{otherwise} \end{cases}
\end{aligned}$$

$\underline{\widehat{\mathsf{CanPutVar}}}$ $\quad : \widehat{\mathsf{Heap}} \times \mathsf{Prop} \rightarrow \widehat{\mathsf{Bool}}$

$$\begin{aligned}
&\underline{\widehat{\mathsf{CanPutVar}}}(\hat{H}, x) = \hat{b}_1 \sqcup \hat{b}_2 \\
&\quad \text{where } \hat{b}_1 = \begin{cases} \hat{H}(\#\hat{Global}_R)(x).1.1.2 & \text{if } \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#\hat{Global}))) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
&\qquad\quad \hat{b}_2 = \begin{cases} \underline{\widehat{\mathsf{CanPut}}}(\hat{H}, \#\hat{Global}_R, \hat{x}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#\hat{Global}))) \\ \bot_{Bool} & \text{otherwise} \end{cases}
\end{aligned}$$

Temporaries and pure local variables are always mutable in non-strict mode.
In strict-mode, “arguments” is immutable AND pure local, which invalidates current approach.

$\underline{\widehat{\mathsf{CreateMutableBinding}}}$ $\quad : \widehat{\mathsf{Heap}} \times \mathsf{Prop} \times \mathsf{Value} \rightarrow \widehat{\mathsf{Heap}}$

$$\begin{aligned}
&\underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar} \\
&\quad \text{where } \hat{H}_1 = \hat{H}[\#Pur\hat{e}Local_R \mapsto \hat{H}(\#Pur\hat{e}Local_R)[x \mapsto \langle \hat{v}, \bot_{Bool}, \bot_{Bool}, \widehat{\mathsf{false}} \rangle]] \\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar} \\
&\quad \text{where } \hat{H}_1 = \bigsqcup_{\hat{l} \in \hat{H}(\#Pur\hat{e}Local_R)(@env).1.2.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[x \mapsto \langle \hat{v}, \widehat{\mathsf{true}}, \bot_{Bool}, \widehat{\mathsf{false}} \rangle]] \\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar} \\
&\quad \text{where } \hat{H}_1 = \hat{H}[\#Col\hat{l}apsed_O \mapsto \hat{H}(\#Col\hat{l}apsed_O)[x \mapsto \langle \hat{v}, \bot_{Bool}, \bot_{Bool}, \widehat{\mathsf{false}} \rangle]] \\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{GlobalVar} \\
&\quad \text{where } \hat{H}_1 = \hat{H}[\#G\hat{l}obal_R \mapsto \hat{H}(\#G\hat{l}obal_R)[x \mapsto \langle \hat{v}, \widehat{\mathsf{true}}, \widehat{\mathsf{true}}, \widehat{\mathsf{false}} \rangle]]
\end{aligned}$$

$\hat{H}(\hat{l})(\hat{s}).1.1.4$ *means the configurable attribute of the property.*

$\underline{\widehat{\mathsf{Delete}}}$ $\quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \to \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{Delete}}}(\hat{H}, \hat{l}, \hat{s}) = (\hat{H}_1 \sqcup \hat{H}_2, \hat{b}_1 \sqcup \hat{b}_2)$$

$$\text{where } \ (\hat{H}_1, \hat{b}_1) = \begin{cases} (\hat{H}, \widehat{\mathsf{false}}) & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \wedge \ \widehat{\mathsf{false}} \sqsubseteq \hat{H}(\hat{l})(\hat{s}).1.1.4 \\ \bot_{Heap \times Bool} & \text{otherwise} \end{cases}$$

$$(\hat{H}_2, \hat{b}_2) = \begin{cases} (\hat{H}[\hat{l} \mapsto \hat{H}(\hat{l}) - \hat{s}], \widehat{\mathsf{true}}) & \text{if } \begin{array}{l} \left(\begin{array}{l} \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \\ \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{H}(\hat{l})(\hat{s}).1.1.4) \end{array} \right) \\ \vee (\widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s})) \end{array} \\ \bot_{Heap \times Bool} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{DeleteAll}}}$ $\quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \to \widehat{\mathsf{Heap}}$

$$\underline{\widehat{\mathsf{DeleteAll}}}(\hat{H}, \hat{l}, \hat{s}) = \hat{H}_1$$

$$\text{where } \ \hat{H}_2 = \underline{\widehat{\mathsf{Delete}}}(\hat{H}, \hat{l}, \hat{s}).1$$

$$\hat{H}_1 = \begin{cases} \underline{\widehat{\mathsf{DeleteAll}}}(\hat{H}_2, \hat{l}_1, \hat{s}) & \text{if } \ \hat{H}(\hat{l})(@proto).1.1.1.1.2 \sqsubseteq \bot_{Null} \\ & \quad \wedge \hat{H}(\hat{l})(@proto).1.1.1.2 = \{\hat{l}_1\} \\ \hat{H}_2 & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{RaiseException}}}$ $\quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \wp(\widehat{\mathsf{Exception}}) \to \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}}$

$$\underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \widehat{es}) = (\hat{H}_1, \hat{C}_1)$$

$$\text{where } \ \hat{v}_{old} = \hat{H}(\#\widehat{PureLocal}_R)(@exception\_all).1.2$$

$$\hat{v}_e = \langle \bot_{PValue}, \textstyle\bigsqcup_{\widehat{exc} \in \widehat{es}} \underline{\widehat{\mathsf{NewExceptionLoc}}}(\widehat{exc}) \rangle$$

$$\hat{H}_e = \hat{H}\left[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R) \begin{bmatrix} @exception \mapsto \hat{v}_e, \\ @exception\_all \mapsto \hat{v}_e \sqcup \hat{v}_{old} \end{bmatrix} \right]$$

$$(\hat{H}_1, \hat{C}_1) = \begin{cases} (\hat{H}_e, \hat{C}) & \text{if } \ \widehat{es} \neq \{\} \\ (\bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{NewExceptionLoc}}}$ $\quad : \widehat{\mathsf{Exception}} \to \widehat{\mathsf{Loc}}$

$$\underline{\widehat{\mathsf{NewExceptionLoc}}}(\hat{H}, \widehat{exc}) = \begin{cases} \#\widehat{Err}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{Error}} \\ \#\widehat{EvalErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{EvalError}} \\ \#\widehat{RangeErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{RangeError}} \\ \#\widehat{RefErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{ReferenceError}} \\ \#\widehat{SyntaxErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{SyntaxError}} \\ \#\widehat{TypeErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{TypeError}} \\ \#\widehat{URIErr}_O & \text{if } \ \widehat{exc} = \widehat{\mathsf{URIError}} \end{cases}$$

$\underline{\widehat{\mathsf{getRel}}}$

$$
\begin{array}{l}
: \mathsf{RelExpr} \rightarrow \widehat{\mathsf{State}} \rightarrow \wp(\mathsf{RelExpr}) \\
\begin{array}{lcl}
\underline{\widehat{\mathsf{getRel}}}(pe\S e, \hat{S}) & = & \{pe\S e\} \\
\underline{\widehat{\mathsf{getRel}}}((e_1 + e_2)\S e_3,\, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S(e_3 - e_2), \hat{S}) \cup \underline{\widehat{\mathsf{getRel}}}(e_2\S(e_3 - e_1), \hat{S}) \quad \text{if } \underline{\widehat{\mathsf{validity}}}_3(e_1, e_2, e_3, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((e_1 - e_2)\S e_3, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S(e_3 + e_2), \hat{S}) \cup \underline{\widehat{\mathsf{getRel}}}(e_2\S^t(e_1 - e_3), \hat{S}) \quad \text{if } \underline{\widehat{\mathsf{validity}}}_3(e_1, e_2, e_3, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((n * e_1)\S e_2, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}((e_1 * n)\S e_2, \hat{S}) \quad \text{if } \underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((e_1 * n)\S e_2, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S(e_2/n), \hat{S}) \quad \text{if } n > 0 \wedge \underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((e_1 * n)\S e_2, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S^t(e_2/n), \hat{S}) \quad \text{if } n < 0 \wedge \underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((e_1/n)\S e_2, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S(e_2 * n), \hat{S}) \quad \text{if } n > 0 \wedge \underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, \hat{S}) \\
\underline{\widehat{\mathsf{getRel}}}((e_1/n)\S e_2, \hat{S}) & = & \underline{\widehat{\mathsf{getRel}}}(e_1\S^t(e_2 * n), \hat{S}) \quad \text{if } n < 0 \wedge \underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, \hat{S})
\end{array} \\
\underline{\widehat{\mathsf{getRel}}}(re) \quad = \quad \emptyset \quad \text{otherwise}
\end{array}
$$

$\underline{\widehat{\mathsf{getThis}}}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Value}} \rightarrow \wp(\widehat{\mathsf{Loc}}) \\
\underline{\widehat{\mathsf{getThis}}}(\hat{H}, \hat{v}) = \hat{L}_1 \cup \hat{L}_2 \cup \hat{L}_3 \\
\quad \text{where } \hat{L}_1 = \begin{cases} \{\#G\hat{l}obal_R\} & \text{if } \widehat{\mathsf{undefined}} \sqsubseteq \hat{v}.1.1 \vee \widehat{\mathsf{null}} \sqsubseteq \hat{v}.1.2 \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \hat{L}_2 = \begin{cases} \{\#G\hat{l}obal_R\} & \text{if } \exists\, \hat{l} \in \hat{v}.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsObject}}}(\hat{h}, \hat{l}) \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \hat{L}_3 = \{\hat{l} \in \hat{v}.2 \mid \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsObject}}}(\hat{h}, \hat{l})\}
\end{array}
$$

$\underline{\widehat{\mathsf{HasConstruct}}}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}} \\
\underline{\widehat{\mathsf{HasConstruct}}}(\hat{H}, \hat{l}) = \hat{b}_1 \sqcup \hat{b}_2 \\
\quad \text{where } \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (@construct \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
\qquad\qquad \hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \widehat{\mathsf{false}} \sqsubseteq (@construct \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{HasInstance}}}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}} \\
\underline{\widehat{\mathsf{HasConstruct}}}(\hat{H}, \hat{l}) = \hat{b}_1 \sqcup \hat{b}_2 \\
\quad \text{where } \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (@hasinstance \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
\qquad\qquad \hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \widehat{\mathsf{false}} \sqsubseteq (@hasinstance \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases}
\end{array}
$$

Cycle in prototype chain is detected at implementation level.

$\underline{\widehat{\mathsf{HasProperty}}}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \widehat{\mathsf{Bool}} \\
\underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}, \hat{s}) = \hat{b}_1 \sqcup \hat{b}_2 \\
\text{where } \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
\qquad\quad \hat{L}_{proto} = \hat{H}(\hat{l})(@proto).1.1.1.2 \\
\qquad\quad \hat{b}_2 = \begin{cases} \hat{b}_3 \sqcup \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}_{proto}, \hat{s}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \\ \bot_{Bool} & \text{otherwise} \end{cases} \\
\qquad\quad \hat{b}_3 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \hat{H}(\hat{l}_1)(@proto).1.1.1.1.2 \not\sqsubseteq \bot_{Null} \\ \bot_{Bool} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{HasOwnProperty}}}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \widehat{\mathsf{Bool}} \\
\underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) = (\hat{s} \dot{\in} dom(\hat{h}(\hat{l})))
\end{array}
$$

$\underline{\widehat{\mathsf{inherit}}}$

Cycle in prototype chain is detected at implementation level.

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Value}}$

$$\underline{\widehat{\mathsf{inherit}}}(\hat{H}, \hat{l}_1, \hat{l}_2) = \begin{cases} \widehat{\mathsf{true}} & \text{if } \hat{l}_1 \hat{=} \hat{l}_2 \\ \hat{v}_1 \sqcup \bigsqcup_{\hat{l} \in \hat{H}(\hat{l}_1)(@proto).1.1.1.2} \underline{\widehat{\mathsf{inherit}}}(\hat{H}, \hat{l}, \hat{l}_2) & \text{if } \hat{l}_1 \hat{\neq} \hat{l}_2 \end{cases}$$

$$\text{where } \hat{v}_1 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \hat{H}(\hat{l}_1)(@proto).1.1.1.1.2 \not\sqsubseteq \bot_{Null} \\ \bot_{Value} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{inheritProto}}_1}$

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Bool}} \rightarrow \wp(\widehat{\mathsf{Loc}})$

$$\underline{\widehat{\mathsf{inheritProto}}_1}(\hat{H}, \hat{l}_1, \hat{l}_2, \hat{b}) = \hat{L}$$

$$\text{where } \hat{L} = \begin{cases} \{\hat{l}_1\} & \text{if } \hat{b} \sqsubseteq \underline{\widehat{\mathsf{inherit}}}(\hat{H}, \hat{l}_1, \hat{l}_2) \\ \{\} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{inheritProto}}_2}$

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Bool}} \rightarrow \wp(\widehat{\mathsf{Loc}})$

$$\underline{\widehat{\mathsf{inheritProto}}_2}(\hat{H}, \hat{l}_1, \hat{l}_2, \hat{b}) = \hat{L}$$

$$\text{where } \hat{L} = \begin{cases} \{\hat{l}_2\} & \text{if } \hat{b} \sqsubseteq \underline{\widehat{\mathsf{inherit}}}(\hat{H}, \hat{l}_1, \hat{l}_2) \\ \{\} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{IsArray}}}$

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}}$

$\hat{H}(\hat{l})(@class).1.2$ *is the* $\widehat{\mathsf{Value}}$

$$\underline{\widehat{\mathsf{IsArray}}}(\hat{H}, \hat{l}) = \hat{b}_1 \sqcup \hat{b}_2$$

$$\text{where } \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \text{``}\widehat{Array}\text{''} \sqsubseteq \hat{H}(\hat{l})(@class).1.2 \\ \bot_{Bool} & \text{otherwise} \end{cases}$$

$$\hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \text{``}\widehat{Array}\text{''} \neq \hat{H}(\hat{l})(@class).1.2 \\ \bot_{Bool} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{IsArrayIndex}}}$

$: \widehat{\mathsf{String}} \rightarrow \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{IsArrayIndex}}}(\hat{s}) = \begin{cases} \top_{Bool} & \text{if } \hat{s} = \top_{String} \\ \top_{Bool} & \text{if } \hat{s} = \widehat{\mathsf{NumStr}} \\ \widehat{\mathsf{false}} & \text{if } \hat{s} = \widehat{\mathsf{OtherStr}} \\ \widehat{\mathsf{true}} & \text{if } \hat{s} = \widehat{\mathsf{NumStrSingle}}(s) \wedge 0 \leq \mathsf{ToNumber}(s) < 2^{32} - 1 \\ \top_{Bool} & \text{if } \hat{s} = \widehat{\mathsf{NumStrSingle}}(s) \wedge (\mathsf{ToNumber}(s) < 0 \vee < 2^{32} - 1 \leq \mathsf{ToNumber}(s)) \\ \widehat{\mathsf{false}} & \text{if } \hat{s} = \widehat{\mathsf{OtherStrSingle}} \\ \bot_{Bool} & \text{if } \hat{s} = \bot_{String} \end{cases}$$

$\underline{\widehat{\mathsf{IsCallable}}}$

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{IsCallable}}}(\hat{H}, \hat{l}) = \hat{b}_1 \sqcup \hat{b}_2$$

$$\text{where } \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (@function \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases}$$

$$\hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \widehat{\mathsf{false}} \sqsubseteq (@function \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Bool} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{IsObject}}}$

$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{IsObject}}}(\hat{H}, \hat{l}) = @class \ \dot{\in} \ dom(\hat{h}(\hat{l}))$$

$$\underline{\widehat{\mathsf{K}}} \quad : \mathsf{IRRelOP} \to \widehat{\mathsf{Value}} \to \widehat{\mathsf{Value}} \times \widehat{\mathsf{Absent}}$$

$$\underline{\widehat{\mathsf{K}}}_{!==} \hat{v}_1 = (\top_{Value}, \hat{\mathsf{absent}})$$

$$\underline{\widehat{\mathsf{K}}}_{===} \hat{v}_1 = (\hat{v}_1, a\hat{b}s)$$

$$\text{where } \; a\hat{b}s = \begin{cases} \hat{\mathsf{absent}} & \text{if } \; \hat{\mathsf{undefined}} \sqsubseteq \hat{v}_1.1.1 \\ \bot_{Absent} & \text{otherwise} \end{cases}$$

$$\underline{\widehat{\mathsf{K}}}_{!=} \hat{v}_1 = (\top_{Value}, \hat{\mathsf{absent}})$$

$$\underline{\widehat{\mathsf{K}}}_{==} \hat{v}_1 = (\langle\langle \hat{v}_1.1.1 \sqcup \hat{pv}_1, \hat{v}_1.1.2 \sqcup \hat{pv}_2, \hat{v}_1.1.3 \sqcup \hat{pv}_3, \hat{v}_1.1.4 \sqcup \hat{pv}_4, \top_{String}\rangle, \top_{\widehat{\mathsf{Loc}}}\rangle, a\hat{b}s)$$

$$\text{where } \; a\hat{b}s = \begin{cases} \hat{\mathsf{absent}} & \text{if } \; \hat{\mathsf{undefined}} \sqsubseteq \hat{v}_1.1.1 \vee \hat{\mathsf{null}} \sqsubseteq \hat{v}_1.1.2 \\ \bot_{Absent} & \text{otherwise} \end{cases}$$

$$n_1 = \begin{cases} \hat{1} & \text{if } \; \hat{\mathsf{true}} \sqsubseteq \hat{v}_1.1.3 \\ \bot_{Number} & \text{otherwise} \end{cases}$$

$$n_2 = \begin{cases} \hat{0} & \text{if } \; \hat{\mathsf{false}} \sqsubseteq \hat{v}_1.1.3 \\ \bot_{Number} & \text{otherwise} \end{cases}$$

$$n_3 = \underline{\widehat{\mathsf{Str2Num}}}((\hat{v}_1.1.5)_{PV\hat{a}lue})$$

$$n_4 = \begin{cases} \bot_{Number} & \text{if } \; \hat{v}_1.1.4 \sqsubseteq N\hat{a}N \\ \hat{v}_1.1.4 & \text{otherwise} \end{cases}$$

$$\hat{pv}_1 = \begin{cases} \hat{\mathsf{undefined}} & \text{if } \; \hat{\mathsf{null}} \sqsubseteq \hat{v}_1.1.2 \\ \bot_{Undef} & \text{otherwise} \end{cases}$$

$$\hat{pv}_2 = \begin{cases} \hat{\mathsf{null}} & \text{if } \; \hat{\mathsf{undefined}} \sqsubseteq \hat{v}_1.1.1 \\ \bot_{Null} & \text{otherwise} \end{cases}$$

$$\hat{pv}_3 = \begin{cases} \top_{Bool} & \text{if } \; \hat{\mathsf{UINT}} \sqsubseteq \hat{v}_1.1.4 \vee \hat{v}_1.2 \neq \emptyset \\ \hat{\mathsf{true}} & \text{if } \; \hat{v}_1.2 = \emptyset \wedge (\hat{1} \sqsubseteq \hat{v}_1.1.4 \vee \hat{1} \sqsubseteq \underline{\widehat{\mathsf{Str2Num}}}((\hat{v}_1.1.5)_{PV\hat{a}lue})) \\ \hat{\mathsf{false}} & \text{if } \; \hat{v}_1.2 = \emptyset \wedge (\hat{0} \sqsubseteq \hat{v}_1.1.4 \vee \hat{0} \sqsubseteq \underline{\widehat{\mathsf{Str2Num}}}((\hat{v}_1.1.5)_{PV\hat{a}lue})) \\ \bot_{Bool} & \text{otherwise} \end{cases}$$

$$\hat{pv}_4 = \begin{cases} n_1 \sqcup n_2 \sqcup n_3 \sqcup n_4 & \text{if } \; \hat{v}_1.2 = \emptyset \\ \top_{Number} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{Lookup}}}$

$$: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \to \widehat{\mathsf{Value}} \times \wp(\widehat{\mathsf{Exception}})$$

$$\underline{\widehat{\mathsf{Lookup}}}(\hat{H}, x) = (\hat{H}(\#Pur\hat{e}Local_R)(x).1.1.1, \{\}) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar}$$

$$\underline{\widehat{\mathsf{Lookup}}}(\hat{H}, x) = (\textstyle\bigsqcup_{\hat{l} \in \hat{H}(\#Pur\hat{e}Local)(@env).1.2.2} \underline{\widehat{\mathsf{LookupL}}}(\hat{H}, \hat{l}, x), \{\}) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar}$$

$$\underline{\widehat{\mathsf{Lookup}}}(\hat{H}, x) = (\hat{H}(\#Col\hat{l}apsed_O)(x).1.1.1, \{\}) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar}$$

$$\underline{\widehat{\mathsf{Lookup}}}(\hat{H}, x) = \underline{\widehat{\mathsf{LookupG}}}(\hat{H}, x) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{GlobalVar}$$

$\underline{\widehat{\mathsf{LookupG}}}$

$$: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \to \widehat{\mathsf{Value}} \times \wp(\widehat{\mathsf{Exception}})$$

$$\underline{\widehat{\mathsf{LookupG}}}(\hat{H}, x) = (\hat{v}_1 \sqcup \hat{v}_2, \hat{es})$$

$$\text{where} \;\; \hat{v}_1 = \begin{cases} \hat{H}(\#G\hat{l}obal_R)(x).1.1.1 & \text{if} \;\; \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#G\hat{l}obal))) \\ \bot_{Value} & \text{otherwise} \end{cases}$$

$$(\hat{v}_2, \hat{es}) = \begin{cases} (\hat{v}_3, e\hat{x}c) & \text{if} \;\; \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#G\hat{l}obal))) \\ (\bot_{Value}, \{\}) & \text{otherwise} \end{cases}$$

$$\hat{L}_{proto} = \hat{H}(\#G\hat{l}obal_R)(@pr\hat{o}to).1.1.1.2$$

$$\hat{v}_3 = \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \begin{cases} \underline{\mathsf{Proto}}(\hat{H}, \hat{l}_{proto}, \hat{x}) & \text{if} \;\; \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}_{proto}, x) \\ \bot_{Value} & \text{otherwise} \end{cases}$$

$$e\hat{x}c = \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \begin{cases} \{\mathsf{Refere\hat{n}ceError}\} & \text{if} \;\; \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}_{proto}, x) \\ \bot_{Exception} & \text{otherwise} \end{cases}$$

Cycle in scope chain is detected at implementation level.

$\underline{\widehat{\mathsf{LookupL}}}$

$$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \mathsf{Prop} \to \widehat{\mathsf{Value}}$$

$$\underline{\widehat{\mathsf{LookupL}}}(\hat{H}, \hat{l}, x) = \hat{v}_1 \sqcup \hat{v}_2$$

$$\text{where} \;\; \hat{v}_1 = \begin{cases} \hat{H}(\hat{l})(x).1.1.1 & \text{if} \;\; \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Value} & \text{otherwise} \end{cases}$$

$$\hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2$$

$$\hat{v}_2 = \begin{cases} \bigsqcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{LookupL}}}(\hat{H}, \hat{l}_{outer}, x) & \text{if} \;\; \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Value} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{LookupBase}}}$

$$: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}})$$

$$\underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, x) = \{\#Pur\hat{e}Local_R\} \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar}$$

$$\underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, x) = \textstyle\bigcup_{\hat{l} \in \hat{H}(\#Pur\hat{e}Local_R)(@env).1.2.2} \underline{\widehat{\mathsf{LookupBaseL}}}(\hat{H}, \hat{l}, x) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar}$$

$$\underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, x) = \{\#Col\hat{l}apsed_O\} \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar}$$

$$\underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, x) = \underline{\widehat{\mathsf{LookupBaseG}}}(\hat{H}, x) \quad \text{if} \;\; \underline{\mathsf{getVarKind}}_P(x) = \mathsf{GlobalVar}$$

$\underline{\widehat{\mathsf{LookupBaseG}}}$

$$: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}})$$

$$\underline{\widehat{\mathsf{LookupBaseG}}}(\hat{H}, x) = \hat{L}_1 \cup \hat{L}_2$$

$$\text{where} \;\; \hat{L}_1 = \begin{cases} \{\#G\hat{l}obal_R\} & \text{if} \;\; \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#G\hat{l}obal_R)) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{L}_2 = \begin{cases} \hat{L}_3 & \text{if} \;\; \mathsf{false} \sqsubseteq (x \dot{\in} dom(\hat{H}(\#G\hat{l}obal_R)) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{L}_{proto} = \hat{H}(\#G\hat{l}obal_R)(@proto).1.1.1.2$$

$$\hat{L}_3 = \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{ProtoBase}}}(\hat{H}, \hat{l}_{proto}, \hat{x})$$

Cycle in scope chain is detected at implementation level.

$\underline{\widehat{\mathsf{LookupBaseL}}}$

$$: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}})$$

$$\underline{\widehat{\mathsf{LookupBaseL}}}(\hat{H}, \hat{l}, x) = \hat{L}_1 \cup \hat{L}_2$$

$$\text{where} \;\; \hat{L}_1 = \begin{cases} \{\hat{l}\} & \text{if} \;\; \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2$$

$$\hat{L}_2 = \begin{cases} \bigcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{LookupBaseL}}}(\hat{H}, \hat{l}_{outer}, x) & \text{if} \;\; \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \\ \{\} & \text{otherwise} \end{cases}$$

$\underline{\widehat{\mathsf{NewBoolean}}}$ $: \widehat{\mathsf{Value}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewBoolean}}}(\hat{v}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Boolean\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#BoolProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value}, \\ @primitive \mapsto \hat{v} \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewNumber}}}$ $: \widehat{\mathsf{Value}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewNumber}}}(\hat{v}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Number\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#NumProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value}, \\ @primitive \mapsto \hat{v} \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewString}}}$ $: \widehat{\mathsf{Value}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewString}}}(\hat{v}) = \hat{o}_1 \sqcup \hat{o}_2$$

$$\text{where } \hat{s} = \hat{v}.1.5 \ \wedge \ \hat{v}_{len} = length(\hat{s})$$

$$\hat{o}_1 = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}String\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#StrProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value}, \\ @primitive \mapsto \hat{v}, \\ \text{“}length\text{”} \mapsto \langle (\hat{v}_{len})_{Value}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle \end{array} \right\}$$

$$\hat{o}_2 = \left\{ \text{“}i\text{”} \mapsto \langle (\hat{v}_{char})_{Value}, \widehat{\mathsf{false}}, \widehat{\mathsf{true}}, \widehat{\mathsf{false}} \rangle \ \middle| \ \begin{array}{l} 0 \le i \\ \wedge \, \exists l \in \gamma(\hat{v}_{len}).i < l \\ \wedge \, \hat{v}_{char} = charAt(\hat{s}, i) \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewDeclEnvRecord}}}$ $: \widehat{\mathsf{Value}} \to \widehat{\mathsf{Obj}}$

outer is either location set or null value

$$\underline{\widehat{\mathsf{NewDeclEnvRecord}}}(\hat{v}) = \{ \ @outer \mapsto \hat{v} \ \}$$

$\underline{\widehat{\mathsf{NewObject}}}$ $: \widehat{\mathsf{Loc}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewObject}}}(\hat{l}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Object\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\hat{l}\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value} \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewArgObject}}}$ $: \widehat{\mathsf{Number}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewArgObject}}}(\hat{n}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Arguments\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#ObjProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ \text{“}length\text{”} \mapsto \langle \hat{n}_{Value}, \widehat{\mathsf{true}}, \widehat{\mathsf{false}}, \widehat{\mathsf{true}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value} \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewArrayObject}}}$ $: \widehat{\mathsf{Number}} \to \widehat{\mathsf{Obj}}$

$$\underline{\widehat{\mathsf{NewArrayObject}}}(\hat{n}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Array\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#ArrayProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ \text{“}length\text{”} \mapsto \langle \hat{n}_{Value}, \widehat{\mathsf{true}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value} \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewFunctionObject}}}$ $: \mathsf{FunctionId} \times \widehat{\mathsf{Value}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Number}} \to \widehat{\mathsf{Obj}}$

scope is either location set or null value

$$\underline{\widehat{\mathsf{NewFunctionObject}}}(fid, \hat{v}, \hat{l}, \hat{n}) = \left\{ \begin{array}{l} @class \mapsto \widehat{\text{“}Function\text{”}}_{Value}, \\ @proto \mapsto \langle \langle \bot_{PValue}, \{\widehat{\#FunctionProto}_R\} \rangle, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ @extensible \mapsto \widehat{\mathsf{true}}_{Value}, \\ @function \mapsto \{fid\}, \\ @construct \mapsto \{fid\}, \\ @hasinstance \mapsto \top_{Null}, \\ @scope \mapsto \hat{v}, \\ \text{“}prototype\text{”} \mapsto \langle \langle \bot_{PValue}, \{\hat{l}\} \rangle, \widehat{\mathsf{true}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle, \\ \text{“}length\text{”} \mapsto \langle \hat{n}_{Value}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}}, \widehat{\mathsf{false}} \rangle \end{array} \right\}$$

$\underline{\widehat{\mathsf{NewPureLocal}}}$ $: \widehat{\mathsf{Value}} \times \wp(\widehat{\mathsf{Loc}}) \to \widehat{\mathsf{Obj}}$

env is either location set or null value

$$\underline{\widehat{\mathsf{NewPureLocal}}}(\hat{v}_{env}, \hat{L}_{this}) = \left\{ \begin{array}{l} @env \mapsto \hat{v}_{env}, \\ @this \mapsto \hat{L}_{this}, \\ @exception \mapsto \bot_{PropValue}, \\ @exception\_all \mapsto \bot_{PropValue}, \\ @return \mapsto \widehat{\mathsf{undefined}}_{Value} \end{array} \right\}$$

$$\underline{\widehat{\mathsf{Oldify}}} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Address}} \rightarrow \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}}$$

$$\underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}) = \begin{cases} (\hat{H}_1, \hat{C}_1) & \text{if } \hat{C} \neq \bot_{Context} \\ (\bot_{Heap}, \bot_{Context}) & \text{if } \hat{C} = \bot_{Context} \end{cases}$$

$$\begin{aligned} \text{where } & \hat{l}_R = (\hat{a}, \widehat{Recent}) \wedge \hat{l}_O = (\hat{a}, \widehat{Old}) \\ & \wedge \hat{H}_1 = \begin{cases} (\hat{H}[\hat{l_O} \mapsto \hat{H}(\hat{l_R})] - \hat{l}_R)\{\hat{l}_O/\hat{l}_R\} & \text{if } \hat{l_R} \in dom(\hat{H}) \\ \hat{H}\{\hat{l}_O/\hat{l}_R\} & \text{if } \hat{l_R} \notin dom(\hat{H}) \end{cases} \\ & \wedge \hat{C}_1 = \langle \{\}, \{\}, \hat{C}.3 \cup \{\hat{a}\}, \hat{C}.4 \cup \{\hat{a}\} \rangle \end{aligned}$$

At function return, this method oldifies bypassed pure local object.

$$\underline{\widehat{\mathsf{FixOldify}}} \quad : \widehat{\mathsf{Context}} \times \widehat{\mathsf{Obj}} \times \wp(\widehat{\mathsf{Address}}) \times \wp(\widehat{\mathsf{Address}}) \rightarrow \widehat{\mathsf{Context}} \times \widehat{\mathsf{Obj}}$$

$$\underline{\widehat{\mathsf{FixOldify}}}(\hat{C}_0, \hat{o}_0, \hat{A}_{may}, \hat{A}_{must}) = \begin{cases} (\hat{C}_n, \hat{o}_n) & \text{if } \hat{C} \neq \bot_{Context} \\ (\bot_{Context}, \bot_{Obj}) & \text{if } \hat{C} = \bot_{Context} \end{cases}$$

$$\begin{aligned} & \text{where } \hat{a}_1 \cdots \hat{a}_n = \hat{A}_{may} \wedge \\ & \forall 1 \leq i \leq n. \\ & \quad \hat{l}_{R_i} = (\hat{a}_i, \widehat{Recent}) \wedge \hat{l}_{O_i} = (\hat{a}_i, \widehat{Old}) \wedge \\ & \quad \hat{C}_i = \begin{cases} \langle \{\}, \{\}, \hat{C}_{i-1}.3 \cup \{\hat{a}_i\}, \hat{C}_{i-1}.4 \cup \{\hat{a}_i\} \rangle & \text{if } a_i \in \hat{A}_{must} \\ \langle \{\}, \{\}, \hat{C}_{i-1}.3 \cup \{\hat{a}_i\}, \hat{C}_{i-1}.4 \rangle & \text{if } a_i \notin \hat{A}_{must} \end{cases} \\ & \quad \hat{o}_i = \begin{cases} \hat{o}_i = \hat{o}_{i-1}\{\hat{l}_{O_i}/\hat{l}_{R_i}\} & \text{if } a_i \in \hat{A}_{must} \\ \hat{o}_i = \hat{o}_{i-1}\{\{\hat{l}_{O_i}, \hat{l}_{R_i}\}/\hat{l}_{R_i}\} & \text{if } a_i \notin \hat{A}_{must} \end{cases} \end{aligned}$$

Cycle in prototype chain is detected at implementation level.

$$\underline{\widehat{\mathsf{Proto}}} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \widehat{\mathsf{Value}}$$

$$\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \hat{s}) = \hat{v}_1 \sqcup \hat{v}_2$$

$$\begin{aligned} \text{where } & \hat{v}_1 = \begin{cases} \hat{H}(\hat{l})(\hat{s}).1.1.1 & \widehat{\mathsf{true}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l})))) \\ \bot_{Value} & \text{otherwise} \end{cases} \\ & \hat{L}_{proto} = \hat{H}(\hat{l})(@proto).1.1.1.2 \\ & \hat{v}_2 = \begin{cases} \hat{v}_3 \sqcup \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}_{proto}, \hat{s}) & \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Value} & \text{otherwise} \end{cases} \\ & \hat{v}_3 = \begin{cases} \widehat{\mathsf{undefined}}_{Value} & \text{if } \hat{H}(\hat{l})(@proto).1.1.1.1.2 \not\sqsubseteq \bot_{Null} \\ \bot_{Value} & \text{otherwise} \end{cases} \end{aligned}$$

Cycle in prototype chain is detected at implementation level.

$$\underline{\widehat{\mathsf{ProtoBase}}} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}})$$

$$\underline{\widehat{\mathsf{ProtoBase}}}(\hat{H}, \hat{l}, \hat{s}) = \hat{L}_1 \cup \hat{L}_2$$

$$\begin{aligned} \text{where } & \hat{l} \in dom(\hat{H}) \\ & \wedge \hat{L}_1 = \begin{cases} \{\ \hat{l}\ \} & \widehat{\mathsf{true}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l})) \\ \{\} & \text{otherwise} \end{cases} \\ & \wedge \hat{L}_{proto} = \hat{H}(\hat{l})(@proto).1.1.1.2 \\ & \wedge \hat{L}_2 = \begin{cases} \bigsqcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{ProtoBase}}}(\hat{H}, \hat{l}_{proto}, \hat{s}) & \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l})) \\ \{\} & \text{otherwise} \end{cases} \end{aligned}$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{Pruning}}}_1 & : \mathsf{PrunExpr} \times \widehat{\mathsf{Value}} \times \mathsf{IRRelOP} \times \widehat{\mathsf{Value}} \times \widehat{\mathsf{State}} \to \widehat{\mathsf{State}} \\
& \underline{\widehat{\mathsf{Pruning}}}_1(pe, \hat{v}_1, \S, \hat{v}_2, (\hat{H}, \hat{C})) = (\hat{H}_1, \hat{C}_1) \\
& \quad \text{where } (\hat{v}, a\hat{b}s) = \underline{\widehat{\mathsf{K}}}_{\S}(\hat{v}_2) \\
& \qquad \hat{s} = \begin{cases} \text{“}\hat{x}\text{”} & \text{if } pe = x \\ \underline{\widehat{\mathsf{toString}}}(\hat{pv}) & \text{if } pe = e_1[e_2] \\ & \text{where } \hat{pv} = \underline{\widehat{\mathsf{toPrimitive}}}((\hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})).1) \end{cases} \\
& \qquad \hat{L}_{base} = \begin{cases} \underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, \hat{C}.1, \text{“}x\text{”}) & \text{if } pe = x \\ \bigsqcup_{\hat{l} \in (\hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C})).1.2} \underline{\widehat{\mathsf{ProtoBase}}}(\hat{H}, \hat{l}, \hat{s}) & \text{if } pe = e_1[e_2] \end{cases} \\
& \qquad pr\hat{o}pv = \begin{cases} \langle\langle \hat{v} \sqcap \hat{v}_1, \hat{ov}.2, \hat{ov}.3, \hat{ov}.4\rangle, \bot_{Value}, \bot_{FunctionId}\rangle & \text{if } \underline{\widehat{\mathsf{size}}}(\hat{L}_{base}) = 1 \\ & \text{where } \hat{l} \in \hat{L}_{base} \\ & \qquad\qquad \hat{ov} = \hat{H}(\hat{l})(\hat{s}).1.1 \\ \bot_{PropValue} & \text{otherwise} \end{cases} \\
& \qquad (\hat{H}_1, \hat{C}_1) = \begin{cases} (\hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\hat{s} \mapsto \langle pr\hat{o}pv, a\hat{b}s \sqcap \hat{H}(\hat{l})(\hat{s}).2\rangle]], \hat{C}) & \text{if } \underline{\widehat{\mathsf{size}}}(\hat{L}_{base}) = 1 \wedge \{x\} = \gamma(\hat{s}) \\ & \text{where } \hat{l} \in \hat{L}_{base} \\ (\bot_{Heap}, \bot_{Context}) & \text{if } \underline{\widehat{\mathsf{size}}}(\hat{L}_{base}) = 0 \\ (\hat{H}, \hat{C}) & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{Pruning}}}_2 & : \mathsf{RelExpr} \times \widehat{\mathsf{State}} \to \widehat{\mathsf{State}} \\
& \underline{\widehat{\mathsf{Pruning}}}_2(re, (\hat{H}, \hat{C})) = (\hat{H}_1, \hat{C}_1) \\
& \quad \text{where } e_1 \S e_2 = re \\
& \qquad \hat{v}_1 = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C})).1 \\
& \qquad \hat{v}_2 = (\hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})).1 \\
& \qquad \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_1)) \\
& \qquad \hat{L}_{base} = \begin{cases} \bigsqcup_{\hat{l} \in \hat{v}_2.2} \underline{\widehat{\mathsf{ProtoBase}}}(\hat{H}, \hat{l}, \hat{s}) & \text{if } \S = \mathtt{in} \\ \hat{v}_2.2 & \text{otherwise} \end{cases} \\
& \qquad (\hat{H}_1, \hat{C}_1) = \begin{cases} (\hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\hat{s} \mapsto (\hat{H}(\hat{l})(\hat{s})).1]], \hat{C}) & \text{if } \{\hat{l}\} = \hat{L}_{base} \wedge \{x\} = \gamma(\hat{s}) \wedge \S = \mathtt{in} \\ (\underline{\widehat{\mathsf{DeleteAll}}}(\hat{H}, \hat{l}, \hat{s}), \hat{C}) & \text{if } \{\hat{l}\} = \hat{L}_{base} \wedge \{x\} = \gamma(\hat{s}) \wedge \S = \mathtt{notIn} \\ (\underline{\widehat{\mathsf{PrunInstanceof}}}(\hat{l}_1, \hat{l}, \widehat{\mathsf{true}}, \hat{H}), \hat{C}) & \text{if } \{\hat{l}\} = \hat{L}_{base} \wedge \{\hat{l}_1\} = \hat{v}_1.2 \\ & \quad \wedge \S = \mathtt{instanceof} \\ (\underline{\widehat{\mathsf{PrunInstanceof}}}(\hat{l}_1, \hat{l}, \widehat{\mathsf{false}}, \hat{H}), \hat{C}) & \text{if } \{\hat{l}\} = \hat{L}_{base} \wedge \{\hat{l}_1\} = \hat{v}_1.2 \\ & \quad \wedge \S = \mathtt{notInstanceof} \\ (\bot_{Heap}, \bot_{Context}) & \text{if } \underline{\widehat{\mathsf{size}}}(\hat{L}_{base}) = 0 \\ (\hat{H}, \hat{C}) & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{PrunInstanceof}}} & : \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Bool}} \times \widehat{\mathsf{Heap}} \to \widehat{\mathsf{Heap}} \\
& \underline{\widehat{\mathsf{PrunInstanceof}}}(\hat{l}_{obj}, \hat{l}_{fun}, \hat{b}, \hat{H}) = \hat{H}_1 \sqcap \hat{H}_2 \\
& \quad \text{where } \hat{L}_{prototype} = \hat{H}(\hat{l}_{fun})(\text{“}\mathit{prototype}\text{”}).1.1.1.2 \\
& \qquad \hat{L}_{proto} = \hat{H}(\hat{l}_{obj})(@\mathit{proto}).1.1.1.2 \\
& \qquad \hat{L}_1 = \bigsqcup_{\hat{l}_1 \in \hat{L}_{proto}} \bigsqcup_{\hat{l}_2 \in \hat{L}_{prototype}} \underline{\widehat{\mathsf{inheritProto}}}_2(\hat{H}, \hat{l}_1, \hat{l}_2, \hat{b}) \\
& \qquad \hat{L}_2 = \bigsqcup_{\hat{l}_1 \in \hat{L}_{proto}} \bigsqcup_{\hat{l}_2 \in \hat{L}_{prototype}} \underline{\widehat{\mathsf{inheritProto}}}_1(\hat{H}, \hat{l}_1, \hat{l}_2, \hat{b}) \\
& \qquad \hat{H}_1 = \hat{H}\left[\hat{l}_{obj} \mapsto \hat{H}(\hat{l}_{obj})\left[@\mathit{proto} \mapsto \left\langle \begin{array}{l} \langle \bot_{PValue}, \hat{L}_1\rangle, \\ \widehat{\mathsf{false}}, \\ \widehat{\mathsf{false}}, \\ \widehat{\mathsf{false}} \end{array} \right\rangle\right]\right] \\
& \qquad \hat{H}_2 = \hat{H}\left[\hat{l}_{fun} \mapsto \hat{H}(\hat{l}_{fun})\left[\text{“}\mathit{prototype}\text{”} \mapsto \left\langle \begin{array}{l} \langle \bot_{PValue}, \hat{L}_2\rangle, \\ \widehat{\mathsf{false}}, \\ \widehat{\mathsf{false}}, \\ \widehat{\mathsf{false}} \end{array} \right\rangle\right]\right]
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{size}}} & : \wp(\widehat{\mathsf{Loc}}) \to \mathsf{Number} \\
& \begin{array}{lcll} \underline{\widehat{\mathsf{size}}}(\{\}) & = & 0 & \\ \underline{\widehat{\mathsf{size}}}(\hat{L}) & = & 1 + \underline{\widehat{\mathsf{size}}}(\hat{L}_1) & \text{where } \hat{l} \in \hat{L} \\ & & & \qquad \hat{L}_1 = \hat{L} - \{\hat{l}\} \end{array}
\end{array}
$$

$\underline{\widehat{\mathsf{VarStore}}}$

$$
\begin{aligned}
&: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{Heap}}\\
&\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if} \ \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar}\\
&\quad \text{where} \ \ \hat{H}_1 = \hat{H}[\#Pur\hat{e}Local_R \mapsto \hat{H}(\#Pur\hat{e}Local_R)[x \mapsto \langle \hat{v}, \bot_{Bool}, \bot_{Bool}, \widehat{\mathsf{false}}\rangle]]\\
&\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if} \ \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar}\\
&\quad \text{where} \ \ \hat{H}_1 = \textstyle\bigsqcup_{\hat{l} \in \hat{H}(\#Pur\hat{e}Local_R)(@env).1.2.2} \underline{\widehat{\mathsf{VarStoreL}}}(\hat{H}, \hat{l}, x, \hat{v})\\
&\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \quad \text{if} \ \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar}\\
&\quad \text{where} \ \ \hat{H}_1 = \hat{H}[\#Col\hat{l}apsed_O \mapsto \hat{H}(\#Col\hat{l}apsed_O)[x \mapsto \langle \hat{v}, \bot_{Bool}, \bot_{Bool}, \widehat{\mathsf{false}}\rangle]]\\
&\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \sqcup \hat{H}_2 \quad \text{if} \ \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{GlobalVar}\\
&\quad \text{where} \ \ \hat{H}_1 = \begin{cases} \underline{\widehat{\mathsf{VarStoreG}}}(\hat{H}, x, \hat{v}) & \text{if} \ \ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPutVar}}}(\hat{H}, x) \\ \bot_{Heap} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge \ \hat{H}_2 = \begin{cases} \hat{H} & \text{if} \ \ \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{CanPutVar}}}(\hat{H}, x) \\ \bot_{Heap} & \text{otherwise} \end{cases}
\end{aligned}
$$

$\underline{\widehat{\mathsf{VarStoreG}}}$

$$
\begin{aligned}
&: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{Heap}}\\
&\underline{\widehat{\mathsf{VarStoreG}}}(\hat{H}, x, \hat{v}) = \hat{H}_1 \sqcup \hat{H}_2\\
&\text{where} \ \ \hat{l}_g = \#G\hat{l}obal_R \ \wedge \ \hat{ov}_{old} = \hat{H}(\hat{l}_g)(x).1.1\\
&\qquad \hat{H}_1 = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}(\hat{H}, \hat{l}_g, \hat{x}, \hat{v}) & \text{if} \ \ \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}_g))) \\ \bot_{Heap} & \text{otherwise} \end{cases}\\
&\qquad \hat{H}_2 = \begin{cases} \hat{H}[\hat{l}_g \mapsto \hat{H}(\hat{l}_g)[x \mapsto \langle \hat{v}, \hat{ov}_{old}.2, \hat{ov}_{old}.3, \hat{ov}_{old}.4\rangle]] & \text{if} \ \ \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}_g))) \\ \bot_{Heap} & \text{otherwise} \end{cases}
\end{aligned}
$$

Writable is false only for function name variables, which is always determined exactly.
Cycle in scope chain is detected at implementation level.

$\underline{\widehat{\mathsf{VarStoreL}}}$

$$
\begin{aligned}
&: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \mathsf{Prop} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{Heap}}\\
&\underline{\widehat{\mathsf{VarStoreL}}}(\hat{H}, \hat{l}, x, \hat{v}) = \hat{H}_1 \sqcup \hat{H}_2\\
&\quad \text{where} \ \ \hat{H}_1 = \begin{cases} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[x \mapsto \langle \hat{v}, \widehat{\mathsf{true}}, \bot_{Bool}, \widehat{\mathsf{false}}\rangle\right]\right] & \text{if} \ \ \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \ \wedge \ \hat{H}(\hat{l})(x).1.1.2 = \widehat{\mathsf{true}} \\ \hat{H} & \text{if} \ \ \widehat{\mathsf{true}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \ \wedge \ \hat{H}(\hat{l})(x).1.1.2 = \widehat{\mathsf{false}} \\ \bot_{Heap} & \text{otherwise} \end{cases}\\
&\qquad \hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2\\
&\qquad \hat{H}_2 = \begin{cases} \textstyle\bigsqcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{VarStoreL}}}(\hat{H}, \hat{l}_{outer}, x, \hat{v}) & \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Heap} & \text{otherwise} \end{cases}
\end{aligned}
$$

$\underline{\widehat{\mathsf{PropStore}}}$

$$
\begin{aligned}
&: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{Heap}}\\
&\underline{\widehat{\mathsf{PropStore}}}(\hat{H}, \hat{l}, \hat{s}, \hat{v}) = \hat{H}_1 \sqcup \hat{H}_2\\
&\quad \text{where} \ \ \hat{H}_1 = \begin{cases} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[\hat{s} \mapsto \langle \hat{v}, \widehat{\mathsf{true}}, \widehat{\mathsf{true}}, \widehat{\mathsf{true}}\rangle\right]\right] & \text{if} \ \ \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Heap} & \text{otherwise} \end{cases}\\
&\qquad \hat{ov}_{old} = \hat{H}(\hat{l})(\hat{s}).1.1\\
&\qquad \hat{H}_2 = \begin{cases} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[\hat{s} \mapsto \langle \hat{v}, \hat{ov}_{old}.2, \hat{ov}_{old}.3, \hat{ov}_{old}.4\rangle\right]\right] & \text{if} \ \ \widehat{\mathsf{true}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \bot_{Heap} & \text{otherwise} \end{cases}
\end{aligned}
$$

$\underline{\widehat{\mathsf{ReturnStore}}}$

$$
\begin{aligned}
&: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{Heap}}\\
&\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \hat{v}) = \hat{H}[\#Pur\hat{e}Local_R \mapsto \hat{H}(\#Pur\hat{e}Local_R)[@return \mapsto \hat{v}]]
\end{aligned}
$$

$\underline{\widehat{\mathsf{toBoolean}}}$ $\quad : \widehat{\mathsf{Value}} \to \widehat{\mathsf{Bool}}$

$$\underline{\widehat{\mathsf{toBoolean}}}(\hat{v}) = \langle\langle\bot,\bot,\bigsqcup_{n=1\cdots8}\hat{b}_n,\bot,\bot\rangle,\{\}\rangle$$

$$\begin{array}{rll}
\text{where } \hat{b}_1 = \widehat{\mathsf{false}} & \text{if} & \widehat{\mathsf{undefined}} \sqsubseteq \hat{v}.1.1 \\
\hat{b}_2 = \widehat{\mathsf{false}} & \text{if} & \widehat{\mathsf{null}} \sqsubseteq \hat{v}.1.2 \\
\hat{b}_3 = \hat{v}.1.3 & & \\
\hat{b}_4 = \widehat{\mathsf{false}} & \text{if} & \hat{0} \sqsubseteq \hat{v}.1.4 \vee \widehat{\mathsf{NaN}} \sqsubseteq \hat{v}.1.4 \\
\hat{b}_5 = \widehat{\mathsf{true}} & \text{if} & \hat{v}.1.4 \not\sqsubseteq \bot_{number} \wedge \hat{v}.1.4 \neq \hat{0} \wedge \hat{v}.1.4 \neq \widehat{\mathsf{NaN}} \\
\hat{b}_6 = \widehat{\mathsf{false}} & \text{if} & \widehat{\text{``''}} \sqsubseteq \hat{v}.1.5 \\
\hat{b}_7 = \widehat{\mathsf{true}} & \text{if} & \hat{v}.1.5 \not\sqsubseteq \bot_{string} \wedge \hat{v}.1.5 \neq \widehat{\text{``''}} \\
\hat{b}_8 = \widehat{\mathsf{true}} & \text{if} & \hat{v}.2 \not\sqsubseteq \bot_{Loc}
\end{array}$$

$\underline{\widehat{\mathsf{toNumber}}}$ $\quad : \widehat{\mathsf{PValue}} \to \widehat{\mathsf{Number}}$

$$\underline{\widehat{\mathsf{toNumber}}}(\hat{pv}) = \hat{n}_1 \sqcup \hat{n}_2 \sqcup \hat{n}_3 \sqcup \hat{n}_4 \sqcup \hat{n}_5$$

$$\begin{array}{rll}
\text{where } \hat{n}_1 = \widehat{\mathsf{NaN}} & \text{if} & \widehat{\mathsf{undefined}}_{Value} \sqsubseteq \hat{pv} \\
\hat{n}_2 = \hat{0} & \text{if} & \widehat{\mathsf{null}} \sqsubseteq \hat{pv} \vee \widehat{\mathsf{false}} \sqsubseteq \hat{pv} \\
\hat{n}_3 = \hat{1} & \text{if} & \widehat{\mathsf{true}} \sqsubseteq \hat{pv} \\
\hat{n}_4 = \hat{pv}.4 & & \\
\hat{n}_5 = \underline{\widehat{\mathsf{Str2Num}}}(\hat{pv}) & \text{if} & \hat{pv}.5 \not\sqsubseteq \bot_{string}
\end{array}$$

$\underline{\widehat{\mathsf{toString}}}$ $\quad : \widehat{\mathsf{PValue}} \to \widehat{\mathsf{String}}$

$$\underline{\widehat{\mathsf{toString}}}(\hat{pv}) = \hat{s}_1 \sqcup \hat{s}_2 \sqcup \hat{s}_3 \sqcup \hat{s}_4 \sqcup \hat{s}_5$$

$$\begin{array}{rll}
\text{where } \hat{s}_1 = \text{``}\widehat{undefined}\text{''} & \text{if} & \hat{pv}.1 \not\sqsubseteq \bot_{Undefined} \\
\hat{s}_2 = \text{``}\widehat{null}\text{''} & \text{if} & \hat{pv}.2 \not\sqsubseteq \bot_{Null} \\
\hat{s}_3 = \text{``}\widehat{pv.3}\text{''} & \text{if} & \hat{pv}.3 \not\sqsubseteq \bot_{Bool} \\
\hat{s}_4 = \text{``}\widehat{pv.4}\text{''} & \text{if} & \hat{pv}.4 \not\sqsubseteq \bot_{Number} \\
\hat{s}_5 = \hat{pv}.5 & &
\end{array}$$

$\underline{\widehat{\mathsf{toStringSet}}}$ $\quad : \widehat{\mathsf{PValue}} \to \wp(\widehat{\mathsf{String}})$

$$\underline{\widehat{\mathsf{toStringSet}}}(\hat{pv}) = \widehat{ss} \textit{ with redundancies removed}$$

$$\begin{array}{rl}
\text{where } \widehat{ss}_1 = & \begin{cases} \{\text{``}\widehat{undefined}\text{''}\} & \text{if } \hat{pv}.1 \not\sqsubseteq \bot_{Undefined} \\ \{\} & \text{otherwise} \end{cases} \\
\widehat{ss}_2 = & \begin{cases} \{\text{``}\widehat{null}\text{''}\} & \text{if } \hat{pv}.2 \not\sqsubseteq \bot_{Null} \\ \{\} & \text{otherwise} \end{cases} \\
\widehat{ss}_3 = & \begin{cases} \{\text{``}\widehat{pv.3}\text{''}\} & \text{if } \hat{pv}.3 \not\sqsubseteq \bot_{Bool} \\ \{\} & \text{otherwise} \end{cases} \\
\widehat{ss}_4 = & \begin{cases} \{\text{``}\widehat{pv.4}\text{''}\} & \text{if } \hat{pv}.4 \not\sqsubseteq \bot_{Number} \\ \{\} & \text{otherwise} \end{cases} \\
\widehat{ss}_5 = & \begin{cases} \{\hat{pv}.5\} & \text{if } \hat{pv}.5 \not\sqsubseteq \bot_{String} \\ \{\} & \text{otherwise} \end{cases} \\
\widehat{ss} = & \widehat{ss}_1 \cup \widehat{ss}_2 \cup \widehat{ss}_3 \cup \widehat{ss}_4 \cup \widehat{ss}_5
\end{array}$$

$$\underline{\widehat{\mathsf{toObject}}} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Value}} \times \widehat{\mathsf{Address}} \rightarrow \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Value}} \times \wp(\widehat{\mathsf{Exception}})$$

$$\underline{\widehat{\mathsf{toObject}}}(\hat{H}, \hat{C}, \hat{v}, \hat{a}) = (\langle \bot_{PValue}, \hat{L}_3 \rangle, \hat{H}_4, \hat{C}_4, \hat{es})$$

$$\begin{aligned}
\text{where } \; & \hat{L} = \hat{v}.2 \\
& \hat{o}_1 = \begin{cases} \underline{\widehat{\mathsf{NewString}}}(\hat{v}.1.5) & \text{if } \; \hat{v}.1.5 \not\sqsubseteq \bot_{string} \\ \bot_{Obj} & \text{otherwise} \end{cases} \\
& \hat{o}_2 = \begin{cases} \underline{\widehat{\mathsf{NewBoolean}}}(\hat{v}.1.3) & \text{if } \; \hat{v}.1.3 \not\sqsubseteq \bot_{boolean} \\ \bot_{Obj} & \text{otherwise} \end{cases} \\
& \hat{o}_3 = \begin{cases} \underline{\widehat{\mathsf{NewNumber}}}(\hat{v}.1.4) & \text{if } \; \hat{v}.1.4 \not\sqsubseteq \bot_{number} \\ \bot_{Obj} & \text{otherwise} \end{cases} \\
& \hat{es} = \begin{cases} \{\mathsf{Type\hat{E}xception}\} & \text{if } \; \hat{v}.1.1 \not\sqsubseteq \bot_{undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{null} \\ \{\} & \text{otherwise} \end{cases} \\
& \hat{o} = \hat{o}_1 \sqcup \hat{o}_2 \sqcup \hat{o}_3 \\
& (\hat{H}_1, \hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new}) \quad \textit{// Recency Abstraction} \\
& \hat{l}_R = (\hat{a}, Re\hat{c}ent) \quad \textit{// Recency Abstraction} \\
& (\hat{L}_1, \hat{H}_2, \hat{C}_2) = \begin{cases} (\{\hat{l}_R\}, \hat{H}_1[\hat{l}_R \mapsto \hat{o}], \hat{C}_1) & \text{if } \; \hat{o} \not\sqsubseteq \bot_{Obj} \\ (\{\}\,, \bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases} \\
& (\hat{L}_2, \hat{H}_3, \hat{C}_3) = \begin{cases} (\hat{L}, \hat{H}, \hat{C} & \text{if } \; \hat{L} \not\sqsubseteq \{\} \\ (\{\}\,, \bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases} \\
& \hat{L}_3 = \hat{L}_1 \sqcup \hat{L}_2 \wedge \; \hat{H}_4 = \hat{H}_2 \sqcup \hat{H}_3 \wedge \; \hat{C}_4 = \hat{C}_2 \sqcup \hat{C}_3
\end{aligned}$$

$$\underline{\widehat{\mathsf{toPrimitive}}} \quad : \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{PValue}}$$

$$\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}) = \hat{v}.1 \sqcup \underline{\widehat{\mathsf{Obj2Str}}}(\hat{v}.2)$$

For all case of $\hat{s}_n$ if the condition is false, the value of $\hat{s}_n$ is $\bot_{String}$.

$$\underline{\widehat{\mathsf{TypeTag}}} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Value}} \rightarrow \widehat{\mathsf{String}}$$

$$\underline{\widehat{\mathsf{TypeTag}}}(\hat{H}, \hat{v}) = \hat{s}_1 \sqcup \hat{s}_2 \sqcup \hat{s}_3 \sqcup \hat{s}_4 \sqcup \hat{s}_5 \sqcup \hat{s}_6 \sqcup \hat{s}_7$$

$$\text{where} \quad \begin{aligned}
\hat{s}_1 &= \text{“}nu\hat{m}ber\text{”} && \text{if } \; \hat{v}.1.4 \not\sqsubseteq \bot_{number} \\
\hat{s}_2 &= \text{“}boo\hat{l}ean\text{”} && \text{if } \; \hat{v}.1.3 \not\sqsubseteq \bot_{boolean} \\
\hat{s}_3 &= \text{“}str\hat{i}ng\text{”} && \text{if } \; \hat{v}.1.5 \not\sqsubseteq \bot_{string} \\
\hat{s}_4 &= \text{“}ob\hat{j}ect\text{”} && \text{if } \; \hat{v}.2 \not\sqsubseteq \bot_{Loc} \wedge \mathsf{false} \sqsubseteq \textstyle\bigsqcup_{\hat{l} \in \hat{v}.2} \underline{\widehat{\mathsf{IsCallable}}}(\hat{H}, \hat{l}) \\
\hat{s}_5 &= \text{“}fun\hat{c}tion\text{”} && \text{if } \; \hat{v}.2 \not\sqsubseteq \bot_{Loc} \wedge \mathsf{true} \sqsubseteq \textstyle\bigsqcup_{\hat{l} \in \hat{v}.2} \underline{\widehat{\mathsf{IsCallable}}}(\hat{H}, \hat{l}) \\
\hat{s}_6 &= \text{“}ob\hat{j}ect\text{”} && \text{if } \; \hat{v}.1.2 \not\sqsubseteq \bot_{null} \\
\hat{s}_7 &= \text{“}unde\hat{f}ined\text{”} && \text{if } \; \hat{v}.1.1 \not\sqsubseteq \bot_{undef}
\end{aligned}$$

$$\underline{\widehat{\mathsf{validity}}}_1 \quad : \mathsf{Expression} \times \widehat{\mathsf{State}} \rightarrow \mathsf{Boolean}$$

$$\underline{\widehat{\mathsf{validity}}}_1(e, (\hat{H}, \hat{C})) = b$$

$$\begin{aligned}
\text{where } \; & \hat{v} = (\hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})).1 \\
& b = \begin{cases} \mathsf{true} & \text{if } \; \hat{v}.1.1 \sqsubseteq \bot_{Undef} \wedge \hat{v}.1.2 \sqsubseteq \bot_{Null} \wedge (\hat{v}.1.4 \sqsubseteq \mathsf{U\hat{I}nt} \vee \hat{v}.1.4 \sqsubseteq \mathsf{N\hat{U}Int}) \\ & \quad \wedge \hat{v}.1.5 \sqsubseteq \bot_{String} \wedge \hat{v}.2 = \{\} \\ \mathsf{false} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\underline{\widehat{\mathsf{validity}}}_2 \quad : \mathsf{Expression} \times \mathsf{Expression} \times \widehat{\mathsf{State}} \rightarrow \mathsf{Boolean}$$

$$\underline{\widehat{\mathsf{validity}}}_2(e_1, e_2, (\hat{H}, \hat{S})) = \underline{\widehat{\mathsf{validity}}}_1(e_1, \hat{S}) \wedge \underline{\widehat{\mathsf{validity}}}_1(e_2, \hat{S})$$

$$\underline{\widehat{\mathsf{validity}}}_3 \quad : \mathsf{Expression} \times \mathsf{Expression} \times \mathsf{Expression} \times \widehat{\mathsf{State}} \rightarrow \mathsf{Boolean}$$

$$\underline{\widehat{\mathsf{validity}}}_3(e_1, e_2, e_3, (\hat{H}, \hat{S})) = \underline{\widehat{\mathsf{validity}}}_1(e_1, \hat{S}) \wedge \underline{\widehat{\mathsf{validity}}}_1(e_2, \hat{S}) \wedge \underline{\widehat{\mathsf{validity}}}_1(e_3, \hat{S})$$

$$\underline{\widehat{\mathsf{X}}} \quad : \mathsf{RelExpr} \rightarrow \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{State}}$$

$$\underline{\widehat{\mathsf{X}}}[\![re]\!](\hat{H}, \hat{C}) = (\hat{H}_1, \hat{C}_1)$$

$$\begin{aligned}
\text{where } \; & e_1 \S\, e_2 = re \\
& \hat{v}_1 = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C})).1 \\
& \hat{v}_2 = (\hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})).1 \\
& (\hat{H}_1, \hat{C}_1) = \begin{cases} \underline{\widehat{\mathsf{Pruning}}}_1(e_1, \hat{v}_1, \S, \hat{v}_2, (\hat{H}, \hat{C})) & \text{if } \; \S \in \mathsf{IRRelOP} \wedge e_1 \in \mathsf{PrunExpression} \\ \underline{\widehat{\mathsf{Pruning}}}_2(re, (\hat{H}, \hat{C})) & \text{if } \; \S \in \mathsf{IRObjOP} \\ (\hat{H}, \hat{C}) & \text{otherwise} \end{cases}
\end{aligned}$$

9.4 Context-sensitivity

`.../jsaf/analysis/typing/CallContext.scala`

9.4.1 Context-insensitive

$$\begin{array}{rcl}
\widehat{\mathsf{CallContext}} & = & \widehat{\mathsf{Address}}\\
\mathsf{global}\widehat{\mathsf{CallContext}} & = & \#Global\widehat{Callsite}
\end{array}$$

$$\begin{array}{rcl}
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}} & : & \widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})\\
 & & \textit{caller context, callee function, callsite, this}\\
 & & \underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) =\\
 & & \quad \begin{cases} \{\langle \#Global\widehat{Callsite},\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{if } \underline{\mathsf{isUserFunction}}_P(fid)\\ \{\langle \hat{l}.1,\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{otherwise} \end{cases}
\end{array}$$

9.4.2 1-callsite sensitivity

$$\begin{array}{rcl}
\widehat{\mathsf{CallContext}} & = & \widehat{\mathsf{Address}} \times \widehat{\mathsf{Address}}\\
\mathsf{global}\widehat{\mathsf{CallContext}} & = & (\#Global\widehat{Callsite}, \#Global\widehat{Callsite})
\end{array}$$

$$\begin{array}{rcl}
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}} & : & \widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})\\
 & & \textit{caller context, callee function, callsite, this}\\
 & & \underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) =\\
 & & \quad \begin{cases} \{\langle (\hat{l}.1, \#Global\widehat{Callsite}),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{if } \underline{\mathsf{isUserFunction}}_P(fid)\\ \{\langle (\hat{cc}.1, \hat{l}.1),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{otherwise} \end{cases}
\end{array}$$

9.4.3 k-callsite sensitivity

$$\begin{array}{rcl}
\widehat{\mathsf{CallContext}} & = & \widehat{\mathsf{Address}}\ \mathsf{list}\\
\mathsf{global}\widehat{\mathsf{CallContext}} & = & \mathsf{nil}
\end{array}$$

$$\begin{array}{rcl}
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}} & : & \widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})\\
 & & \textit{caller context, callee function, callsite, this}\\
 & & \underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) =\\
 & & \quad \begin{cases} \{\langle (\hat{l}.1 :: \hat{cc})|_k,\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{if } \underline{\mathsf{isUserFunction}}_P(fid)\\ \{\langle (\hat{l}.1 :: \hat{cc})|_{k+1},\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{otherwise} \end{cases}
\end{array}$$

9.4.4 callsite-set sensitivity

$$\begin{array}{rcl}
\widehat{\mathsf{CallContext}} & = & \wp(\widehat{\mathsf{Address}})\\
\mathsf{global}\widehat{\mathsf{CallContext}} & = & \{\}
\end{array}$$

$$\begin{array}{rcl}
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}} & : & \widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})\\
 & & \textit{caller context, callee function, callsite, this}\\
 & & \underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) = \{\langle \hat{cc} \cup \{\hat{l}.1\},\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\}
\end{array}$$

9.4.5 1-object sensitivity

$$
\begin{aligned}
\widehat{\mathsf{CallContext}} &= \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Address}} \\
\mathsf{global}\widehat{\mathsf{CallContext}} &= (\#Gl\hat{o}bal_R, \#Globa\hat{l}Callsite)
\end{aligned}
$$

$\underline{\mathsf{New}\widehat{\mathsf{CallContext}}}$: $\widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})$

caller context, callee function, callsite, this

$$
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) =
\begin{cases}
\bigcup_{\hat{l}_{this} \in \hat{L}} \{\langle(\hat{l}_{this}, \#Globa\hat{l}Callsite),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \{\hat{l}_{this}\}\rangle\} & \text{if } \underline{\mathsf{isUserFunction}}_P(fid) \\
\{\langle(\hat{cc}.1, \hat{l}.1),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{otherwise}
\end{cases}
$$

9.4.6 1-object sensitivity (TAJS style)

$$
\begin{aligned}
\widehat{\mathsf{CallContext}} &= \wp(\widehat{\mathsf{Loc}}) \times \widehat{\mathsf{Address}} \\
\mathsf{global}\widehat{\mathsf{CallContext}} &= (\{\#Gl\hat{o}bal_R\}, \#Globa\hat{l}Callsite)
\end{aligned}
$$

$\underline{\mathsf{New}\widehat{\mathsf{CallContext}}}$: $\widehat{\mathsf{CallContext}} \times \mathsf{FunctionId} \times \widehat{\mathsf{Loc}} \times \wp(\widehat{\mathsf{Loc}}) \to \wp(\widehat{\mathsf{CallContext}} \times \widehat{\mathsf{Obj}})$

caller context, callee function, callsite, this

$$
\underline{\mathsf{New}\widehat{\mathsf{CallContext}}}(\hat{cc}, fid, \hat{l}, \hat{L}) =
\begin{cases}
\{\langle(\hat{L}, \#Globa\hat{l}Callsite),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{if } \underline{\mathsf{isUserFunction}}_P(fid) \\
\{\langle(\hat{cc}.1, \hat{l}.1),\ \underline{\mathsf{New}\widehat{\mathsf{PureLocal}}}(\{\hat{l}\}, \hat{L})\rangle\} & \text{otherwise}
\end{cases}
$$

9.5 Semantics

`.../jsaf/analysis/typing/{Typing, Semantics, Operator, Worklist, Fixpoint}.scala`

$$
\begin{array}{lcl}
\hat{\mathcal{E}} & \in & \widehat{\mathsf{IPEdge}} \rightarrow \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{State}} \\
\hat{\mathcal{C}} & \in & \widehat{\mathsf{ControlPoint}} \rightarrow \mathsf{Command} \rightarrow \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{State}} \times \widehat{\mathsf{State}} \\
\hat{\mathcal{I}} & \in & \widehat{\mathsf{ControlPoint}} \rightarrow \mathsf{Instruction} \rightarrow \widehat{\mathsf{State}} \times \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{State}} \times \widehat{\mathsf{State}} \\
\hat{\mathcal{V}} & \in & \mathsf{Expression} \rightarrow \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{Value}} \times \wp(\widehat{\mathsf{Exception}}) \\
\hat{\mathcal{B}} & \in & \mathsf{Expression} \rightarrow \widehat{\mathsf{State}} \times \widehat{\mathsf{State}} \rightarrow \widehat{\mathsf{State}} \times \widehat{\mathsf{State}}
\end{array}
$$

$$\hat{\mathcal{E}}[\![\hat{cp} \hookrightarrow_{\hat{C},\hat{o}} ((fid, \mathsf{ENTRY}), \hat{cc})]\!](\bot_{Heap}, \hat{C}_1) = \bot_{State}$$

$$
\begin{array}{l}
\hat{\mathcal{E}}[\![\hat{cp} \hookrightarrow_{\hat{C},\hat{o}} ((fid, \mathsf{ENTRY}), \hat{cc})]\!](\hat{H}_1, \hat{C}_1) = (\hat{H}_3, \hat{C}) \\
\quad \text{where} \;\; \hat{o}_{env} = \underline{\widehat{\mathsf{NewDeclEnvRecord}}}(\hat{o}(@scope).1.2) \\
\qquad\qquad \wedge\; \hat{o}_2 = \hat{o} - @scope \\
\qquad\qquad \wedge\; \hat{H}_2 = \hat{H}_1[\#\hat{PureLocal}_R \mapsto \hat{o}_2] \\
\qquad\qquad \wedge\; \hat{H}_3 = \bigsqcup_{\hat{l}_{env} \in \hat{o}_2(@env).1.2.2} \hat{H}_2[\hat{l}_{env} \mapsto \hat{o}_{env}]
\end{array}
$$

$$\hat{\mathcal{E}}[\![((fid, \mathsf{EXIT}), \hat{cc}) \hookrightarrow_{\hat{C},\hat{o}} \hat{cp}]\!](\bot_{Heap}, \hat{C}_1) = \bot_{State}$$

$$
\begin{array}{l}
\hat{\mathcal{E}}[\![((fid, \mathsf{EXIT}), \hat{cc}) \hookrightarrow_{\hat{C},\hat{o}} \hat{cp}]\!](\hat{H}_1, \hat{C}_1) = \begin{cases} (\hat{H}_3, \hat{C}_2) & \text{if } \; \hat{C}_2 \neq \bot_{Context} \\ \bot_{State} & \text{if } \; \hat{C}_2 = \bot_{Context} \end{cases} \\
\quad \text{where} \;\; (\hat{C}_2, \hat{o}_1) = \underline{\widehat{\mathsf{FixOldify}}}(\hat{C}, \hat{o}, \hat{C}_1.3, \hat{C}_1.4) \\
\qquad\qquad \wedge\; \hat{v} = \hat{H}_1(\#\hat{PureLocal})(@return).1.2 \\
\qquad\qquad \wedge\; \hat{H}_2 = \hat{H}_1[\#\hat{PureLocal}_R \mapsto \hat{o}_1] \\
\qquad\qquad \wedge\; \hat{H}_3 = \underline{\widehat{\mathsf{VarStore}}}(\hat{H}_2, \underline{\mathsf{getReturnVar}}_P(\hat{cp}.1), \hat{v})
\end{array}
$$

$$\hat{\mathcal{E}}[\![((fid, \mathsf{EXIT\text{-}EXC}), \hat{cc}) \hookrightarrow_{\hat{C},\hat{o}} \hat{cp}]\!](\bot_{Heap}, \hat{C}_1) = \bot_{State}$$

$$
\begin{array}{l}
\hat{\mathcal{E}}[\![((fid, \mathsf{EXIT\text{-}EXC}), \hat{cc}) \hookrightarrow_{\hat{C},\hat{o}} \hat{cp}]\!](\hat{H}_1, \hat{C}_1) = \begin{cases} (\hat{H}_2, \hat{C}_2) & \text{if } \; \hat{C}_2 \neq \bot_{Context} \\ \bot_{State} & \text{if } \; \hat{C}_2 = \bot_{Context} \end{cases} \\
\quad \text{where} \;\; (\hat{C}_2, \hat{o}_1) = \underline{\widehat{\mathsf{FixOldify}}}(\hat{C}, \hat{o}, \hat{C}_1.3, \hat{C}_1.4) \\
\qquad\qquad \wedge\; \hat{v} = \hat{H}_1(\#\hat{PureLocal})(@exception).1.2 \\
\qquad\qquad \wedge\; \hat{v}_{old} = \hat{o}_1(@exception\_all).1.2 \\
\qquad\qquad \wedge\; \hat{H}_2 = \hat{H}_1 \left[\#\hat{PureLocal}_R \mapsto \hat{o}_1 \left[\begin{array}{l} @exception \mapsto \hat{v}, \\ @exception\_all \mapsto \hat{v} \sqcup \hat{v}_{old} \end{array} \right] \right]
\end{array}
$$

$$\hat{\mathcal{C}}_{\hat{cp}}[\![c]\!](\bot_{Heap}, \bot_{Context}) = (\bot_{State}, \bot_{State})$$

$$\begin{aligned}
&\hat{\mathcal{C}}_{\hat{cp}}[\![\mathsf{entry}]\!](\hat{H}_0, \hat{C}) = \Big((\hat{H}_m, \hat{C}), \bot_{State}\Big)\\
&\quad \text{where } ((fid_{this}, \mathsf{ENTRY}), \hat{cc}) = \hat{cp}\\
&\qquad\quad \wedge\ x_1 \cdots x_n = \underline{\mathsf{getArgVars}}_P(fid_{this})\ \wedge\ x_{n+1} \cdots x_m = \underline{\mathsf{getLocalVars}}_P(fid_{this})\\
&\qquad\quad \wedge\ \hat{L}_{arg} = \hat{H}_0(\#\widehat{PureLocal}_R)(\underline{\mathsf{getArgumentsName}}(fid_{this})).1.1.1.2\\
&\qquad\quad \wedge\ \forall 1 \le i \le n.\ \hat{H}_i = \underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}_{i-1}, x_i, \textstyle\bigsqcup_{\hat{l}_{arg} \in \hat{L}_{arg}} \underline{\widehat{\mathsf{Proto}}}(\hat{H}_{i-1}, \hat{l}_{arg}, \text{“}i \mathbin{\hat{-}} 1\text{”})\\
&\qquad\quad \wedge\ \forall n+1 \le j \le m.\ \hat{H}_j = \underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H}_{j-1}, x_j, \widehat{\mathsf{undefined}}_{Value})
\end{aligned}$$

$$\hat{\mathcal{C}}_{\hat{cp}}[\![\mathsf{exit}]\!](\hat{H}, \hat{C}) = \Big((\hat{H}, \hat{C}), \bot_{State}\Big)$$

$$\hat{\mathcal{C}}_{\hat{cp}}[\![\mathsf{exit\text{-}exc}]\!](\hat{H}, \hat{C}) = \Big((\hat{H}, \hat{C}), \bot_{State}\Big)$$

$$\hat{\mathcal{C}}_{\hat{cp}}[\![i^+]\!](\hat{H}, \hat{C}) = \Big(\hat{\mathcal{I}}_{\hat{cp}}[\![i]\!]\ \Big((\hat{H}, \hat{C}), \bot_{State}\Big)\Big)^+$$

$$\hat{\mathcal{I}}_{\hat{cp}}[\![i]\!]\ \Big((\bot_{Heap}, \hat{C}), \hat{S}\Big) = \Big((\bot_{State}, \hat{S}\Big)$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![x\texttt{:=alloc(}e^?\texttt{)}\ {}_{\hat{a}_{new}}]\!]\ \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_3, \hat{C}_1), \hat{S}_1\Big)\\
&\quad \text{where } \hat{l}_R = (\hat{a}_{new}, \widehat{Recent}) \wedge\ (\hat{H}_1, \hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad\quad \wedge\ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}_1, \hat{C}_1) \quad \textit{// if e is None, } \hat{v} \textit{ is considered as an element of PValue.}\\
&\qquad\quad \wedge\ \hat{L}_p = \hat{v}.2 \wedge\ \hat{L}_v = \begin{cases} \{\ \#\widehat{ObjProto}_R\ \} & \text{if } \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge\ \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \textstyle\bigsqcup_{\hat{l}_p \in \hat{L}_p \cup \hat{L}_v} \underline{\widehat{\mathsf{NewObject}}}(\hat{l}_p)]\\
&\qquad\quad \wedge\ \hat{H}_3 = \underline{\widehat{\mathsf{VarStore}}}(\hat{H}_2, x, \langle \bot_{PValue}, \{\hat{l}_R\}\rangle)\\
&\qquad\quad \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![x\texttt{:=}\mathsf{allocArray}\texttt{(n)}\ {}_{\hat{a}_{new}}]\!]\ \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_3, \hat{C}_1), \hat{S}\Big)\\
&\quad \text{where } \hat{l}_R = (\hat{a}_{new}, \widehat{Recent}) \wedge\ (\hat{H}_1, \hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad\quad \wedge\ \hat{n} = (\hat{\mathcal{V}}[\![\texttt{n}]\!](\hat{H}_1, \hat{C}_1)).1.1.4\\
&\qquad\quad \wedge\ \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \underline{\widehat{\mathsf{NewArrayObject}}}(\hat{n})]\\
&\qquad\quad \wedge\ \hat{H}_3 = \underline{\widehat{\mathsf{VarStore}}}(\hat{H}_2, x, \langle \bot_{PValue}, \{\hat{l}_R\}\rangle)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![x\texttt{:=}\mathsf{allocArg}\texttt{(n)}\ {}_{\hat{a}_{new}}]\!]\ \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_3, \hat{C}_1), \hat{S}\Big)\\
&\quad \text{where } \hat{l}_R = (\hat{a}_{new}, \widehat{Recent}) \wedge\ (\hat{H}_1, \hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad\quad \wedge \hat{n} = (\hat{\mathcal{V}}[\![\texttt{n}]\!](\hat{H}_1, \hat{C}_1)).1.1.4\\
&\qquad\quad \wedge\ \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \underline{\widehat{\mathsf{NewArgObject}}}(\hat{n})]\\
&\qquad\quad \wedge\ \hat{H}_3 = \underline{\widehat{\mathsf{VarStore}}}(\hat{H}_2, x, \langle \bot_{PValue}, \{\hat{l}_R\}\rangle)
\end{aligned}$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x \mathbin{:=} e \rrbracket \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_1, \hat{C}_1), \hat{S}_1\Big)\\
&\quad \text{where } \ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e \rrbracket(\hat{H}, \hat{C})\\
&\qquad\qquad \wedge (\hat{H}_1, \hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{v}), \hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x_1 \mathbin{:=} \mathsf{delete}\,(x_2)\,\rrbracket \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\underline{\widehat{\mathsf{VarStore}}}(\hat{H}_1, x_1, \hat{b}_{Value}), \hat{C}), \hat{S}\Big)\\
&\quad \text{where } \ \hat{L}_{base} = \underline{\widehat{\mathsf{LookupBase}}}(\hat{H}, x_2)\\
&\qquad\qquad \wedge (\hat{H}_1, \hat{b}) = \textstyle\bigsqcup_{\hat{l}_{base} \in \hat{L}_{base}} \underline{\widehat{\mathsf{Delete}}}(\hat{H}, \hat{l}_{base}, \hat{x}_2)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x \mathbin{:=} \mathsf{delete}\,(e)\,\rrbracket \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_1, \hat{C}_1), \hat{S}_1\Big)\\
&\quad \text{where } \ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e \rrbracket(\hat{H}, \hat{C})\\
&\qquad\qquad \wedge (\hat{H}_1, \hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H}, x, \hat{\mathsf{true}}_{Value}), \hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x \mathbin{:=} \mathsf{delete}\,(e_1, e_2)\,\rrbracket \Big((\hat{H}, \hat{C}), \hat{S}\Big) = \Big((\hat{H}_2, \hat{C}_2), \hat{S}_1\Big)\\
&\quad \text{where } \ \hat{L} = (\hat{\mathcal{V}}\llbracket e_1 \rrbracket(\hat{H}, \hat{C})).1.2 \wedge \ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e_2 \rrbracket(\hat{H}, \hat{C})\\
&\qquad\qquad \wedge \hat{ss} = \begin{cases} \underline{\widehat{\mathsf{toStringSet}}}(\underline{\widehat{\mathsf{toPrimivite}}}(\hat{v})) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge (\hat{H}_1, \hat{b}) = \textstyle\bigsqcup_{\hat{l} \in \hat{L}} \bigsqcup_{\hat{s} \in \hat{ss}} \underline{\widehat{\mathsf{Delete}}}(\hat{H}, \hat{l}, \hat{s})\\
&\qquad\qquad \wedge (\hat{H}_2, \hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H}_1, x, \hat{b}_{Value}), \hat{C}) & \text{if } \ \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket e_1\,[\,e_2\,]\,{=}e_3\rrbracket\Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_1,\hat{C}_1),\hat{S}_1\Big)\\
&\quad\text{where } \hat{L} = (\hat{\mathcal{V}}\llbracket e_1\rrbracket(\hat{H},\hat{C})).1.2 \wedge (\hat{s},\hat{es}_s) = (\hat{\mathcal{V}}\llbracket e_2\rrbracket(\hat{H},\hat{C})) \wedge (\hat{v},\hat{es}) = \hat{\mathcal{V}}\llbracket e_3\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge \hat{v}_{newLen} = \underline{\widehat{\mathsf{ToUInt32}}}(\hat{v}) \wedge \hat{v}_{oldLen} = \hat{H}(\hat{l})(\text{“}length\text{”}).1.1.1.1.4\\
&\qquad\quad \wedge \hat{L}_{\mathit{NArr}} = \Big\{\ \hat{l}\ \mid\ \hat{l} \in \hat{L} \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \Big\}\\
&\qquad\quad \wedge \hat{L}_{\mathit{Arr}} = \Big\{\ \hat{l}\ \mid\ \hat{l} \in \hat{L} \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \Big\}\\
&\qquad\quad \wedge \hat{H}_{\mathit{CantPut}} = \begin{cases} \hat{H} & \text{if } \exists\hat{l} \in \hat{L} : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{NArr}} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{\mathit{NArr}}} \underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\hat{s},\hat{v})\\
&\qquad\quad \wedge (\hat{H}_{\mathit{Arr}},\hat{es}_{\mathit{Arr}}) = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{\mathit{Arr}}}(\hat{H}_{\mathit{Arr}_{length}} \sqcup \hat{H}_{\mathit{Arr}_{index}} \sqcup \hat{H}_{\mathit{Arr}_{other}},\hat{es}_1)\\
&\qquad\quad \wedge (\hat{H}_{\mathit{Arr}_{length}},\hat{es}_1) = \begin{cases} (\hat{H}_{\mathit{Arr}_{length1}},\hat{es}_{len}) & \text{if “}\widehat{length}\text{” } \sqsubseteq \hat{s}\\ (\bot_{Heap},\bot_{Exception}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{length1}} = \begin{cases} \hat{H}_{\mathit{Arr}_{length2}} \sqcup \hat{H}_{\mathit{Arr}_{length3}} \sqcup \hat{H}_{\mathit{Arr}_{length4}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\underline{\widehat{\mathsf{ToNumber}}}(\hat{v})\hat{=}\hat{v}_{newLen})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{length2}} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”},\hat{v}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{v}_{oldLen}\hat{\le}\hat{v}_{newLen}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{length3}} = \begin{cases} \hat{H} & \text{if } \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{length4}} = \begin{cases} \bigsqcup_{x=\hat{v}_{oldLen}-1 \text{ to } \hat{v}_{newLen}} \underline{\widehat{\mathsf{Delete}}}(\underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”},\hat{v}),\hat{l},x) & \text{if } \hat{v}_{newLen}\hat{<}\hat{v}_{oldLen}\\ & \quad\wedge\ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”}))\\ \underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”},\hat{v}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{es}_{len} = \begin{cases} \{\widehat{\mathsf{RangeError}}\} & \text{if } \widehat{\mathsf{false}} \sqsubseteq (\underline{\widehat{\mathsf{ToNumber}}}(\hat{v})\hat{=}\underline{\widehat{\mathsf{ToUInt32}}}(\hat{v}))\\ \bot_{Exception} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{index}} = \begin{cases} \hat{H}_{\mathit{Arr}_{index1}} \sqcup \hat{H}_{\mathit{Arr}_{index2}} \sqcup \hat{H}_{\mathit{Arr}_{index3}} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsArrayIndex}}}(\hat{s})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{index1}} = \begin{cases} \hat{H} & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{v}_{oldLen}\hat{\le}\underline{\widehat{\mathsf{ToUInt32}}}(\hat{s})) \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{index2}} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\hat{s},\hat{v}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\underline{\widehat{\mathsf{ToUInt32}}}(\hat{s})\hat{<}\hat{v}_{oldLen})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{index3}} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}(\underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\hat{s},\hat{v}),\hat{l},\text{“}\widehat{length}\text{”},\underline{\widehat{\mathsf{ToUInt32}}}(\hat{s}\hat{+}\hat{1})) & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{v}_{oldLen}\hat{\le}\underline{\widehat{\mathsf{ToUInt32}}}(\hat{s}))\\ & \quad\wedge\ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_{\mathit{Arr}_{other}} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}(\hat{H},\hat{l},\hat{s},\hat{v}) & \text{if } \hat{s} \neq \text{“}\widehat{length}\text{”} \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsArrayIndex}}}(\hat{s})\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = (\hat{H}_{\mathit{CantPut}} \sqcup \hat{H}_{\mathit{NArr}} \sqcup \hat{H}_{\mathit{Arr}},\hat{C})\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_s \sqcup \hat{es} \sqcup \hat{es}_{\mathit{Arr}})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![x_1 \mathrel{:=} \mathsf{function}\,(fid)\,_{\hat{a}_{new1},\hat{a}_{new2}}]\!] \left((\hat{H},\hat{C}), \hat{S} \right) \\
&= \left(\left(\hat{H}_3 \left[\begin{array}{l} \hat{l}_{R1} \mapsto \widehat{\underline{\mathsf{NewFunctionObject}}}(fid, \hat{H}_3(\widehat{\#PureLocal}_R)(@env).1.2, \hat{l}_{R2}, \hat{n}), \\ \hat{l}_{R2} \mapsto \hat{o}_{new} \left[\text{“}constructor\text{”} \mapsto \langle\langle \bot_{PValue}, \{\hat{l}_{R1}\}\rangle, \hat{\mathsf{true}}, \hat{\mathsf{false}}, \hat{\mathsf{true}} \rangle \right] \end{array} \right], \hat{C}_2 \right), \hat{S} \right) \\
&\quad \text{where } \hat{n} = \alpha(|\, \widehat{\underline{\mathsf{getArgVars}}}_P(fid)\,|) \\
&\qquad \wedge \hat{o}_{new} = \widehat{\underline{\mathsf{NewObject}}}(\widehat{\#ObjProto}_R) \\
&\qquad \wedge \hat{l}_{R1} = (\hat{a}_{new1}, \widehat{Recent}) \wedge\ (\hat{H}_1, \hat{C}_1) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new1}) \quad \textit{// Recency Abstraction} \\
&\qquad \wedge \hat{l}_{R2} = (\hat{a}_{new2}, \widehat{Recent}) \wedge\ (\hat{H}_2, \hat{C}_2) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H}_1, \hat{C}_1, \hat{a}_{new2}) \quad \textit{// Recency Abstraction} \\
&\qquad \wedge \hat{H}_3 = \widehat{\underline{\mathsf{VarStore}}}(\hat{H}_2, x_1, \langle \bot_{PValue}, \{\hat{l}_{R1}\}\rangle)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![x_1 \mathrel{:=} \mathsf{function}\ x_2\,(fid)\,_{\hat{a}_{new1},\hat{a}_{new2},\hat{a}_{new3}}]\!] \left((\hat{H},\hat{C}), \hat{S} \right) \\
&= \left(\left(\hat{H}_5 \left[\begin{array}{l} \hat{l}_{R1} \mapsto \widehat{\underline{\mathsf{NewFunctionObject}}}(fid, \{\hat{l}_{R3}\}_{Value}, \hat{l}_{R2}, \hat{n}), \\ \hat{l}_{R2} \mapsto \hat{o}_{new} \left[\text{“}constructor\text{”} \mapsto \langle\langle \bot_{PValue}, \{\hat{l}_{R1}\}\rangle, \hat{\mathsf{true}}, \hat{\mathsf{false}}, \hat{\mathsf{true}} \rangle \right] \end{array} \right], \hat{C}_3 \right), \hat{S} \right) \\
&\quad \text{where } \hat{n} = \alpha(|\, \widehat{\underline{\mathsf{getArgVars}}}_P(fid)\,|) \\
&\qquad \wedge \hat{o}_{new} = \widehat{\underline{\mathsf{NewObject}}}(\widehat{\#ObjProto}_R) \\
&\qquad \wedge \hat{l}_{R1} = (\hat{a}_{new1}, \widehat{Recent}) \wedge\ (\hat{H}_1, \hat{C}_1) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new1}) \quad \textit{// Recency Abstraction} \\
&\qquad \wedge \hat{l}_{R2} = (\hat{a}_{new2}, \widehat{Recent}) \wedge\ (\hat{H}_2, \hat{C}_2) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H}_1, \hat{C}_1, \hat{a}_{new2}) \quad \textit{// Recency Abstraction} \\
&\qquad \wedge \hat{l}_{R3} = (\hat{a}_{new3}, \widehat{Recent}) \wedge\ (\hat{H}_3, \hat{C}_3) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H}_2, \hat{C}_2, \hat{a}_{new3}) \quad \textit{// Recency Abstraction} \\
&\qquad \wedge \hat{o}_{env} = \widehat{\underline{\mathsf{NewDeclEnvRecord}}}(\hat{H}_3(\widehat{\#PureLocal}_R)(@env).1.2) \\
&\qquad \wedge \hat{H}_4 = \hat{H}_3[\hat{l}_{R3} \mapsto \hat{o}_{env}[x_2 \mapsto \langle\langle \bot_{PValue}, \{\hat{l}_{R1}\}\rangle, \hat{\mathsf{false}}, \bot_{Bool}, \hat{\mathsf{false}} \rangle]] \\
&\qquad \wedge \hat{H}_5 = \widehat{\underline{\mathsf{VarStore}}}(\hat{H}_4, x_1, \langle \bot_{PValue}, \{\hat{l}_{R1}\}\rangle)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![\textsf{construct}\,(e_1,e_2,e_3)\,_{\hat{a}_{new}}]\!]\Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_3,\hat{C}_1),\hat{S}_1\Big)\\
&\quad \text{where } \hat{l}_R = (\hat{a}_{new}, \widehat{Recent}) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\textsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge (\hat{v}_1,\hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}_1,\hat{C}_1) \wedge \hat{L}_f = \Big\{\ \hat{l} \mid \hat{l}\in\hat{v}_1.2 \wedge \widehat{\textsf{true}} \sqsubseteq \underline{\widehat{\textsf{HasConstruct}}}(\hat{H}_1,\hat{l})\ \Big\}\\
&\qquad \wedge \hat{L}_{this} = \underline{\widehat{\textsf{getThis}}}(\hat{H}_1,\hat{\mathcal{V}}[\![e_2]\!](\hat{H}_1,\hat{C}_1).1)\\
&\qquad \wedge \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H}_1,\hat{C}_1).1\\
&\qquad \wedge \hat{o}_{old} = \hat{H}_1(\#\widehat{PureLocal}_R)\\
&\qquad \wedge \hat{cc}_{caller} = \hat{cp}.2\\
&\qquad \wedge n_{\textit{after-call}} = \underline{\textsf{getAftercallFromCall}}_P(\hat{cp}.1)\\
&\qquad \wedge \hat{cp}_{\textit{after-call}} = (n_{\textit{after-call}},\hat{cc}_{caller})\\
&\qquad \wedge \hat{cp}_{exc} = (\underline{\textsf{getExcSucc}}_P(n_{\textit{after-call}}),\hat{cc}_{caller})\\
&\qquad \wedge \overset{\textsf{ip}}{\hookrightarrow} := \overset{\textsf{ip}}{\hookrightarrow} \cup \bigcup_{\hat{l}_f\in\hat{L}_f}\bigcup_{fid\in\hat{H}_1(\hat{l}_f)(@construct).1.3}\bigcup_{(\hat{cc}_{new},\hat{o}_{new})\in\underline{\widehat{\textsf{NewCallContext}}}(\hat{cc}_{caller},fid,\hat{l}_R,\hat{L}_{this})}\\
&\qquad\qquad \left\{\begin{array}{l}
\hat{cp}\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_{new},\hat{o}_{new_2}}\ ((fid,\textsf{ENTRY}),\hat{cc}_{new})\\
\quad \text{where } \hat{C}_{new} = \langle\{\},\{\},\{\},\{\}\rangle\\
\qquad\qquad \hat{o}_{new_2} = \hat{o}_{new}\left[\begin{array}{l}\underline{\textsf{getArgumentsName}}(fid) \mapsto \langle\hat{v}_{arg},\widehat{\textsf{true}},\widehat{\textsf{false}},\widehat{\textsf{false}}\rangle,\\ @scope \mapsto \hat{H}_1(\hat{l}_f)(@scope).1\end{array}\right]\\
((fid,\textsf{EXIT}),\hat{cc}_{new})\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_1,\hat{o}_{old}}\ \hat{cp}_{\textit{after-call}},\\
((fid,\textsf{EXIT-EXC}),\hat{cc}_{new})\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_1,\hat{o}_{old}}\ \hat{cp}_{exc}
\end{array}\right\}\\
&\qquad \wedge \hat{H}_2 = \bigsqcup_{\hat{l}\in\hat{v}_{arg}.2}\hat{H}_1\Big[\hat{l}\mapsto\hat{H}_1(\hat{l})\Big[\text{“}callee\text{”}\mapsto\langle\langle\bot_{PValue},\hat{L}_f\rangle,\widehat{\textsf{true}},\widehat{\textsf{false}},\widehat{\textsf{true}}\rangle\Big]\Big]\\
&\qquad \wedge \hat{es}_2 = \{\widehat{\textsf{TypeError}}\}\quad \text{if } \exists\hat{l}\in\hat{v}_1.2 : \widehat{\textsf{false}} \sqsubseteq \underline{\widehat{\textsf{HasConstruct}}}(\hat{H}_1,\hat{l})\\
&\qquad \wedge \hat{es}_3 = \{\widehat{\textsf{TypeError}}\}\quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\
&\qquad \wedge \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3\\
&\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H}_1,\hat{C}_1,\hat{es})\\
&\qquad \wedge \hat{H}_3 = \begin{cases}\hat{H}_2 & \text{if } \hat{L}_f \neq \{\}\\ \bot_{Heap} & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}[\![\textsf{call}\,(e_1,e_2,e_3)\,_{\hat{a}_{new}}]\!]\Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_3,\hat{C}_1),\hat{S}_1\Big)\\
&\quad \text{where } \hat{l}_R = (\hat{a}_{new}, \widehat{Recent}) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\textsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge (\hat{v}_1,\hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}_1,\hat{C}_1) \wedge \hat{L}_f = \Big\{\ \hat{l} \mid \hat{l}\in\hat{v}_1.2 \wedge \widehat{\textsf{true}} \sqsubseteq \underline{\widehat{\textsf{IsCallable}}}(\hat{H}_1,\hat{l})\ \Big\}\\
&\qquad \wedge \hat{L}_{this} = \underline{\widehat{\textsf{getThis}}}(\hat{H}_1,\hat{\mathcal{V}}[\![e_2]\!](\hat{H}_1,\hat{C}_1).1)\\
&\qquad \wedge \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H}_1,\hat{C}_1).1\\
&\qquad \wedge \hat{o}_{old} = \hat{H}_1(\#\widehat{PureLocal}_R)\\
&\qquad \wedge \hat{cc}_{caller} = \hat{cp}.2\\
&\qquad \wedge n_{\textit{after-call}} = \underline{\textsf{getAftercallFromCall}}_P(\hat{cp}.1)\\
&\qquad \wedge \hat{cp}_{\textit{after-call}} = (n_{\textit{after-call}},\hat{cc}_{caller})\\
&\qquad \wedge \hat{cp}_{exc} = (\underline{\textsf{getExcSucc}}_P(n_{\textit{after-call}}),\hat{cc}_{caller})\\
&\qquad \wedge \overset{\textsf{ip}}{\hookrightarrow} := \overset{\textsf{ip}}{\hookrightarrow} \cup \bigcup_{\hat{l}_f\in\hat{L}_f}\bigcup_{fid\in\hat{H}_1(\hat{l}_f)(@function).1.3}\bigcup_{(\hat{cc}_{new},\hat{o}_{new})\in\underline{\widehat{\textsf{NewCallContext}}}(\hat{cc}_{caller},fid,\hat{l}_R,\hat{L}_{this})}\\
&\qquad\qquad \left\{\begin{array}{l}
\hat{cp}\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_{new},\hat{o}_{new_2}}\ ((fid,\textsf{ENTRY}),\hat{cc}_{new})\\
\quad \text{where } \hat{C}_{new} = \langle\{\},\{\},\{\},\{\}\rangle\\
\qquad\qquad \hat{o}_{new_2} = \hat{o}_{new}\left[\begin{array}{l}\underline{\textsf{getArgumentsName}}(fid) \mapsto \langle\hat{v}_{arg},\widehat{\textsf{true}},\widehat{\textsf{false}},\widehat{\textsf{false}}\rangle,\\ @scope \mapsto \hat{H}_1(\hat{l}_f)(@scope).1\end{array}\right]\\
((fid,\textsf{EXIT}),\hat{cc}_{new})\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_1,\hat{o}_{old}}\ \hat{cp}_{\textit{after-call}},\\
((fid,\textsf{EXIT-EXC}),\hat{cc}_{new})\ \overset{\textsf{ip}}{\hookrightarrow}_{\hat{C}_1,\hat{o}_{old}}\ \hat{cp}_{exc}
\end{array}\right\}\\
&\qquad \wedge \hat{H}_2 = \bigsqcup_{\hat{l}\in\hat{v}_{arg}.2}\hat{H}_1\Big[\hat{l}\mapsto\hat{H}_1(\hat{l})\Big[\text{“}callee\text{”}\mapsto\langle\langle\bot_{PValue},\hat{L}_f\rangle,\widehat{\textsf{true}},\widehat{\textsf{false}},\widehat{\textsf{true}}\rangle\Big]\Big]\\
&\qquad \wedge \hat{es}_2 = \{\widehat{\textsf{TypeError}}\}\quad \text{if } \exists\hat{l}\in\hat{v}_1.2 : \widehat{\textsf{false}} \sqsubseteq \underline{\widehat{\textsf{IsCallable}}}(\hat{H}_1,\hat{l})\\
&\qquad \wedge \hat{es}_3 = \{\widehat{\textsf{TypeError}}\}\quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\
&\qquad \wedge \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3\\
&\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H}_1,\hat{C}_1,\hat{es})\\
&\qquad \wedge \hat{H}_3 = \begin{cases}\hat{H}_2 & \text{if } \hat{L}_f \neq \{\}\\ \bot_{Heap} & \text{otherwise}\end{cases}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{\hat{cp}}\llbracket \mathsf{assert}\,(e_1 \otimes e_2)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big(\hat{\mathcal{B}}\llbracket e_1 \otimes e_2\rrbracket(\hat{H},\hat{C}),\hat{S}\Big)$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket \mathsf{catch}\,(x)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_2,\hat{C}),\bot_{State}\Big)\\
&\quad \text{where } \hat{v}_{old} = \hat{H}(\#\widehat{PureLocal}_R)(@exception\_all).1.2\\
&\qquad\quad \wedge \hat{H}_1 = \underline{\widehat{\mathsf{CreateMutableBinding}}}(\hat{H},x,\hat{H}(\#\widehat{PureLocal}_R)(@exception).1.2)\\
&\qquad\quad \wedge \hat{H}_2 = \hat{H}_1[\#\widehat{PureLocal}_R \mapsto \hat{H}_1(\#\widehat{PureLocal}_R)[@exception \mapsto \hat{v}_{old}]]
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket \mathsf{return}\,(e)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_1,\hat{C}_1),\hat{S}_1\Big)\\
&\quad \text{where } (\hat{v},\hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value}\\ (\bot_{Heap},\bot_{Context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket \mathsf{return}\,(\,)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_1,\hat{C}),\hat{S}\Big)\\
&\quad \text{where } \hat{H}_1 = \hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \widehat{\mathsf{undefined}}_{Value}]]
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket \mathsf{throw}\,(e)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big(\bot_{State},\hat{S}_1\Big)\\
&\quad \text{where } (\hat{v},\hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge \hat{v}_{old} = \hat{H}(\#\widehat{PureLocal}_R)(@exception\_all).1.2\\
&\qquad\quad \wedge \hat{H}_1 = \hat{H}\left[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)\left[\begin{array}{l} @return \mapsto \widehat{\mathsf{undefined}}_{Value},\\ @exception \mapsto \hat{v},\\ @exception\_all \mapsto \hat{v} \sqcup \hat{v}_{old}\end{array}\right]\right]\\
&\qquad\quad \wedge (\hat{H}_e,\hat{C}_e) = \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}) \wedge\ \hat{S}_1 = \hat{S} \sqcup (\hat{H}_1 \sqcup \hat{H}_e, \hat{C} \sqcup \hat{C}_e)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x\!:=\!\underline{\diamond\widehat{\mathsf{toObject}}}\,(e)\;{}_{a_{new}}\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_3,\hat{C}_3),\hat{S}_1\Big)\\
&\quad \text{where } (\hat{v},\hat{es}_1) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1,\hat{v}_1,\hat{es}_2) = \underline{\widehat{\mathsf{toObject}}}(\hat{H},\hat{C},\hat{v},\hat{a}_{new})\\
&\qquad\quad \wedge (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H}_1,x,\hat{v}_1),\hat{C}_1) & \text{if } \hat{v}_1 \not\sqsubseteq \bot_{Value}\\ (\bot_{Heap},\bot_{Context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge (\hat{H}_3,\hat{C}_3,\hat{es}_3) = \begin{cases} (\hat{H}_2,\hat{C}_2,\hat{es}_1 \sqcup \hat{es}_2) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value}\\ (\bot_{Heap},\bot_{Context},\hat{es}_1) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_3)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{\hat{cp}}\llbracket x\!:=\!\underline{\diamond\widehat{\mathsf{isObject}}}\,(e)\,\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_1,\hat{C}_1),\hat{S}_1\Big)\\
&\quad \text{where } (\hat{v},\hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H},x,\hat{b}_{Value}),\hat{C}) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value}\\ (\bot_{Heap},\bot_{Context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b}_1 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \hat{v}.2 \not\sqsubseteq \bot_{Loc}\\ \bot_{Bool} & \text{otherwise}\end{cases} \quad \wedge\ \hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \bot_{Bool} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b} = \hat{b}_1 \sqcup \hat{b}_2\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{toNumber}}}\,(e)\,]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\hat{H}_1,\hat{C}_1),\hat{S}_1\right)$$

$$\begin{aligned}
\text{where }\ & (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C}) \\
& \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{VarStore}}}(\hat{H},x,(\underline{\widehat{\mathsf{toNumber}}}(\widehat{pv}))_{Value}),\hat{C}) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{Heap},\bot_{Context}) & \text{otherwise} \end{cases} \\
& \wedge \widehat{pv} = \underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}) \\
& \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\widehat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![x_1\,\texttt{:=}\diamond\underline{\widehat{\mathsf{getBase}}}\,(x_2)\,]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\underline{\widehat{\mathsf{VarStore}}}(\hat{H},x_1,\langle\bot_{PValue},\hat{L}_{base}\rangle),\hat{C}),\hat{S}\right)$$

$$\text{where }\ \hat{L}_{base} = \underline{\widehat{\mathsf{LookupBase}}}(\hat{H},x_2)$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorInit}}}\,(e)\,]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\hat{H},\hat{C}),\hat{S}\right)$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorHasNext}}}\,(e_1,e_2)\,]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\underline{\widehat{\mathsf{VarStore}}}(\hat{H},x,(\top_{Bool})_{Value}),\hat{C}),\hat{S}\right)$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorNext}}}\,(e_1,e_2)\,]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\underline{\widehat{\mathsf{VarStore}}}(\hat{H},x,(\top_{String})_{Value}),\hat{C}),\hat{S}\right)$$

$$\hat{\mathcal{I}}_{\widehat{cp}}[\![\mathsf{noop}]\!]\left((\hat{H},\hat{C}),\hat{S}\right) = \left((\hat{H},\hat{C}),\hat{S}\right)$$

$$\hat{\mathcal{V}}[\![x]\!](\hat{H},\hat{C}) = \underline{\widehat{\mathsf{Lookup}}}(\hat{H},x)$$

$$\hat{\mathcal{V}}[\![e_1 \otimes e_2]\!](\hat{H},\hat{C}) = (\hat{v},\widehat{es})$$

$$\begin{aligned}
\text{where }\ & (\hat{v}_1,\widehat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C}) \wedge (\hat{v}_2,\widehat{es}_2) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C}) \\
& \wedge (\hat{v},\widehat{es}) = \begin{cases} (\bot_{Value},\widehat{es}_1) & \text{if } \hat{v}_1 \sqsubseteq \bot_{Value} \\ (\bot_{Value},\widehat{es}_1 \sqcup \widehat{es}_2) & \text{if } \hat{v}_1 \not\sqsubseteq \bot_{Value} \wedge \hat{v}_2 \sqsubseteq \bot_{Value} \\ \left(\hat{v}_1 \hat{\otimes} \hat{v}_2,\widehat{es}_1 \sqcup \widehat{es}_2\right) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{V}}[\![\ominus e]\!](\hat{H},\hat{C}) = \left(\hat{\ominus}\hat{v},\widehat{es}\right) \quad \text{where }\ (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})$$

$$\hat{\mathcal{V}}[\![e_1\,\texttt{[}\,e_2\,\texttt{]}\,]\!](\hat{H},\hat{C}) = (\hat{v}_1,\widehat{es})$$

$$\begin{aligned}
\text{where }\ & \hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2 \ \wedge\ (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C}) \\
& \wedge \widehat{ss} = \begin{cases} \underline{\widehat{\mathsf{toStringSet}}}(\underline{\widehat{\mathsf{toPrimivite}}}(\hat{v})) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value} \\ \{\} & \text{otherwise} \end{cases} \\
& \wedge \hat{v}_1 = \bigsqcup_{\hat{l}\in\hat{L}} \bigsqcup_{\hat{s}\in\widehat{ss}} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\hat{s})
\end{aligned}$$

$$\hat{\mathcal{V}}[\![\mathrm{n}]\!](\hat{H},\hat{C}) = (\hat{n}_{Value},\bot_{Exception})$$

$$\hat{\mathcal{V}}[\![\text{“s”}]\!](\hat{H},\hat{C}) = (\hat{s}_{Value},\bot_{Exception})$$

$$\hat{\mathcal{V}}[\![\texttt{true}]\!](\hat{H},\hat{C}) = \left(\widehat{\mathsf{true}}_{Value},\bot_{Exception}\right)$$

$$\hat{\mathcal{V}}[\![\texttt{false}]\!](\hat{H},\hat{C}) = \left(\widehat{\mathsf{false}}_{Value},\bot_{Exception}\right)$$

$$\hat{\mathcal{V}}[\![\texttt{null}]\!](\hat{H},\hat{C}) = \left(\widehat{\mathsf{null}}_{Value},\bot_{Exception}\right)$$

$$\hat{\mathcal{V}}[\![\texttt{this}]\!](\hat{H},\hat{C}) = \left(\langle\bot_{PValue},\hat{H}(\#\widehat{PureLocal})(@this).1.2.2\rangle,\bot_{Exception}\right)$$

$$
\begin{aligned}
&\hat{\mathcal{V}}\llbracket e_1\,\texttt{instanceof}\,e_2\rrbracket(\hat{H},\hat{C}) = \Big(\hat{b}_{Value}, \hat{es}\Big)\\
&\quad \text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}\llbracket e_1\rrbracket(\hat{H},\hat{C}) \wedge\ (\hat{v}_2, \hat{es}_2) = \hat{\mathcal{V}}\llbracket e_2\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge \hat{L}_1 = \hat{v}_1.2 \wedge\ \hat{L}_2 = \hat{v}_2.2\\
&\qquad\quad \wedge \hat{L}_3 = \Big\{\ \hat{l} \mid \hat{l} \in \hat{L}_2 \wedge\ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasInstance}}}(\hat{H},\hat{l})\ \Big\}\\
&\qquad\quad \wedge \hat{v}_{proto} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_3} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l}, \text{“}\widehat{prototype}\text{”})\\
&\qquad\quad \wedge \hat{L}_4 = \hat{v}_{proto}.2\\
&\qquad\quad \wedge \hat{L}_5 = \Big\{\ \hat{l} \mid \hat{l} \in \hat{L}_2 \wedge\ \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasInstance}}}(\hat{H},\hat{l})\ \Big\}\\
&\qquad\quad \wedge \hat{b}_1 = \textstyle\bigsqcup_{\hat{l_1}\in\hat{L}_1} \bigsqcup_{\hat{l_2}\in\hat{L}_4} \underline{\widehat{\mathsf{inherit}}}(\hat{H},\hat{l}_1,\hat{l}_2)\\
&\qquad\quad \wedge \hat{b}_2 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \wedge \hat{L}_4 \not\sqsubseteq \{\} \\ \bot_{Bool} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge \hat{es}_3 = \begin{cases} \{\widehat{\mathsf{TypeError}}\} & \text{if } \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \vee \hat{L}_5 \not\sqsubseteq \{\} \vee \hat{v}_{proto}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge \hat{b} = \hat{b}_1 \sqcup \hat{b}_2 \wedge\ \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{V}}\llbracket e_1\,\texttt{in}\,e_2\rrbracket(\hat{H},\hat{C}) = \Big(\hat{b}_{Value}, \hat{es}\Big)\\
&\quad \text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}\llbracket e_1\rrbracket(\hat{H},\hat{C}) \wedge\ (\hat{v}_2, \hat{es}_2) = \hat{\mathcal{V}}\llbracket e_2\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_1))\\
&\qquad\quad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}_2.2} \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s})\\
&\qquad\quad \wedge \hat{es}_3 = \begin{cases} \{\widehat{\mathsf{TypeError}}\} & \text{if } \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{V}}\llbracket \texttt{typeof}\,\mathtt{x}\rrbracket(\hat{H},\hat{C}) = ((\hat{s}_1 \sqcup \hat{s}_2)_{Value}, \{\})\\
&\quad \text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \wedge \hat{s}_1 = \underline{\widehat{\mathsf{TypeTag}}}(\hat{H},\hat{v})\\
&\qquad\quad \wedge \hat{s}_2 = \begin{cases} \text{“}\widehat{undefined}\text{”} & \text{if } \widehat{\mathsf{ReferenceError}} \in \hat{es} \\ \bot_{String} & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{V}}\llbracket \texttt{typeof}\,e\rrbracket(\hat{H},\hat{C}) = \Big((\underline{\widehat{\mathsf{TypeTag}}}(\hat{H},\hat{v}))_{Value}, \hat{es}\Big)\\
&\quad \text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{B}}\llbracket e\rrbracket \Big((\hat{H},\hat{C}),\hat{S}\Big) = \Big((\hat{H}_1,\hat{C}_1),\hat{S}_1\Big)\\
&\quad \text{where } relSet = \begin{cases} \underline{\widehat{\mathsf{getRel}}}(e_1 \S e_2, (\hat{H},\hat{C})) \cup \underline{\widehat{\mathsf{getRel}}}(e_2 \S^t e_1, (\hat{H},\hat{C})) & \text{if } e_1 \S e_2 = e \wedge \S \in \mathsf{IRRelOP} \\ \{e\} & \text{if } e_1 \S e_2 = e \wedge \S \in \mathsf{IRObjOP} \\ \emptyset & \text{otherwise} \end{cases}\\
&\qquad\quad (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H},\hat{C})\\
&\qquad\quad \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
&\qquad\quad (\hat{H}_1,\hat{C}_1) = \begin{cases} \textstyle\bigsqcap_{re\in relSet} \underline{\widehat{\mathsf{X}}}\llbracket re\rrbracket(\hat{H},\hat{C}) & \text{if } relSet \neq \emptyset \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{toBoolean}}}(\hat{v}).1.3 \\ (\hat{H},\hat{C}) & \text{if } relSet = \emptyset \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{toBoolean}}}(\hat{v}).1.3 \\ (\bot_{Heap}, \bot_{Context}) & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\ominus &::= \mathsf{void} \mid \texttt{+} \mid \texttt{-} \mid \texttt{\~{}} \mid \texttt{!}\\
\otimes &::= \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>}\\
&\quad\ \mid \texttt{+} \mid \texttt{-} \mid \star \mid \texttt{/} \mid \texttt{\%} \mid \texttt{==} \mid \texttt{!=} \mid \texttt{===} \mid \texttt{!==} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=}
\end{aligned}
$$

Chapter 10

Sparse Analysis(Incomplete)

10.1 Access Analysis

- New location and property name pairs $\langle \#C\hat{o}ntext, 1\rangle, \langle \#C\hat{o}ntext, 2\rangle$ are introduced to stand for each of $\widehat{\mathsf{Context}}$ values.

10.1.1 Helper functions for definition set

$$\underline{\widehat{\mathsf{CreateMutableBinding}}}_{def} \quad : \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})$$

$$\begin{aligned}
&\underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \#\widehat{PureLocal}_R, x\rangle\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{PureLocalVar}\\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \hat{l}, x\rangle \mid \hat{l} \in \hat{L}\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedVar}\\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \#\widehat{Collapsed}_O, x\rangle\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedCatchVa}\\
&\underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \#\widehat{Global}_R, x\rangle\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{GlobalVar}
\end{aligned}$$

$$\underline{\widehat{\mathsf{Oldify}}}_{def} \quad : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Address}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})$$

$$\begin{aligned}
&\underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } \hat{l}_R = (\hat{a}, \widehat{Recent}) \ \wedge\ \hat{l}_O = (\hat{a}, \widehat{Old})\\
&\qquad LP_1 = \begin{cases} \{\ \langle \hat{l}_O, s\rangle, \langle \hat{l}_R, s\rangle \mid s \in dom(\hat{H}(\hat{l}_R))\ \} & \text{if } \hat{l}_R \in dom(\hat{H})\\ \{\} \quad \text{otherwise} & \end{cases}\\
&\qquad LP_2 = \begin{cases} \{\ \langle \hat{l}_O, s\rangle, \langle \hat{l}_R, s\rangle \mid s \in dom(\hat{H}(\hat{l}_O))\ \} & \text{if } \hat{l}_O \in dom(\hat{H})\\ \{\} \quad \text{otherwise} & \end{cases}\\
&\qquad LP_3 = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \quad \text{if } \hat{l}_R \in \hat{C}.1\\
&\qquad LP_4 = \{\ \langle \#\widehat{Context}, 2\rangle\ \} \quad \text{if } \hat{l}_R \in \hat{C}.2\\
&\qquad LP_5 = \{\ \langle \hat{l}, s\rangle \mid \hat{l} \in dom(\hat{H}),\ s \in dom(\hat{H})(\hat{l}),\ \hat{l}_R \in \hat{H}(l)(s).1.1.1.2 \vee \hat{l}_R \in \hat{H}(s).1.2.2\ \}
\end{aligned}$$

$$\underline{\widehat{\mathsf{NewObject}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewObject}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewArrayObject}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewArrayObject}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, \text{“}length\text{”}, @\widehat{extensible}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewArgObject}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewArgObject}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, \text{“}length\text{”}, @\widehat{extensible}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewFunctionObject}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewFunctionObject}}}_{def}() = \left\{ \begin{array}{l} @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{function}, @\widehat{construct}, @\widehat{scope},\\ @\widehat{default\_UInt}, @\widehat{default\_NUInt}, \text{“}prototype\text{”}, \text{“}length\text{”} \end{array} \right\}$$

$$\underline{\widehat{\mathsf{NewDeclEnvRecord}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewDeclEnvRecord}}}_{def}() = \{\ @\widehat{outer}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewBoolean}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewBoolean}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{primitive}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewNumber}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewNumber}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{primitive}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewDate}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewDate}}}_{def}() = \{\ @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{primitive}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\underline{\widehat{\mathsf{NewString}}}_{def} \quad : \widehat{\mathsf{Value}} \to \wp(\mathsf{Prop})$$

$$\begin{aligned}
&\underline{\widehat{\mathsf{NewString}}}_{def}(\hat{v}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v}_{len} = length(\hat{v}.1.5)\\
&\qquad LP_1 = \{\ @\widehat{class}, @\widehat{proto}, @\widehat{extensible}, @\widehat{primitive}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}\\
&\qquad LP_2 = \{\ \text{“}i\text{”} \mid 0 \leq i \wedge \exists l \in \gamma(\hat{v}_{len}).i < l\ \}
\end{aligned}$$

$$\underline{\widehat{\mathsf{NewPureLocal}}}_{def} \quad : \mathsf{Unit} \to \wp(\mathsf{Prop})$$

$$\underline{\widehat{\mathsf{NewPureLocal}}}_{def}() = \{\ @\widehat{exception}, @\widehat{exception\_all}, @\widehat{return}, @\widehat{default\_UInt}, @\widehat{default\_NUInt}\ \}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{VarStore}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \#Pur\hat{e}Local_R, x\rangle\ \} \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar}\\
& \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{L}, x) = \textstyle\bigcup_{\hat{l} \in \hat{L}} \underline{\widehat{\mathsf{VarStoreL}}}_{def}(\hat{H}, \hat{l}, x) \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar}\\
& \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{L}, x) = \{\ \langle \#Col\hat{l}apsed_O, x\rangle\ \} \quad \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar}\\
& \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{L}, x) = \underline{\widehat{\mathsf{VarStoreG}}}_{def}(\hat{H}, x)
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{VarStoreL}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{VarStoreL}}}_{def}(\hat{H}, \hat{l}, x) = LP_1 \cup LP_2\\
& \quad \text{where } LP_1 = \begin{cases} \{\ \langle \hat{l}, x\rangle\ \} & \text{if } \mathsf{tr\hat{u}e} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}))) \ \wedge\ \hat{H}(\hat{l})(x).1.1.2 = \mathsf{tr\hat{u}e}\\ \{\} & \text{otherwise} \end{cases}\\
& \qquad\quad \hat{L}_{outer} = \hat{H}(\hat{l})(@o\hat{u}ter).1.2.2\\
& \qquad\quad LP_2 = \begin{cases} \bigcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{VarStoreL}}}_{def}(\hat{H}, \hat{l}_{outer}, x) & \text{if } \mathsf{fal\hat{s}e} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l})))\\ \{\} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{VarStoreG}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{VarStoreG}}}_{def}(\hat{H}, x) = LP_1 \cup LP_2\\
& \quad \text{where } \hat{l}_g = \#G\hat{l}obal_R\\
& \qquad\quad LP_1 = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}_g, \alpha(x)) & \text{if } \mathsf{fal\hat{s}e} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}_g)))\\ \{\} & \text{otherwise} \end{cases}\\
& \qquad\quad LP_2 = \begin{cases} \{\ \langle \hat{l}_g, x\rangle\ \} & \text{if } \mathsf{tr\hat{u}e} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}_g)))\\ \{\} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{PropStore}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \hat{s}) = \{\ \langle \hat{l}, \hat{s}\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{ReturnStore}}}_{def} \quad & : \mathsf{Unit} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{ReturnStore}}}_{def}() = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{Delete}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \hat{s}) = LP\\
& \quad \text{where } LP = \{\ \langle \hat{l}, \hat{s}\rangle\ \} \quad \text{if } \Big(\mathsf{tr\hat{u}e} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \wedge \mathsf{tr\hat{u}e} \sqsubseteq \hat{H}(\hat{l})(\hat{s}).1.1.4\Big)\\
& \qquad\qquad\qquad\qquad\qquad\quad \vee (\mathsf{fal\hat{s}e} \sqsubseteq \underline{\widehat{\mathsf{hasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}))
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{toObject}}}_{def} \quad & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Value}} \times \widehat{\mathsf{Address}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{toObject}}}_{def}(\hat{H}, \hat{C}, \hat{v}, \hat{a}) = LP\\
& \quad \text{where } O_1 = \begin{cases} \underline{\widehat{\mathsf{NewString}}}_{def}(\hat{v}.1.5) & \text{if } \hat{v}.1.5 \not\sqsubseteq \bot_{string}\\ \{\} & \text{otherwise} \end{cases}\\
& \qquad\qquad O_2 = \begin{cases} \underline{\widehat{\mathsf{NewBoolean}}}_{def}() & \text{if } \hat{v}.1.3 \not\sqsubseteq \bot_{boolean}\\ \{\} & \text{otherwise} \end{cases}\\
& \qquad\qquad O_3 = \begin{cases} \underline{\widehat{\mathsf{NewNumber}}}_{def}() & \text{if } \hat{v}.1.4 \not\sqsubseteq \bot_{number}\\ \{\} & \text{otherwise} \end{cases}\\
& \qquad\qquad O = O_1 \cup O_2 \cup O_3\\
& \qquad\qquad LP = \begin{cases} \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}) \cup \{\ \langle \hat{l}_R, s\rangle \mid s \in O\ \} & \text{if } O \neq \emptyset\\ \{\} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
\underline{\widehat{\mathsf{RaiseException}}}_{def} \quad & : \wp(\widehat{\mathsf{Exception}})\wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})\\
& \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}) = LP\\
& \quad \text{where } LP = \begin{cases} \left\{ \begin{array}{l} \langle \#Pur\hat{e}Local_R, @exception\_all\rangle,\\ \langle \#Pur\hat{e}Local_R, @exception\rangle \end{array} \right\} & \text{if } \hat{es} \neq \{\}\\ \{\} & \text{otherwise} \end{cases}
\end{aligned}$$

10.1.2 Helper functions for use set

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{Oldify}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Address}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \\
& \quad \text{where } \ \hat{l}_R = (\hat{a}, \widehat{Recent}) \ \wedge \ \hat{l}_O = (\hat{a}, \widehat{Old}) \\
& \qquad LP_1 = \begin{cases} \{ \ \langle \hat{l}_O, s \rangle, \langle \hat{l}_R, s \rangle \mid s \in dom(\hat{H}(\hat{l}_R)) \ \} & \text{if } \ \hat{l}_R \in dom(\hat{H}) \\ \{\} \quad \text{otherwise} & \end{cases} \\
& \qquad LP_2 = \begin{cases} \{ \ \langle \hat{l}_O, s \rangle, \langle \hat{l}_R, s \rangle \mid s \in dom(\hat{H}(\hat{l}_O)) \ \} & \text{if } \ \hat{l}_O \in dom(\hat{H}) \\ \{\} \quad \text{otherwise} & \end{cases} \\
& \qquad LP_3 = \{ \ \langle \#\widehat{Context}, 1 \rangle, \langle \#\widehat{Context}, 2 \rangle \ \} \\
& \qquad LP_4 = \{ \ \langle \hat{l}, s \rangle \mid \hat{l} \in dom(\hat{H}), \ s \in dom(\hat{H})(\hat{l}), \ \hat{l}_R \in \hat{H}(l)(s).1.1.1.2 \vee \hat{l}_R \in \hat{H}(s).1.2.2 \ \}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{VarStore}}}_{use} & : \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{L}, x) = \{\} \quad \text{if } \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar} \\
& \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{L}, x) = \bigcup_{\hat{l} \in \hat{L}} \underline{\widehat{\mathsf{VarStoreL}}}_{use}(\hat{H}, \hat{l}, x) \quad \text{if } \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar} \\
& \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{L}, x) = \{ \ \langle \#\widehat{Collapsed}_O, x \rangle \ \} \quad \text{if } \ \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchVar} \\
& \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{L}, x) = \underline{\widehat{\mathsf{VarStoreG}}}_{use}(\hat{H}, x) \cup \underline{\widehat{\mathsf{CanPutVar}}}_{use}(\hat{H}, x)
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{VarStoreL}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{VarStoreL}}}_{use}(\hat{H}, \hat{l}, x) = \{ \ \langle \hat{l}, x \rangle \ \} \cup LP \\
& \quad \text{where } \ \hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2 \\
& \qquad LP = \begin{cases} \{ \ \langle \hat{l}, @outer \rangle \ \} \cup \bigcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{VarStoreL}}}_{use}(\hat{H}, \hat{l}_{outer}, x) & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq (x \mathrel{\dot\in} dom(\hat{H}(\hat{l}))) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{VarStoreG}}}_{use} & : \widehat{\mathsf{Heap}} \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{VarStoreG}}}_{use}(\hat{H}, x) = \{ \ \langle \hat{l}_g, x \rangle \ \} \cup LP \\
& \quad \text{where } \ \hat{l}_g = \#\widehat{Global}_R \\
& \qquad LP = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H}, \hat{l}_g, \alpha(x)) & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq (x \mathrel{\dot\in} dom(\hat{H}(\hat{l}_g))) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{PropStore}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = \{ \ \langle \hat{l}, \hat{s} \rangle \ \}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{CanPutVar}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{CanPutVar}}}_{use}(\hat{H}, \hat{s}) = \{ \ \langle \#\widehat{Global}_R, x \rangle \ \} \cup LP \\
& \quad \text{where } \ LP = \begin{cases} \underline{\widehat{\mathsf{CanPut}}}_{use}(\hat{H}, \#\widehat{Global}_R, \alpha(x)) & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq (x \mathrel{\dot\in} dom(\hat{H}(\#\widehat{Global}_R))) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{CanPut}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{CanPut}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = \underline{\widehat{\mathsf{CanPutHelp}}}_{use}(\hat{H}, \hat{l}, \hat{s}, \hat{l})
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{CanPutHelp}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \times \widehat{\mathsf{Loc}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{CanPutHelp}}}_{use}(\hat{H}, \hat{l}_1, \hat{s}, \hat{l}_2) = \{ \ \langle \hat{l}_1, \hat{s} \rangle, \langle \hat{l}_1, @proto \rangle \ \} \cup LP_1 \cup LP_2 \\
& \quad \text{where } \ \hat{L}_{proto} = \hat{H}(\hat{l}_1)(@proto).1.1.1.2 \\
& \qquad LP_1 = \begin{cases} \bigcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{CanPutHelp}}}_{use}(\hat{H}, \hat{l}_{proto}, \hat{s}, \hat{l}_2) & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \mathrel{\dot\in} dom(\hat{H}(\hat{l}_1))) \\ \{\} & \text{otherwise} \end{cases} \\
& \qquad LP_2 = \begin{cases} \{ \ \langle \hat{l}_2, \text{“}@extensible\text{”} \rangle \ \} & \text{if } \ \hat{H}(\hat{l}_1)(@proto).1.1.1.1.2 \not\sqsubseteq \bot_{Null} \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{Delete}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{Delete}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = \{ \ \langle \hat{l}, \hat{s} \rangle \ \}
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\mathsf{IsArray}}}_{use} & : \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
& \underline{\widehat{\mathsf{IsArray}}}_{use}(\hat{H}, \hat{l}) = \{ \ \langle \hat{l}, @\widehat{class} \rangle \ \}
\end{array}
$$

$\underline{\widehat{\mathsf{LookupBase}}}_{use}$

$$
\begin{array}{ll}
\multicolumn{2}{l}{: \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})} \\
\underline{\widehat{\mathsf{LookupBase}}}_{use}(\hat{H}, \hat{L}, x) = \{\} & \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{PureLocalVar} \\
\underline{\widehat{\mathsf{LookupBase}}}_{use}(\hat{H}, \hat{L}, x) = \bigcup_{\hat{l} \in \hat{L}} \underline{\widehat{\mathsf{LookupBaseL}}}_{use}(\hat{H}, \hat{l}, x) & \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedVar} \\
\underline{\widehat{\mathsf{LookupBase}}}_{use}(\hat{H}, \hat{L}, x) = \{\} & \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{CapturedCatchV} \\
\underline{\widehat{\mathsf{LookupBase}}}_{use}(\hat{H}, \hat{L}, x) = \underline{\widehat{\mathsf{LookupBaseG}}}_{use}(\hat{H}, x) & \text{if } \underline{\mathsf{getVarKind}}_P(x) = \mathsf{GlobalVar}
\end{array}
$$

$\underline{\widehat{\mathsf{LookupBaseL}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{LookupBaseL}}}_{use}(\hat{H}, \hat{l}, x) = \{\ \langle \hat{l}, x \rangle\ \} \cup LP \\
\quad \text{where } \hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2 \\
\qquad\qquad LP = \begin{cases} \{\ \langle \hat{l}, @outer \rangle\ \} \cup \bigcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{LookupBaseL}}}_{use}(\hat{H}, \hat{l}_{outer}, x) & \text{if } \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{LookupBaseG}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{LookupBaseG}}}_{use}(\hat{H}, x) = \{\ \langle \#G\hat{l}obal_R, x \rangle, \langle \hat{l}, @outer \rangle\ \} \cup LP_1 \cup LP_2 \\
\quad \text{where } \hat{L}_{proto} = \hat{H}(\#G\hat{l}obal_R)(@proto).1.1.1.2 \\
\qquad\qquad LP_1 = \{\ \langle \hat{l}_{proto}, x \rangle \mid \hat{l}_{proto} \in \hat{L}_{proto}\ \} \\
\qquad\qquad LP_2 = \bigcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \begin{cases} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}_{proto}, \alpha(x)) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}_{proto}, \alpha(x)) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{Proto}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = \{\ \langle \hat{l}, \hat{s} \rangle, \langle \hat{l}, @proto \rangle\ \} \cup LP \\
\quad \text{where } \hat{L}_{proto} = \hat{H}(\hat{l})(@proto).1.1.1.2 \\
\qquad\qquad LP = \begin{cases} \bigcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}_{proto}, \hat{s}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq (\hat{s} \dot{\in} dom(\hat{H}(\hat{l}))) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{HasConstruct}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{HasConstruct}}}_{use}(\hat{H}, \hat{l}) = \{\ \langle \hat{l}, @construct \rangle\ \}
\end{array}
$$

$\underline{\widehat{\mathsf{IsCallable}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{IsCallable}}}_{use}(\hat{H}, \hat{l}) = \{\ \langle \hat{l}, @function \rangle\ \}
\end{array}
$$

$\underline{\widehat{\mathsf{getThis}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Value}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{getThis}}}_{use}(\hat{H}, \hat{v}) = LP \\
\quad \text{where } LP = \{\ \langle \hat{l}, @class \rangle \mid \hat{l} \in \hat{v}.2\ \}
\end{array}
$$

$\underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{L}, x) = \{\} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{PureLocalVar} \\
\underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{L}, x) = \{\ \langle \hat{l}, x \rangle \mid \hat{l} \in \hat{L}\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedVar} \\
\underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{L}, x) = \{\ \langle \#Col\hat{l}apsed_O, x \rangle\ \} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedCatchV} \\
\underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{L}, x) = \{\} \quad \text{if } \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{GlobalVar}
\end{array}
$$

$\underline{\widehat{\mathsf{inherit}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{Loc}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{inherit}}}_{use}(\hat{H}, \hat{l}_1, \hat{l}_2) = \{\ \langle \hat{l}_1, @proto \rangle\ \} \cup LP \\
\quad \text{where } LP = \begin{cases} \bigcup_{\hat{l} \in \hat{H}(\hat{l}_1)(@proto).1.1.1.2} \underline{\widehat{\mathsf{inherit}}}_{use}(\hat{H}, \hat{l}, \hat{l}_2) & \text{if } \hat{l}_1 \neq \hat{l}_2 \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{TypeTag}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Value}} \rightarrow \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{TypeTag}}}_{use}(\hat{H}, \hat{v}) = \bigcup_{\hat{l} \in \hat{v}.2} \underline{\widehat{\mathsf{IsCallable}}}_{use}(\hat{H}, \hat{l})
\end{array}
$$

$\underline{\widehat{\mathsf{Lookup}}}_{use}$

$$
\begin{array}{lll}
: \widehat{\mathsf{Heap}} \times \wp(\widehat{\mathsf{Loc}}) \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) & & \\
\underline{\widehat{\mathsf{Lookup}}}_{use}(\hat{H}, \hat{L}, x) = \{\ \langle \#\widehat{PureLocal}_R, x \rangle\ \} & \text{if} & \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{PureLocalVar} \\
\underline{\widehat{\mathsf{Lookup}}}_{use}(\hat{H}, \hat{L}, x) = \bigcup_{\hat{l} \in \hat{L}} \underline{\widehat{\mathsf{LookupL}}}_{use}(\hat{H}, \hat{l}, x) & \text{if} & \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedVar} \\
\underline{\widehat{\mathsf{Lookup}}}_{use}(\hat{H}, \hat{L}, x) = \{\ \langle \#\widehat{Collapsed}_O, x \rangle\ \} & \text{if} & \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{CapturedCatchVar} \\
\underline{\widehat{\mathsf{Lookup}}}_{use}(\hat{H}, \hat{L}, x) = \underline{\widehat{\mathsf{LookupG}}}_{use}(\hat{H}, x) & \text{if} & \underline{\widehat{\mathsf{getVarKind}}}_P(x) = \mathsf{GlobalVar}
\end{array}
$$

$\underline{\widehat{\mathsf{LookupL}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{LookupL}}}_{use}(\hat{H}, \hat{l}, x) = \{\ \langle \hat{l}, x \rangle\ \} \cup LP \\
\quad \text{where } \hat{L}_{outer} = \hat{H}(\hat{l})(@outer).1.2.2 \\
\qquad\quad LP = \begin{cases} \{\ \langle \hat{l}, @outer \rangle\ \} \cup \bigcup_{\hat{l}_{outer} \in \hat{L}_{outer}} \underline{\widehat{\mathsf{LookupL}}}_{use}(\hat{H}, \hat{l}_{outer}, x) & \text{if } \widehat{\mathsf{false}} \sqsubseteq (x \dot{\in} dom(\hat{H}(\hat{l}) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{LookupG}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \mathsf{Prop} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{LookupG}}}_{use}(\hat{H}, x) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{L}_{proto} = \hat{H}(\#\widehat{Global}_R)(@\widehat{proto}).1.1.1.2 \\
\qquad\quad LP_1 = \{\ \langle \#\widehat{Global}, x \rangle, \langle \#\widehat{Global}, @proto \rangle\ \} \\
\qquad\quad LP_2 = \{\ \langle \hat{l}_{proto}, x \rangle \mid \hat{l}_{proto} \in \hat{L}_{proto}\ \} \\
\qquad\quad LP_3 = \bigcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \begin{cases} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}_{proto}, \alpha(x)) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}_{proto}, \alpha(x)) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{toObject}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Context}} \times \widehat{\mathsf{Value}} \times \widehat{\mathsf{Address}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{toObject}}}_{use}(\hat{H}, \hat{C}, \hat{v}, \hat{a}) = LP \\
\quad \text{where } LP = \begin{cases} \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}) & \text{if } \hat{v}.1.5 \not\sqsubseteq \bot_{string} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{boolean} \vee \hat{v}.1.4 \not\sqsubseteq \bot_{number} \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{HasOwnProperty}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{HasOwnProperty}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = \{\ \langle \hat{l}, \hat{s} \rangle\ \}
\end{array}
$$

$\underline{\widehat{\mathsf{RaiseException}}}_{use}$

$$
\begin{array}{l}
: \wp(\widehat{\mathsf{Exception}}) \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es}) = LP \\
\quad \text{where } LP = \begin{cases} \{\ \langle \#\widehat{PureLocal}_R, @exception\_all \rangle\ \} & \text{if } \widehat{es} \neq \{\} \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

$\underline{\widehat{\mathsf{HasProperty}}}_{use}$

$$
\begin{array}{l}
: \widehat{\mathsf{Heap}} \times \widehat{\mathsf{Loc}} \times \widehat{\mathsf{String}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\underline{\widehat{\mathsf{HasProperty}}}_{use}(\hat{H}, \hat{l}, \hat{s}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{L}_{proto} = \hat{H}(\hat{l})(@proto).1.1.1.2 \\
\qquad\quad LP_1 = \underline{\widehat{\mathsf{HasOwnProperty}}}_{use}(\hat{H}, \hat{l}, \hat{s}) \\
\qquad\quad LP_2 = \{\ \langle \hat{l}, @proto \rangle\ \} \\
\qquad\quad LP_3 = \begin{cases} \bigcup_{\hat{l}_{proto} \in \hat{L}_{proto}} \underline{\widehat{\mathsf{HasProperty}}}_{use}(\hat{H}, \hat{l}_{proto}, \hat{s}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H}, \hat{l}, \hat{s}) \\ \{\} & \text{otherwise} \end{cases}
\end{array}
$$

10.1.3 Semantic functions

$$
\begin{aligned}
\hat{\mathcal{C}}_{def} &\in \mathsf{Command} \to \widehat{\mathsf{State}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\hat{\mathcal{C}}_{use} &\in \mathsf{Command} \to \widehat{\mathsf{State}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\hat{\mathcal{I}}_{def} &\in \mathsf{Instruction} \to \widehat{\mathsf{State}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\hat{\mathcal{I}}_{use} &\in \mathsf{Instruction} \to \widehat{\mathsf{State}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop}) \\
\hat{\mathcal{V}}_{use} &\in \mathsf{Expression} \to \widehat{\mathsf{State}} \to \wp(\widehat{\mathsf{Loc}} \times \mathsf{Prop})
\end{aligned}
$$

$$\begin{array}{l}
\hat{\mathcal{E}}_{def}\llbracket \hat{cp} \hookrightarrow_{\hat{C},\hat{o}} ((fid, \mathsf{ENTRY}), \hat{cc})\rrbracket(\hat{H}, \hat{C}_1) = LP_1 \cup LP_2 \\
\quad \text{where } \ \hat{o}_2 = \hat{o} - @scope \\
\qquad\qquad LP_1 = \left\{ \ \langle \widehat{\#PureLocal}_R, x\rangle \mid x \in dom(\hat{o}_2) \ \right\} \\
\qquad\qquad LP_2 = \left\{ \ \langle \hat{l}_{env}, x\rangle \mid \hat{l}_{env} \in \hat{C}.1, \ x \in \underline{\widehat{\mathsf{NewDeclEnvRecord}}}_{def}() \ \right\}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{E}}_{use}\llbracket \hat{cp} \hookrightarrow_{\hat{C},\hat{o}} ((fid, \mathsf{ENTRY}), \hat{cc})\rrbracket(\hat{H}, \hat{C}_1) = LP \\
\quad \text{where } \ \hat{o}_2 = \hat{o} - @scope \\
\qquad\qquad LP = \left\{ \ \langle \hat{l}_{env}, x\rangle \mid \hat{l}_{env} \in \hat{C}.1, \ x \in \underline{\widehat{\mathsf{NewDeclEnvRecord}}}_{def}() \ \right\}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{C}}_{def}\llbracket \mathsf{entry}\rrbracket(\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\quad \text{where } \ ((fid_{this}, \mathsf{ENTRY}), \hat{cc}) = \hat{cp} \\
\qquad\qquad \wedge\, x_1 \cdots x_n = \underline{\mathsf{getArgVars}}_P(fid_{this}) \ \wedge \ x_{n+1} \cdots x_m = \underline{\mathsf{getLocalVars}}_P(fid_{this}) \\
\qquad\qquad LP_1 = \bigcup_{1 \le i \le n} \underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{C}.1, x_i) \\
\qquad\qquad LP_2 = \bigcup_{n+1 \le j \ne m} \underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{C}.1, x_j)
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{C}}_{use}\llbracket \mathsf{entry}\rrbracket(\hat{H}, \hat{C}) = \left\{ \ \langle \#Context, 1\rangle \ \right\} \cup LP_1 \cup LP_2 \\
\quad \text{where } \ ((fid_{this}, \mathsf{ENTRY}), \hat{cc}) = \hat{cp} \\
\qquad\qquad \hat{L}_{arg} = \hat{H}(\widehat{\#PureLocal}_R)(\underline{\mathsf{getArgumentsName}}(fid_{this})).1.1.1.2 \\
\qquad\qquad \wedge\, x_1 \cdots x_n = \underline{\mathsf{getArgVars}}_P(fid_{this}) \ \wedge \ x_{n+1} \cdots x_m = \underline{\mathsf{getLocalVars}}_P(fid_{this}) \\
\qquad\qquad LP_1 = \left\{ \ \langle \widehat{\#PureLocal}_R, \underline{\mathsf{getArgumentsName}}(fid_{this})\rangle \ \right\} \\
\qquad\qquad LP_2 = \bigcup_{\hat{l} \in \hat{L}_{arg}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}i \mathbin{\hat{-}} 1\text{”}) \\
\qquad\qquad LP_3 = \bigcup_{1 \le i \le n} \underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{C}.1, x_i) \\
\qquad\qquad LP_4 = \bigcup_{n+1 \le j \ne m} \underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{C}.1, x_j)
\end{array}$$

$$\hat{\mathcal{C}}_{def}\llbracket \mathsf{exit}\rrbracket(\hat{H}, \hat{C}) = \{\}$$

$$\hat{\mathcal{C}}_{use}\llbracket \mathsf{exit}\rrbracket(\hat{H}, \hat{C}) = \{\}$$

$$\hat{\mathcal{C}}_{def}\llbracket \mathsf{exit\text{-}exc}\rrbracket(\hat{H}, \hat{C}) = \{\}$$

$$\hat{\mathcal{C}}_{use}\llbracket \mathsf{exit\text{-}exc}\rrbracket(\hat{H}, \hat{C}) = \{\}$$

$$\hat{\mathcal{C}}_{def}\llbracket i^{+}\rrbracket(\hat{H}, \hat{C}) = \bigcup_{i \in i^{+}} \hat{\mathcal{I}}_{def}\llbracket i\rrbracket(\hat{H}, \hat{C})$$

$$\hat{\mathcal{C}}_{use}\llbracket i^{+}\rrbracket(\hat{H}, \hat{C}) = \bigcup_{i \in i^{+}} \hat{\mathcal{I}}_{use}\llbracket i\rrbracket(\hat{H}, \hat{C})$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}\llbracket x\texttt{:=alloc(}e^{?}\texttt{)}_{\hat{a}_{new}}\rrbracket(\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \\
\quad \text{where } \ \hat{l}_R = (\hat{a}_{new}, Recent) \\
\qquad\qquad (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H}, \hat{C}) \\
\qquad\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}_{new}) \\
\qquad\qquad LP_2 = \left\{ \ \langle \hat{l}_R, s\rangle \mid s \in \underline{\widehat{\mathsf{NewObject}}}_{def}() \ \right\} \\
\qquad\qquad LP_3 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{C}.1, x) \\
\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}\llbracket x\texttt{:=alloc(}e^{?}\texttt{)}_{\hat{a}_{new}}\rrbracket(\hat{H}, \hat{C}) = \left\{ \ \langle \widehat{\#Context}, 1\rangle \ \right\} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \\
\quad \text{where } \ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}\llbracket e\rrbracket(\hat{H}, \hat{C}) \\
\qquad\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_{new}) \\
\qquad\qquad LP_2 = \hat{\mathcal{V}}_{use}\llbracket e\rrbracket(\hat{H}, \hat{C}) \quad \textit{// if e is None, } LP_2 \textit{ is an empty set.} \\
\qquad\qquad LP_3 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{C}.1, x) \\
\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})
\end{array}$$

$\hat{\mathcal{I}}_{def}[\![x\texttt{:=allocArray(n)}_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$
where $\hat{l}_R = (\hat{a}_{new}, Recent)$
$LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new})$
$LP_2 = \left\{ \ \langle \hat{l}_R, s\rangle \mid s \in \underline{\widehat{\mathsf{NewArrayObject}}}_{def}() \ \right\}$
$LP_3 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)$

$\hat{\mathcal{I}}_{use}[\![x\texttt{:=allocArray(n)}_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = \{ \ \langle \#\widehat{Context}, 1\rangle \ \} \cup LP_1 \cup LP_2$
where $LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new})$
$LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)$

$\hat{\mathcal{I}}_{def}[\![x\texttt{:=allocArg(n)}_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$
where $\hat{l}_R = (\hat{a}_{new}, Recent)$
$LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new})$
$LP_2 = \left\{ \ \langle \hat{l}_R, s\rangle \mid s \in \underline{\widehat{\mathsf{NewArgObject}}}_{def}() \ \right\}$
$LP_3 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)$

$\hat{\mathcal{I}}_{use}[\![x\texttt{:=allocArg(n)}_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = \{ \ \langle \#\widehat{Context}, 1\rangle \ \} \cup LP_1 \cup LP_2$
where $LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new})$
$LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)$

$\hat{\mathcal{I}}_{def}[\![x\texttt{:=}e]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$
where $(\hat{v}, \widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})$
$LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{H}(\#\widehat{PureLocal}_R)(@env).1.2.2, x)$
$LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})$

$\hat{\mathcal{I}}_{use}[\![x\texttt{:=}e]\!](\hat{H},\hat{C}) = \{ \ \langle \#\widehat{PureLocal}_R, @env\rangle \ \} \cup LP_1 \cup LP_2 \cup LP_3$
where $(\hat{v}, \widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})$
$LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{H}(\#\widehat{PureLocal}_R)(@env).1.2.2, x)$
$LP_2 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})$
$LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x_1 \mathbin{:=} \textsf{delete}\,(x_2)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{L}_{base} = \underline{\widehat{\textsf{LookupBase}}}(\hat{H},\hat{C}.1,x_2)\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x_1)\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}_{base}\in\hat{L}_{base}} \underline{\widehat{\textsf{Delete}}}_{def}(\hat{H},\hat{l}_{base},\hat{x}_2)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x_1 \mathbin{:=} \textsf{delete}\,(x_2)\,]\!](\hat{H},\hat{C}) = \{\ \langle \widehat{\#Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{L}_{base} = \underline{\widehat{\textsf{LookupBase}}}(\hat{H},\hat{C}.1,x_2)\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x_1)\\
&\qquad\quad LP_2 = \underline{\widehat{\textsf{LookupBase}}_{use}}(\hat{H},\hat{C}.1,x_2)\\
&\qquad\quad LP_3 = \textstyle\bigcup_{\hat{l}_{base}\in\hat{L}_{base}} \underline{\widehat{\textsf{Delete}}}_{use}(\hat{H},\hat{l}_{base},\hat{x}_2)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x \mathbin{:=} \textsf{delete}\,(e)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } (\hat{v},\hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\quad LP_2 = \underline{\widehat{\textsf{RaiseException}}_{def}}(\hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x \mathbin{:=} \textsf{delete}\,(e)\,]\!](\hat{H},\hat{C}) = \{\ \langle \widehat{\#Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } (\hat{v},\hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\quad LP_2 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_3 = \underline{\widehat{\textsf{RaiseException}}_{use}}(\hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x \mathbin{:=} \textsf{delete}\,(e_1,e_2)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2 \wedge\ (\hat{v},\hat{es}) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C})\\
&\qquad\quad \hat{ss} = \underline{\widehat{\textsf{toStringSet}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}))\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{L}} \bigcup_{\hat{s}\in\hat{ss}} \underline{\widehat{\textsf{Delete}}}_{def}(\hat{H},\hat{l},\hat{s})\\
&\qquad\quad LP_3 = \underline{\widehat{\textsf{RaiseException}}_{def}}(\hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x \mathbin{:=} \textsf{delete}\,(e_1,e_2)\,]\!](\hat{H},\hat{C}) = \{\ \langle \widehat{\#Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } \hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2 \wedge\ (\hat{v},\hat{es}) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C})\\
&\qquad\quad \hat{ss} = \underline{\widehat{\textsf{toStringSet}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}))\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\quad LP_2 = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_3 = \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_4 = \textstyle\bigcup_{\hat{l}\in\hat{L}} \bigcup_{\hat{s}\in\hat{ss}} \underline{\widehat{\textsf{Delete}}}_{use}(\hat{H},\hat{l},\hat{s})\\
&\qquad\quad LP_5 = \underline{\widehat{\textsf{RaiseException}}_{use}}(\hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x \mathbin{:=} \textsf{delete}\,(e_1,e_2)\,]\!](\hat{H},\hat{C}) = \{\ \langle \widehat{\#Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2 \wedge\ \hat{s} = (\hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C})).1.1.5\\
&\qquad\quad LP_1 = \underline{\widehat{\textsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\quad LP_2 = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_3 = \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H},\hat{C})\\
&\qquad\quad LP_4 = \textstyle\bigcup_{\hat{l}\in\hat{L}} \underline{\widehat{\textsf{Delete}}}_{use}(\hat{H},\hat{l},\hat{s})
\end{aligned}
$$

$$\hat{\mathcal{I}}_{def}[\![e_1\,[\,e_2\,]\,{=}e_3]\!](\hat{H},\hat{C}) = LP_1 \cup LP_{ex}$$

where $\hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2$

$$(\hat{v}_{index}, \hat{es}_{index}) = (\hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C}))$$

$$(\hat{v}_{rhs}, \hat{es}_{rhs}) = \hat{\mathcal{V}}[\![e_3]\!](\hat{H},\hat{C})$$

$$LP_{ex} = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1)$$

$$(LP_1, \hat{es}_1) = \begin{cases} (\{\}, \hat{es}_{index}) & \text{if } \hat{v}_{index} \sqsubseteq \bot_{Value} \\ (LP_2, \hat{es}_2) & \text{otherwise} \end{cases}$$

$$(LP_2, \hat{es}_2) = \begin{cases} (\{\}, \hat{es}_{index} \cup \hat{es}_{rhs}) & \text{if } \hat{v}_{rhs} \sqsubseteq \bot_{Value} \\ (LP_3, \hat{es}_3 \cup \hat{es}_{index} \cup \hat{es}_{rhs}) & \text{otherwise} \end{cases}$$

$$\hat{S} = \underline{\widehat{\mathsf{toStringSet}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{index}))$$

$$(LP_3, \hat{es}_3) = \bigcup_{\hat{s}\in\hat{S}}(LP_{NArr} \cup LP_{Arr}, \hat{es}_{Arr})$$

$$\hat{L}_{NArr} = \left\{\ \hat{l} \mid \hat{l} \in \hat{L} \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \right\}$$

$$\hat{L}_{Arr} = \left\{\ \hat{l} \mid \hat{l} \in \hat{L} \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \right\}$$

$$LP_{NArr} = \bigcup_{\hat{l}\in\hat{L}_{NArr}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s})$$

$$(LP_{Arr}, \hat{es}_{Arr}) = \bigcup_{\hat{l}\in\hat{L}}(LP_{length} \cup LP_{index} \cup LP_{normal}, \hat{es}_{len})$$

$$(LP_{length}, \hat{es}_{len}) = \begin{cases} (LP_{length_2}, \hat{es}_{len_2}) & \text{if } \widehat{\text{“}length\text{”}} \sqsubseteq \hat{s} \\ (\{\},\{\}) & \text{otherwise} \end{cases}$$

$$LP_{len_2} = \begin{cases} LP_{len_3} \cup LP_{len_4} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{v}_{value} \hat{=} \hat{v}_{newLen}.1.4 \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{es}_{len_2} = \begin{cases} \{\widehat{\mathsf{RangeError}}\} & \text{if } \widehat{\mathsf{false}} \sqsubseteq \hat{v}_{value} \hat{=} \hat{v}_{newLen}.1.4 \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{len_3} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{v}_{value} \hat{\leq} \hat{v}_{newLen}.1.4 \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{len_4} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) \cup \bigcup_{x=\hat{v}_{oldLen}-1 \text{ to } \hat{v}_{newLen}} \underline{\widehat{\mathsf{Delete}}}(\hat{H},\hat{l},x) & \text{if } \widehat{\mathsf{false}} \sqsubseteq \hat{v}_{value} \hat{=} \hat{v}_{newLen}.1.4 \wedge \mathsf{tr} \\ \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}},\hat{v}) & \text{otherwise} \end{cases}$$

$$LP_{index} = \begin{cases} LP_{index_1} \cup LP_{index_2} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsArrayIndex}}}(\hat{s}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{index_1} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{index} \hat{<} \hat{n}_{oldLen}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{index_2} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}) \cup \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{oldLen} \hat{\leq} \hat{n}_{index}) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l}, \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{normal} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}) & \text{if } \hat{s} \neq \widehat{\text{“}length\text{”}} \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsArrayIndex}}}(\hat{s}) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{\mathcal{I}}_{use}[\![e_1\,[\,e_2\,]\,{=}e_3]\!](\hat{H},\hat{C}) = \left\{\ \langle \#\widehat{Context}, 1\rangle, \langle \#\widehat{Context}, 2\rangle\ \right\} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7$$

where $\hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C})).1.2 \wedge (\hat{v}_{index}, \hat{es}_{index}) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C}) \wedge (\hat{v}_{rhs}, \hat{es}_{rhs}) = \hat{\mathcal{V}}[\![e_3]\!](\hat{H},\hat{C})$

$$\wedge\ \hat{S} = \underline{\widehat{\mathsf{toStringSet}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{index})) \wedge \hat{T} = \left\{\ \hat{l} \mid \hat{l} \in \hat{L} \wedge \exists \hat{s} \in \hat{S} : \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \right\}$$

$$LP_1 = \bigcup_{\hat{l}\in\hat{T}} \bigcup_{\hat{s}\in\hat{S}} \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H},\hat{l},\hat{s})$$

$$LP_2 = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H},\hat{C})$$

$$LP_3 = \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H},\hat{C})$$

$$LP_4 = \hat{\mathcal{V}}_{use}[\![e_3]\!](\hat{H},\hat{C})$$

$$LP_5 = \bigcup_{\hat{l}\in\hat{L}} \bigcup_{\hat{s}\in\hat{S}} \underline{\widehat{\mathsf{CanPut}}}_{use}(\hat{H},\hat{l},\hat{s})$$

$$\hat{n}_{value} = \underline{\widehat{\mathsf{ToNumber}}}(\underline{\widehat{\mathsf{ToPrimitive}}}(\hat{v}_{rhs}))$$

$$\hat{v}_{newLen} = \underline{\widehat{\mathsf{ToUInt32}}}(\hat{v}_{rhs}))$$

$$\hat{es}_{len} = \begin{cases} \{\widehat{\mathsf{RangeError}}\} & \text{if } \widehat{\mathsf{false}} \sqsubseteq \hat{v}_{value} \hat{=} \hat{v}_{newLen}.1.4 \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_6 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_{index} \sqcup \hat{es}_{rhs} \sqcup \hat{es}_{len})$$

$$\hat{L}_{NArr} = \left\{\ \hat{l} \mid \hat{l} \in \hat{L} \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \exists \hat{s} \in \hat{S} : \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \right\}$$

$$LP_{NArr} = \bigcup_{\hat{l}\in\hat{L}_{NArr}} \bigcup_{\hat{s}\in\hat{S}} \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H},\hat{l},\hat{s})$$

$$\hat{L}_{Arr} = \left\{\ \hat{l} \mid \hat{l} \in \hat{L} \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsArray}}}(\hat{H},\hat{l}) \wedge \exists \hat{s} \in \hat{S} : \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{CanPut}}}(\hat{H},\hat{l},\hat{s})\ \right\}$$

$$LP_{Arr} = \bigcup_{\hat{l}\in\hat{L}_{NArr}} \left(\begin{array}{l} \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) \cup \underline{\widehat{\mathsf{CanPut}}}_{use}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}) \\ \cup\ \underline{\widehat{\mathsf{Delete}}}_{use}(\hat{H},\hat{l},SF\widehat{Num}Str) \cup \{\langle \hat{l}, \widehat{\text{“}length\text{”}}\rangle\} \\ \cup\ \bigcup_{\hat{s}\in\hat{S}} \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H},\hat{l},\hat{s}) \end{array} \right)$$

$$LP_7 = LP_{NArr} \cup LP_{Arr}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![x_1 := \mathsf{function}\,(fid)\;_{\hat{a}_{new1},\hat{a}_{new2}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \\
\quad \text{where } \hat{l}_{R1} = (\hat{a}_{new1}, \widehat{Recent}) \;\wedge\; \hat{l}_{R2} = (\hat{a}_{new2}, \widehat{Recent}) \\
\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new1}) \\
\qquad LP_2 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new2}) \\
\qquad LP_3 = \left\{\; \langle \hat{l}_{R1}, s\rangle \mid s \in \underline{\widehat{\mathsf{NewFunctionObject}}}_{def}() \;\right\} \\
\qquad LP_4 = \left\{\; \langle \hat{l}_{R2}, s\rangle \mid s \in \underline{\widehat{\mathsf{NewObject}}}_{def}() \;\right\} \\
\qquad LP_5 = \left\{\; \langle \hat{l}_{R2}, \text{“}constructor\text{”}\rangle \;\right\} \\
\qquad LP_6 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x_1)
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![x_1 := \mathsf{function}\,(fid)\;_{\hat{a}_{new1},\hat{a}_{new2}}]\!](\hat{H},\hat{C}) = \left\{\; \langle \#\widehat{Context}, 1\rangle \;\right\} \cup LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new1}) \\
\qquad LP_2 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new2}) \\
\qquad LP_3 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x_1)
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![x_1 := \mathsf{function}\; x_2\,(fid)\;_{\hat{a}_{new1},\hat{a}_{new2},\hat{a}_{new3}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7 \cup LP_8 \cup LP_9 \\
\quad \text{where } \hat{l}_{R1} = (\hat{a}_{new1}, \widehat{Recent}) \;\wedge\; \hat{l}_{R2} = (\hat{a}_{new2}, \widehat{Recent}) \\
\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new1}) \\
\qquad LP_2 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new2}) \\
\qquad LP_3 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new3}) \\
\qquad LP_4 = \left\{\; \langle \hat{l}_{R1}, s\rangle \mid s \in \underline{\widehat{\mathsf{NewFunctionObject}}}_{def}() \;\right\} \\
\qquad LP_5 = \left\{\; \langle \hat{l}_{R2}, s\rangle \mid s \in \underline{\widehat{\mathsf{NewObject}}}_{def}() \;\right\} \\
\qquad LP_6 = \left\{\; \langle \hat{l}_{R2}, \text{“}constructor\text{”}\rangle \;\right\} \\
\qquad LP_7 = \left\{\; \langle \hat{l}_{R3}, s\rangle \mid s \in \underline{\widehat{\mathsf{NewDeclEnvRecord}}}_{def}() \;\right\} \\
\qquad LP_8 = \left\{\; \langle \hat{l}_{R3}, x_2\rangle \;\right\} \\
\qquad LP_9 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x_1)
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![x_1 := \mathsf{function}\; x_2\,(fid)\;_{\hat{a}_{new1},\hat{a}_{new2},\hat{a}_{new3}}]\!](\hat{H},\hat{C}) = \left\{\; \langle \#\widehat{Context}, 1\rangle \;\right\} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \\
\quad \text{where } LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new1}) \\
\qquad LP_2 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new2}) \\
\qquad LP_3 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new3}) \\
\qquad LP_4 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x_1)
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{construct}\,(e_1,e_2,e_3)\;_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } (\hat{v}_1, \widehat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C}) \\
\qquad \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H},\hat{C}).1 \\
\qquad \widehat{es}_2 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \exists \hat{l} \in \hat{v}_1.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasConstruct}}}(\hat{H},\hat{l}) \\
\qquad \widehat{es}_3 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\
\qquad \widehat{es} = \widehat{es}_1 \sqcup \widehat{es}_2 \sqcup \widehat{es}_3 \\
\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_{new}) \\
\qquad LP_2 = \left\{\; \langle \hat{l}, \text{“}callee\text{”}\rangle \mid \hat{l} \in \hat{v}_{arg}.2 \;\right\} \\
\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{construct}\,(e_1,e_2,e_3)\;_{\hat{a}_{new}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7 \cup LP_8 \cup LP_9 \cup LP_{10} \\
\quad \text{where } (\hat{v}_1, \widehat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H},\hat{C}) \\
\qquad \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H},\hat{C}).1 \\
\qquad \hat{L}_f = \left\{\; \hat{l} \mid \hat{l} \in \hat{v}_1.2 \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasConstruct}}}(\hat{H}_1,\hat{l}) \;\right\} \\
\qquad \widehat{es}_2 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \exists \hat{l} \in \hat{v}_1.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasConstruct}}}(\hat{H},\hat{l}) \\
\qquad \widehat{es}_3 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\
\qquad \widehat{es} = \widehat{es}_1 \sqcup \widehat{es}_2 \sqcup \widehat{es}_3 \\
\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_{new}) \\
\qquad LP_2 = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H},\hat{C}) \\
\qquad LP_3 = \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H},\hat{C}) \\
\qquad LP_4 = \hat{\mathcal{V}}_{use}[\![e_3]\!](\hat{H},\hat{C}) \\
\qquad LP_5 = \bigcup_{\hat{l}\in\hat{v}_1.2} \underline{\widehat{\mathsf{HasConstruct}}}_{use}(\hat{H},\hat{l}) \\
\qquad LP_6 = \underline{\widehat{\mathsf{getThis}}}_{use}(\hat{H},\hat{\mathcal{V}}[\![e_2]\!](\hat{H},\hat{C}).1) \\
\qquad LP_7 = \left\{\; \langle \hat{l}_f, @construct\rangle \mid \hat{l}_f \in \hat{L}_f \;\right\} \\
\qquad LP_8 = \left\{\; \langle \hat{l}, \text{“}callee\text{”}\rangle \mid \hat{l} \in \hat{v}_{arg}.2 \;\right\} \\
\qquad LP_9 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es}) \\
\qquad LP_{10} = \left\{\; \langle \#\widehat{PureLocal}_R, x\rangle \mid \widehat{\mathsf{true}} \sqsubseteq x\,\dot{\in}\, dom(\hat{H}(\#\widehat{PureLocal}_R)) \;\right\}
\end{array}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{call}\,(e_1, e_2, e_3)\,_{\hat{a}_{new}}]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}_1, \hat{C}_1)\\
&\qquad\quad \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H}, \hat{C}).1\\
&\qquad\quad \hat{es}_2 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \exists \hat{l} \in \hat{v}_1.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H}, \hat{l})\\
&\qquad\quad \hat{es}_3 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\
&\qquad\quad \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}_{new})\\
&\qquad\quad LP_2 = \{\ \langle \hat{l}, \text{“}callee\text{”}\rangle \mid \hat{l} \in \hat{v}_{arg}.2\ \}\\
&\qquad\quad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{call}\,(e_1, e_2, e_3)\,_{\hat{a}_{new}}]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7 \cup LP_8 \cup LP_9 \cup LP_{10}\\
&\quad\text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}_1, \hat{C}_1)\\
&\qquad\quad \hat{v}_{arg} = \hat{\mathcal{V}}[\![e_3]\!](\hat{H}, \hat{C}).1\\
&\qquad\quad \hat{L}_f = \left\{\ \hat{l} \mid \hat{l} \in \hat{v}_1.2 \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H}_1, \hat{l})\ \right\}\\
&\qquad\quad \hat{es}_2 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \exists \hat{l} \in \hat{v}_1.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H}, \hat{l})\\
&\qquad\quad \hat{es}_3 = \{\widehat{\mathsf{TypeError}}\} \quad \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\
&\qquad\quad \hat{es} = \hat{es}_1 \sqcup \hat{es}_2 \sqcup \hat{es}_3\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_{new})\\
&\qquad\quad LP_2 = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_3 = \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_4 = \hat{\mathcal{V}}_{use}[\![e_3]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_5 = \textstyle\bigcup_{\hat{l} \in \hat{v}_1.2} \underline{\widehat{\mathsf{IsCallable}}}_{use}(\hat{H}, \hat{l})\\
&\qquad\quad LP_6 = \underline{\widehat{\mathsf{getThis}}}_{use}(\hat{H}, \hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C}).1)\\
&\qquad\quad LP_7 = \{\ \langle \hat{l}_f, @function\rangle \mid \hat{l}_f \in \hat{L}_f\ \}\\
&\qquad\quad LP_8 = \{\ \langle \hat{l}, \text{“}callee\text{”}\rangle \mid \hat{l} \in \hat{v}_{arg}.2\ \}\\
&\qquad\quad LP_9 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\quad LP_{10} = \{\ \langle \#\widehat{PureLocal}_R, x\rangle \mid \widehat{\mathsf{true}} \sqsubseteq x \dot{\in} dom(\hat{H}(\#\widehat{PureLocal}_R))\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{assert}\,(e_1 \otimes e_2)\,]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C})$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{assert}\,(e_1 \otimes e_2)\,]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C})$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{catch}\,(x)\,]\!](\hat{H}, \hat{C}) = \{\ \langle \#\widehat{PureLocal}_R, @exception\rangle\ \} \cup LP\\
&\quad\text{where } LP = \underline{\widehat{\mathsf{CreateMutableBinding}}}_{def}(\hat{H}, \hat{C}.1, x)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{catch}\,(x)\,]\!](\hat{H}, \hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2\\
&\quad\text{where } LP_1 = \underline{\widehat{\mathsf{CreateMutableBinding}}}_{use}(\hat{H}, \hat{C}.1, x)\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @exception\_all\rangle, \langle \#\widehat{PureLocal}_R, @exception\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{return}\,(e)\,]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}\\
&\qquad\quad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{return}\,(e)\,]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{return}\,()\,]\!](\hat{H}, \hat{C}) = LP\\
&\quad\text{where } LP = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{return}\,()\,]\!](\hat{H}, \hat{C}) = \{\}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{throw}\,(e)\,]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @exception\rangle, \langle \#\widehat{PureLocal}_R, @exception\_all\rangle\ \}\\
&\qquad\quad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{throw}\,(e)\,]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad\text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @exception\_all\rangle\ \}\\
&\qquad\quad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toObject}}}\,(e)\,{}_{a_{new}}]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{toObject}}}_{def}(\hat{H},\hat{C},\hat{v},\hat{a}_{new})\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toObject}}}\,(e)\,{}_{a_{new}}]\!](\hat{H},\hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{toObject}}}_{use}(\hat{H},\hat{C},\hat{v},\hat{a}_{new})\\
&\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{isObject}}}\,(e)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{isObject}}}\,(e)\,]\!](\hat{H},\hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toString}}}\,(e)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toString}}}\,(e)\,]\!](\hat{H},\hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toNumber}}}\,(e)\,]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad\text{where } LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{toNumber}}}\,(e)\,]\!](\hat{H},\hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad\text{where } (\hat{v},\widehat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_1 = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H},\hat{C})\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x)\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x_1\,{:}{=}\underline{\diamond\widehat{\mathsf{getBase}}}\,(x_2)\,]\!](\hat{H},\hat{C}) = LP\\
&\quad\text{where } LP = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H},\hat{C}.1,x_1)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x_1\,{:}{=}\underline{\diamond\widehat{\mathsf{getBase}}}\,(x_2)\,]\!](\hat{H},\hat{C}) = \{\ \langle \#\widehat{Context}, 1\rangle\ \} \cup LP_1 \cup LP_2\\
&\quad\text{where } LP_1 = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H},\hat{C}.1,x_1)\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{LookupBase}}}_{use}(\hat{H},\hat{C}.1,x_2)
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{iteratorInit}}}\,(e)\,]\!](\hat{H},\hat{C}) = \{\}$$

$$\hat{\mathcal{I}}_{use}[\![x\,{:}{=}\underline{\diamond\widehat{\mathsf{iteratorInit}}}\,(e)\,]\!](\hat{H},\hat{C}) = \{\}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorHasNext}}}\,(e_1, e_2)\,]\!](\hat{H}, \hat{C}) = LP\\
&\quad \text{where } LP = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{C}.1, x)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorHasNext}}}\,(e_1, e_2)\,]\!](\hat{H}, \hat{C}) = \left\{\ \langle \#\widehat{Context}, 1\rangle\ \right\} \cup LP\\
&\quad \text{where } LP = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{C}.1, x)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorNext}}}\,(e_1, e_2)\,]\!](\hat{H}, \hat{C}) = LP\\
&\quad \text{where } LP = \underline{\widehat{\mathsf{VarStore}}}_{def}(\hat{H}, \hat{C}.1, x)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![x\,\texttt{:=}\diamond\underline{\widehat{\mathsf{iteratorNext}}}\,(e_1, e_2)\,]\!](\hat{H}, \hat{C}) = \left\{\ \langle \#\widehat{Context}, 1\rangle\ \right\} \cup LP\\
&\quad \text{where } LP = \underline{\widehat{\mathsf{VarStore}}}_{use}(\hat{H}, \hat{C}.1, x)
\end{aligned}$$

$$\hat{\mathcal{V}}_{use}[\![x]\!](\hat{H}, \hat{C}) = \left\{\ \langle \#\widehat{Context}, 1\rangle\ \right\} \cup \underline{\widehat{\mathsf{Lookup}}}_{use}(\hat{H}, \hat{C}.1, x)$$

$$\hat{\mathcal{V}}_{use}[\![e_1 \otimes e_2]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C})$$

$$\hat{\mathcal{V}}_{use}[\![\ominus e]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H}, \hat{C})$$

$$\begin{aligned}
&\hat{\mathcal{V}}_{use}[\![e_1\,[\,e_2\,]\,]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C}) \cup LP\\
&\quad \text{where } \hat{L} = (\hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C})).1.2 \ \wedge\ (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})\\
&\qquad\quad\ \ \hat{ss} = \underline{\widehat{\mathsf{toStringSet}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad\quad\ \ LP = \textstyle\bigcup_{\hat{l} \in \hat{L}} \bigcup_{\hat{s} \in \hat{ss}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \hat{s})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{V}}_{use}[\![e_1\ \texttt{instanceof}\ e_2]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C}) \cup LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C}) \wedge\ (\hat{v}_2, \hat{es}_2) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})\\
&\qquad\quad\ \ \hat{L}_1 = \hat{v}_1.2 \wedge\ \hat{L}_2 = \hat{v}_2.2\\
&\qquad\quad\ \ \hat{L}_3 = \left\{\ \hat{l}\ \mid\ \hat{l} \in \hat{L}_2 \wedge\ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasConstruct}}}(\hat{H}, \hat{l})\ \right\}\\
&\qquad\quad\ \ \hat{L}_4 = \hat{v}_{proto}.2\\
&\qquad\quad\ \ \hat{v}_{proto} = \textstyle\bigsqcup_{\hat{l} \in \hat{L}_3} \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \widehat{\text{“}prototype\text{”}})\\
&\qquad\quad\ \ LP_1 = \textstyle\bigcup_{\hat{l} \in \hat{L}_2} \left\{\ \hat{l}, @construct\ \right\}\\
&\qquad\quad\ \ LP_2 = \textstyle\bigcup_{\hat{l} \in \hat{L}_3} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \widehat{\text{“}prototype\text{”}})\\
&\qquad\quad\ \ LP_3 = \textstyle\bigcup_{\hat{l}_1 \in \hat{L}_1} \bigcup_{\hat{l}_2 \in \hat{L}_4} \underline{\widehat{\mathsf{inherit}}}_{use}(\hat{H}, \hat{l}_1, \hat{l}_2)
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{V}}_{use}[\![e_1\ \texttt{in}\ e_2]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e_1]\!](\hat{H}, \hat{C}) \cup \hat{\mathcal{V}}_{use}[\![e_2]\!](\hat{H}, \hat{C}) \cup LP\\
&\quad \text{where } (\hat{v}_1, \hat{es}_1) = \hat{\mathcal{V}}[\![e_1]\!](\hat{H}, \hat{C})\\
&\qquad\quad\ \ (\hat{v}_2, \hat{es}_2) = \hat{\mathcal{V}}[\![e_2]\!](\hat{H}, \hat{C})\\
&\qquad\quad\ \ \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_1))\\
&\qquad\quad\ \ LP = \textstyle\bigcup_{\hat{l} \in \hat{v}_2.2} \underline{\widehat{\mathsf{HasProperty}}}_{use}(\hat{H}, \hat{l}, \hat{s})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{V}}_{use}[\![\texttt{typeof}\ e]\!](\hat{H}, \hat{C}) = \hat{\mathcal{V}}_{use}[\![e]\!](\hat{H}, \hat{C}) \cup LP\\
&\quad \text{where } (\hat{v}, \hat{es}) = \hat{\mathcal{V}}[\![e]\!](\hat{H}, \hat{C})\\
&\qquad\quad\ \ LP = \underline{\widehat{\mathsf{TypeTag}}}_{u\,se}(\hat{H}, \hat{v})
\end{aligned}$$

Chapter 11

Built-in Objects

11.1 Concrete Semantics

11.1.1 Helper Functions

$$\underline{\mathsf{getMatcher}} \quad : \mathsf{MatcherId} \rightarrow (\mathsf{String} \times \mathsf{Int} \rightarrow \mathsf{MatchResult})$$

$$\underline{\mathsf{NewRegExp}} \quad : \mathsf{Value} \times \mathsf{Bool} \times \mathsf{Bool} \times \mathsf{Bool} \times \mathsf{MatcherId} \rightarrow \mathsf{Obj}$$

$$\underline{\mathsf{NewRegExp}}(v_{source}, b_g, b_i, b_m, mid) = \left\{ \begin{array}{l} @class \mapsto \text{“}RegExp\text{”}, \\ @proto \mapsto \left\{ \begin{array}{l} value = \#RegExpProto; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ @matcher \mapsto mid, \\ \text{“}source\text{”} \mapsto \left\{ \begin{array}{l} value = v_{source}; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ \text{“}global\text{”} \mapsto \left\{ \begin{array}{l} value = b_g; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ \text{“}ignoreCase\text{”} \mapsto \left\{ \begin{array}{l} value = b_i; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ \text{“}multiline\text{”} \mapsto \left\{ \begin{array}{l} value = b_m; \\ writable = \mathsf{false}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\}, \\ \text{“}lastIndex\text{”} \mapsto \left\{ \begin{array}{l} value = 0; \\ writable = \mathsf{true}; \\ enumerable = \mathsf{false}; \\ configurable = \mathsf{false} \end{array} \right\} \end{array} \right\}$$

11.1.2 Global

$$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{isNaN}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$
$$\text{where } v = \left\{ \begin{array}{ll} \mathsf{true} & \text{if } \ toNumber(toPrimitive(getArgValue(args, \text{“}0\text{”}))) = \mathsf{NaN} \\ \mathsf{false} & \text{otherwise} \end{array} \right.$$

$$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{isFinite}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$
$$\text{where } v = \left\{ \begin{array}{ll} \mathsf{false} & \text{if } \ toNumber(toPrimitive(getArgValue(args, \text{“}0\text{”})) \in \{ \ \mathsf{NaN}, \mathsf{Inf}, \text{-}\mathsf{Inf} \ \} \\ \mathsf{true} & \text{otherwise} \end{array} \right.$$

11.1.3 Object

$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.constructor}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$
where $v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \textsf{Loc}$
$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.constructor}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H(\#temp)[@return \mapsto l]], A)$
where $v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \textsf{String} \cup \textsf{Number} \cup \textsf{Bool}$
$\wedge\, l = \mathit{newLocation}() \wedge o = \mathit{toObject}(v) \wedge H_1 = H\,[l \mapsto o]$
$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.constructor}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)$
where $v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \{\textsf{undefined}, \textsf{null}\}$
$\wedge\, l = \mathit{newLocation}() \wedge o = \mathit{NewObject}(\#ObjProto) \wedge H_1 = H\,[l \mapsto o]$

$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.getPrototypeOf}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)$
where $v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}$
$\wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})]$
$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.getPrototypeOf}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v_2]], A)$
where $v_1 = \mathit{getArgValue}(args, \text{“0”}) \wedge v_1 \in \mathit{Loc} \wedge v_2 = H(v)(@proto).value$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \notin \mathit{Loc}\\
&\qquad\qquad \land\, l_e = \mathit{newLocation}() \land\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \mathsf{undefined}]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \in \mathit{Loc} \land s = \mathit{toString}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) \land s \notin \mathit{dom}(H(v))\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \in \mathit{Loc} \land s = \mathit{toString}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) \land s \in \mathit{dom}(H(v))\\
&\qquad\qquad \land\, l = \mathit{newLocation}() \land o = \mathit{NewObject}(\#ObjProto)\\
&\qquad\qquad \land\, o_1 = \begin{cases} o\left[\begin{array}{ll} value \mapsto & \left\{\begin{array}{l} value : H(v)(s).value \\ writable : \mathsf{true} \\ enumerable : \mathsf{true} \\ configurable : \mathsf{true} \end{array}\right\} \\ writable \mapsto & \left\{\begin{array}{l} value : H(v)(s).writable \\ writable : \mathsf{true} \\ enumerable : \mathsf{true} \\ configurable : \mathsf{true} \end{array}\right\} \end{array}\right] & \text{if } \; \mathit{IsDataDescriptor}(H(v), s) \\ o & \text{otherwise} \end{cases}\\
&\qquad\qquad \land\, o_2 = o_1\left[\begin{array}{ll} enumerable \mapsto & \left\{\begin{array}{ll} value : H(v)(s).enumerable, & writable : \mathsf{true} \\ enumerable : \mathsf{true}, & configurable : \mathsf{true} \end{array}\right\} \\ configurable \mapsto & \left\{\begin{array}{ll} value : H(v)(s).configurable, & writable : \mathsf{true} \\ enumerable : \mathsf{true}, & configurable : \mathsf{true} \end{array}\right\} \end{array}\right]\\
&\qquad\qquad \land\, H_1 = H\,[l \mapsto o_2]
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyNames}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \notin \mathit{Loc}\\
&\qquad\qquad \land\, l_e = \mathit{newLocation}() \land\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyNames}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \in \mathit{Loc}\\
&\qquad\qquad \land\, l = \mathit{newLocation}() \land o = \mathit{NewArrayObject}(0) \land\; n = 0\\
&\qquad\qquad \land\, o_1 = o\left[\;\forall s \in \mathit{dom}(H(v)) : \mathit{toString}(n^{++}) \mapsto \left\{\begin{array}{l} value : s \\ writable : \mathsf{true} \\ enumerable : \mathsf{true} \\ configurable : \mathsf{true} \end{array}\right\}\;\right]\\
&\qquad\qquad \land H_1 = H\,[l \mapsto o_1]
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land (v \notin \mathit{Loc} \lor v \in \{\mathsf{null}\})\\
&\qquad\qquad \land\, l_e = \mathit{newLocation}() \land\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \; v = \mathit{getArgValue}(args, \text{“0”}) \land v \in \mathit{Loc} \land |args| = 1\\
&\qquad\qquad \land\, l = \mathit{newLocation}() \land o = \mathit{NewObject}(v) \land H_1 = H\,[l \mapsto o_1]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \; v_1 = \mathit{getArgValue}(args, \text{“0”}) \land v \in \mathit{Loc} \land |args| > 1 \land v_2 = \mathit{getArgValue}(args, \text{“1”})\\
&\qquad\qquad \land\, l = \mathit{newLocation}() \land o = \mathit{NewObject}(v_1)\\
&\qquad\qquad \land\, o_1 = \begin{cases} o & v_2 = \mathsf{undefined} \\ o\left[\forall x \in \mathit{dom}(H(v_2)) : x \mapsto \left\{\begin{array}{l} value : H(v_2)(x)(\text{“}value\text{”}) \\ writable : H(v_2)(x)(\text{“}writable\text{”}) \\ enumerable : H(v_2)(x)(\text{“}enumerable\text{”}) \\ configurable : H(v_2)(x)(\text{“}configurable\text{”}) \end{array}\right\}\right] & \text{otherwise} \end{cases}\\
&\qquad\qquad \land\, H_1 = H\,[l \mapsto o_1]
\end{aligned}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperty}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \notin Loc \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperty}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v_1 = getArgValue(args, \text{“0”}) \wedge v_1 \in Loc \wedge \; v_2 = getArgValue(args, \text{“2”}) \wedge v_2 \notin Loc \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperty}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v_1]], A) \\
\quad \text{where } \; v_1 = getArgValue(args, \text{“0”}) \wedge v_1 \in Loc \wedge \; v_2 = getArgValue(args, \text{“2”}) \wedge v_2 \in Loc \\
\qquad\quad \wedge\, s = toString(getArgValue(args, \text{“1”})) \\
\qquad\quad \wedge\, o = H(v_1) \left[s \mapsto \left\{ \begin{array}{l} value : Proto(H, v_2, \text{“}value\text{”}) \\ writable : Proto(H, v_2, \text{“}writable\text{”}) \\ enumerable : Proto(H, v_2, \text{“}enumerable\text{”}) \\ configurable : Proto(H, v_2, \text{“}configurable\text{”}) \end{array} \right\} \right] \\
\qquad\quad \wedge\, H_1 = H\,[l \mapsto o]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperties}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \notin Loc \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperties}\text{”}, args)\rrbracket(H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v_1 = getArgValue(args, \text{“0”}) \wedge v_1 \in Loc \wedge \; v_2 = getArgValue(args, \text{“1”}) \wedge (H_1, exc) = ToObject(H, v_2) \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_2 = H_1[l_e \mapsto NewExceptionObject(exc)] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.defineProperties}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v_1]], A) \\
\quad \text{where } \; v_1 = getArgValue(args, \text{“0”}) \wedge v_1 \in Loc \wedge \; v_2 = getArgValue(args, \text{“1”}) \wedge (H_1, l_1) = ToObject(H, v_2) \\
\qquad\quad \wedge\, o = H(v_1) \left[\forall x \in dom(H(v_2)) : x \mapsto \left\{ \begin{array}{l} value : H(v_2)(x)(\text{“}value\text{”}) \\ writable : H(v_2)(x)(\text{“}writable\text{”}) \\ enumerable : H(v_2)(x)(\text{“}enumerable\text{”}) \\ configurable : H(v_2)(x)(\text{“}configurable\text{”}) \end{array} \right\} \right] \\
\qquad\quad \wedge\, H_1 = H\,[l \mapsto o]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.seal}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \notin Loc \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.seal}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \in Loc \\
\qquad\quad \wedge\, H_1 = H \left[H(v) \mapsto \left[\begin{array}{l} \forall x \in Dom(H(v)) : x \mapsto H(v)(x) \text{ with } configurable = \textsf{false};\,, \\ @extensible \mapsto \textsf{false}; \end{array} \right] \right]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.freeze}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \notin Loc \\
\qquad\quad \wedge\, l_e = newLocation() \wedge \; H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})] \\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Object.seal}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A) \\
\quad \text{where } \; v = getArgValue(args, \text{“0”}) \wedge v \in Loc \\
\qquad\quad \wedge\, H_1 = H \left[H(v) \left[\begin{array}{l} \forall x \in P_1 : x \mapsto H(v)(x) \text{ with } writable = \textsf{false}; configurable = \textsf{false};\,, \\ \forall y \in P_2 : y \mapsto H(v)(y) \text{ with } configurable = \textsf{false};\,, \\ @extensible \mapsto \textsf{false}; \end{array} \right] \right] \\
\qquad\quad \wedge\, P_1 = \{x \mid x \in dom(H(v)(x)) \wedge IsDataDescriptor(x)\} \\
\qquad\quad \wedge\, P_2 = \{x \mid x \in dom(H(v)(x)) \wedge \neg IsDataDescriptor(x)\}
\end{array}
$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.preventExtensions”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}\\
&\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.preventExtensions”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc}\\
&\qquad\qquad \wedge\, H_1 = H\,[H(v) \mapsto [@extensible \mapsto \mathsf{false}]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isSealed”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}\\
&\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isSealed”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \mathsf{false}]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc} \wedge \exists x \in \mathit{dom}(H(v)) : H(v)(x).configurable = \mathsf{true}\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isSealed”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto b]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc} \wedge \forall x \in \mathit{dom}(H(v)) : H(v)(x).configurable = \mathsf{false}\\
&\qquad\qquad \wedge\, b = \begin{cases} \mathsf{true} & \text{if } \ H(v)(@extensible) = \mathsf{false}\\ \mathsf{false} & \text{otherwise}\end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isFrozen”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}\\
&\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isFrozen”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \mathsf{false}]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc}\\
&\qquad\qquad \wedge\, \exists x \in \mathit{dom}(H(v)) : \left(\begin{array}{l} (\mathit{IsDataDescription}(x) \wedge (H(v)(x).writable = \mathsf{true} \vee H(v)(x).configurable = \mathsf{true}))\\ \vee(\neg \mathit{IsDataDescription} \wedge H(v)(x).configurable = \mathsf{true}) \end{array} \right)\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isFrozen”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto b]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc}\\
&\qquad\qquad \wedge\, \forall x \in \mathit{dom}(H(v)) : \left(\begin{array}{l} (\mathit{IsDataDescription}(x) \wedge H(v)(x).writable = \mathsf{false} \wedge H(v)(x).configurable = \mathsf{false})\\ \vee(\neg \mathit{IsDataDescription} \wedge H(v)(x).configurable = \mathsf{false}) \end{array} \right)\\
&\qquad\qquad \wedge\, b = \begin{cases} \mathsf{true} & \text{if } \ H(v)(@extensible) = \mathsf{false}\\ \mathsf{false} & \text{otherwise}\end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isExtensible”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}\\
&\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.isExtensible”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto H(v)(@extensible)]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.keys”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}\\
&\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Object.keys”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathit{Loc}\ \wedge P = |\{x \mid x \in \mathit{dom}(H(v)) \wedge H(v)(x).enumerable = \mathsf{true}\}|\\
&\qquad\qquad \wedge\, n_1 = |P| \wedge l = \mathit{newLocation}() \wedge o = \mathit{NewArrayObject}(n_1) \wedge\ n_2 = 0\\
&\qquad\qquad \wedge\, o_1 = o \left[\ \forall x \in P : \mathit{toString}(n_2^{++}) \mapsto \left\{ \begin{array}{l} value : x\\ writable : \mathsf{true}\\ enumerable : \mathsf{true}\\ configurable : \mathsf{true} \end{array} \right\} \right]\\
&\qquad\qquad \wedge\, H_1 = H\,[l \mapsto o_1]
\end{aligned}$$

11.1.4 Object.prototype

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toString”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)$
  where $\mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) = \text{undefined} \wedge s = \text{“[object Undefined]”}$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toString”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)$
  where $\mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) = \text{null} \wedge s = \text{“[object Null]”}$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toString”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto s]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_1 \notin \{\text{undefined}, \text{null}\}$
    $\wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge s = \text{“[object”} + H(v_2)(@class) + \text{“]”}$

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{toObject}(H, v)$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge H_1(v_2)(\text{“}\mathit{toString}\text{”}) \notin \mathit{Loc}$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\text{TypeError})]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge v_3 = H_1(v_2)(\text{“}\mathit{toString}\text{”}) \wedge v_3 \in \mathit{Loc}$
    $\neg \mathit{IsCallable}(H_1, v_3) \wedge\ l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\text{TypeError})]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto l_e]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge v_3 = H_1(v_2)(\text{“}\mathit{toString}\text{”}) \wedge v_3 \in \mathit{Loc}$
    $\mathit{IsCallable}(H_1, v_3)$?????

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.valueOf”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{toObject}(H, v)$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.valueOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v_1]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge H(v)(@class) = \text{“}\mathit{Object}\text{”}$

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.hasOwnProperty”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{toObject}(H, v)$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.hasOwnProperty”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto b]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge v_3 = \mathit{getArgValue}(args, \text{“0”})$
    $\wedge\, s = \mathit{toString}(\mathit{toPrimitive}(v_3)) \wedge b = \mathit{HasOwnProperty}(H_1, v_2, s)$

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.isPrototypeOf”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \text{false}]], A)$
  where $v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathit{Loc}$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.isPrototypeOf”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v_1 = \mathit{getArgValue}(args, \text{“0”}) \wedge v_1 \in \mathit{Loc} \wedge v_2 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{toObject}(H, v_2)$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.isPrototypeOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v_4]], A)$
  where $v_1 = \mathit{getArgValue}(args, \text{“0”}) \wedge v_1 \in \mathit{Loc} \wedge v_2 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_3) = \mathit{toObject}(H, v_2)$
    $v_4 = \mathit{inherit}(H_1, v_1, v_3)$

$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.propertyIsEnumerable”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)$
  where $v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{toObject}(H, v)$
    $\wedge\, l_e = \mathit{newLocation}() \wedge\ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.propertyIsEnumerable”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto \text{false}]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge v_3 = \mathit{getArgValue}(args, \text{“0”})$
    $\wedge\, s = \mathit{toString}(\mathit{toPrimitive}(v_3)) \wedge \neg \mathit{HasOwnProperty}(H_1, v_2, s)$
$\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Object.prototype.propertyIsEnumerable”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto b]], A)$
  where $v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, v_2) = \mathit{toObject}(H, v_1) \wedge v_3 = \mathit{getArgValue}(args, \text{“0”})$
    $\wedge\, s = \mathit{toString}(\mathit{toPrimitive}(v_3)) \wedge \mathit{HasOwnProperty}(H_1, v_2, s) \wedge b = H_1(v_2)(s).enumerable$

11.1.5 Function

11.1.6 Function.prototype

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.toString”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v \notin \mathsf{Loc} \land\ l_e = newLocation() \land\ H_1 = H[l_e \mapsto NewExceptionObject(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.toString”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v \in \mathsf{Loc} \land H(v)(@class) \neq \text{“}Function\text{”}\\
&\qquad\qquad \land l_e = newLocation() \land\ H_1 = H[l_e \mapsto NewExceptionObject(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.toString”}, args)\rrbracket(H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v \in \mathsf{Loc} \land H(v)(@class) = \text{“}Function\text{”} \land\ s = fid2String(H(v)(@function))
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.apply”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v \notin \mathsf{Loc} \land\ l_e = newLocation() \land\ H_1 = H[l_e \mapsto NewExceptionObject(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.apply”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v \in \mathsf{Loc} \land \neg IsCallable(H, v)\\
&\qquad\qquad \land l_e = newLocation() \land\ H_1 = H[l_e \mapsto NewExceptionObject(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.apply”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{fun} = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v_{fun} \in \mathsf{Loc} \land IsCallable(H, v_{fun})\\
&\qquad\qquad \land v_{arg} = getArgValue(args, \text{“1”}) \land v_{arg} \notin \{\ \mathsf{null}, \mathsf{undefined}\ \} \land v_{arg} \notin \mathsf{Loc}\\
&\qquad\qquad \land l_e = newLocation() \land\ H_1 = H[l_e \mapsto NewExceptionObject(\mathsf{TypeError})]\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.apply”}, args)\rrbracket(H,A) = (H_2, A_1)\\
&\quad \text{where } v_{fun} = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v_{fun} \in \mathsf{Loc} \land IsCallable(H, v_{fun})\\
&\qquad\qquad \land v_{arg} = getArgValue(args, \text{“1”}) \land v_{arg} \in \{\ \mathsf{null}, \mathsf{undefined}\ \} \land\ o_{arg} = NewArgObject(0)\\
&\qquad\qquad \land l_{arg} = newLocation() \land H_1 = H[l_{arg} \mapsto o_{arg}] \land\ l_{scope} = newLocation()\\
&\qquad\qquad \land A_1 = PushStack(H(v_{fun})(@scope), l_{scope})\\
&\qquad\qquad \land o_{scope} = \left\{ \begin{array}{l} arguments \mapsto \left\{ \begin{array}{l} value : l_{arg}, \\ writable : \mathsf{true}, \\ unumarable : \mathsf{false}, \\ configurable : \mathsf{false} \end{array} \right\}, \\ @this \mapsto getArgValue(args, \text{“0”}), \\ @up \mapsto A, \\ @return \mapsto H(\#temp)(@return) \end{array} \right\} \land H_2 = H_1[l_{new} \mapsto o_{scope}]\\
&\qquad\qquad \land fid_{callee} = H(v_{fun})(@function) \land cp_{\textit{after-call}} = getAftercallFromCall_P(cp)\\
&\qquad\qquad \land \hookrightarrow := \hookrightarrow \cup \{\ (cp, (fid_{callee}, \mathsf{ENTRY})), ((fid_{callee}, \mathsf{EXIT}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \land \overset{\mathsf{exc}}{\hookrightarrow} := \overset{\mathsf{exc}}{\hookrightarrow} \cup \{\ ((fid_{callee}, \mathsf{EXIT\text{-}EXC}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \land BelongsTo := BelongsTo \cup \{\ (l_{new}, cp)\ \}\\
&\mathcal{I}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Function.prototype.apply”}, args)\rrbracket(H,A) = (H_2, A_1)\\
&\quad \text{where } v_{fun} = \mathcal{V}_{cp}\llbracket \mathtt{this}\rrbracket(H,A) \land v_{fun} \in \mathsf{Loc} \land IsCallable(H, v_{fun})\\
&\qquad\qquad \land v_{arg} = getArgValue(args, \text{“1”}) \land v_{arg} \notin \{\ \mathsf{null}, \mathsf{undefined}\ \} \land v_{arg} \in \mathsf{Loc} \land n_{len} = Proto(H, v_{arg}, \text{“}length\text{”})\\
&\qquad\qquad \land o_{arg} = NewArgObject(n_{len}) \left[\forall i \in \{\ 0, ..., n_{len} - 1\ \} : \text{“}i\text{”} \mapsto Proto(H, v_{arg}, \text{“}i\text{”})\right] \land l_{arg} = newLocation()\\
&\qquad\qquad \land H_1 = H[l_{arg} \mapsto o_{arg}] \land o_{arg} = NewArgObject(0) \land\ l_{arg} = newLocation() \land H_1 = H[l_{arg} \mapsto o_{arg}]\\
&\qquad\qquad \land l_{scope} = newLocation() \land\ A_1 = PushStack(H(v_{fun})(@scope), l_{scope})\\
&\qquad\qquad \land o_{scope} = \left\{ \begin{array}{l} arguments \mapsto \left\{ \begin{array}{l} value : l_{arg}, \\ writable : \mathsf{true}, \\ unumarable : \mathsf{false}, \\ configurable : \mathsf{false} \end{array} \right\}, \\ @this \mapsto getArgValue(args, \text{“0”}), \\ @up \mapsto A, \\ @return \mapsto H(\#temp)(@return) \end{array} \right\} \land H_2 = H_1[l_{new} \mapsto o_{scope}]\\
&\qquad\qquad \land fid_{callee} = H(v_{fun})(@function) \land cp_{\textit{after-call}} = getAftercallFromCall_P(cp)\\
&\qquad\qquad \land \hookrightarrow := \hookrightarrow \cup \{\ (cp, (fid_{callee}, \mathsf{ENTRY})), ((fid_{callee}, \mathsf{EXIT}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \land \overset{\mathsf{exc}}{\hookrightarrow} := \overset{\mathsf{exc}}{\hookrightarrow} \cup \{\ ((fid_{callee}, \mathsf{EXIT\text{-}EXC}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \land BelongsTo := BelongsTo \cup \{\ (l_{new}, cp)\ \}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Function.prototype.call''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \notin \textsf{Loc} \wedge\ l_e = newLocation() \wedge\ H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Function.prototype.call''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \in \textsf{Loc} \wedge \neg IsCallable(H,v)\\
&\qquad\qquad \wedge\ l_e = newLocation() \wedge\ H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Function.prototype.call''}, args)]\!](H,A) = (H_2, A_1)\\
&\quad \text{where } v_{fun} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{fun} \in \textsf{Loc} \wedge IsCallable(H, v_{fun}) \wedge n_{len} = ToUInt32(getArgValue(args, \text{``}length\text{''}))\\
&\qquad\qquad \wedge\ o_{arg} = NewArgObject(n_{len} - 1)\,[\forall i \in \{\ 1, ..., n_{len}\ \} : \text{``}i\text{''} \mapsto getArgValue(args, \text{``}i\text{''})]\\
&\qquad\qquad \wedge\ l_{arg} = newLocation() \wedge H_1 = H[l_{arg} \mapsto o_{arg}]\\
&\qquad\qquad \wedge\ l_{scope} = newLocation() \wedge\ A_1 = PushStack(H(v_{fun})(@scope), l_{scope})\\
&\qquad\qquad \wedge\ o_{scope} = \left\{ \begin{array}{l} arguments \mapsto \left\{ \begin{array}{l} value : l_{arg},\\ writable : \textsf{true},\\ unumarable : \textsf{false},\\ configurable : \textsf{false} \end{array} \right\},\\ @this \mapsto getArgValue(args, \text{``0''}),\\ @up \mapsto A,\\ @return \mapsto H(\#temp)(@return) \end{array} \right\} \wedge H_2 = H_1[l_{new} \mapsto o_{scope}]\\
&\qquad\qquad \wedge\ fid_{callee} = H(v_{fun})(@function) \wedge cp_{\textit{after-call}} = getAftercallFromCall_P(cp)\\
&\qquad\qquad \wedge \hookrightarrow := \hookrightarrow \cup \{\ (cp, (fid_{callee}, \textsf{ENTRY})), ((fid_{callee}, \textsf{EXIT}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \wedge \overset{\text{exc}}{\hookrightarrow} := \overset{\text{exc}}{\hookrightarrow} \cup \{\ ((fid_{callee}, \textsf{EXIT-EXC}), cp_{\textit{after-call}})\ \}\\
&\qquad\qquad \wedge BelongsTo := BelongsTo \cup \{\ (l_{new}, cp)\ \}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Function.prototype.bind''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \notin \textsf{Loc} \wedge\ l_e = newLocation() \wedge\ H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Function.prototype.bind''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \in \textsf{Loc} \wedge \neg IsCallable(H,v)\\
&\qquad\qquad \wedge\ l_e = newLocation() \wedge\ H_1 = H[l_e \mapsto NewExceptionObject(\textsf{TypeError})]
\end{aligned}$$

11.1.7 Array

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.constuctor''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } getArgValue(args, \text{``}length\text{''}) = 0 \wedge l = newLocation() \wedge H_1 = H[l \mapsto NewArrayObject(0)]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.constuctor''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{len} = getArgValue(args, \text{``}length\text{''}) \wedge v_{len} = 1 \wedge v_{len} \in \textsf{Number} \wedge v_{len} \neq ToUInt32(v_{len})\\
&\qquad\qquad \wedge\ l_e = newLocation() \wedge H_1 = H[l_e \mapsto NewExceptionObject(\textsf{RangeError})]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.constuctor''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } v_{len} = getArgValue(args, \text{``}length\text{''}) \wedge v_{len} = 1 \wedge v_{len} \in \textsf{Number} \wedge n_{len32} = ToUInt32(v_{len}) \wedge v_{len} = n_{len32}\\
&\qquad\qquad \wedge\ l = newLocation() \wedge H_1 = H[l \mapsto NewArrayObject(n_{len32})]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.constuctor''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } v_{len} = getArgValue(args, \text{``}length\text{''}) \wedge v_{len} = 1 \wedge v_{len} \notin \textsf{Number}\\
&\qquad\qquad \wedge\ l = newLocation() \wedge H_1 = H[l \mapsto NewArrayObject(1)[\text{``0''} \mapsto v_{len}]]\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.constuctor''}, args)]\!](H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } v_{len} = getArgValue(args, \text{``}length\text{''}) \wedge v_{len} > 1 \wedge l = newLocation()\\
&\qquad\qquad \wedge\ H_1 = H[l \mapsto NewArrayObject(v_{len})[\forall i \in \{\ 0, ..., v_{len} - 1\ \} : \text{``}i\text{''} \mapsto getArgValue(args, \text{``}i\text{''})]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.isArray''}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto \textsf{false}]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \notin \textsf{Loc}\\
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{``Array.isArray''}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto b]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v \in \textsf{Loc} \wedge b = \left\{ \begin{array}{ll} \textsf{true} & \text{if } H(v)(@class) = \text{``}Array\text{''}\\ \textsf{false} & \text{otherwise} \end{array} \right.
\end{aligned}$$

11.1.8 Array.prototype

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.toString}\text{”}, args)\rrbracket(H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, exc) = \textit{ToObject}(H, v)\\
&\qquad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.toString}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto s_{join}]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, l) = \textit{ToObject}(H, v)\\
&\qquad \wedge\, s = \begin{cases} \text{“}\textsf{[object}\text{”} + H(v_2)(@class) + \text{“}\textsf{]}\text{”} & \text{if } \textit{Proto}(H, l, \text{“}join\text{”}) \notin \textsf{Loc} \vee \textit{IsCallable}(H_1, \textit{Proto}(H, l, \text{“}join\text{”}))\\ s_{join} & \text{otherwise} \end{cases}\\
&\qquad \wedge\, n_{len} = \textit{Proto}(H_1, l, \text{“}length\text{”})\\
&\qquad \wedge\, s_i = \begin{cases} \text{“”} & \text{if } \textit{Proto}(H_1, l, \text{“}i\text{”}) \in \{\textsf{null}, \textsf{undefined}\}\\ \textit{ToString}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}i\text{”}))) & \text{otherwise} \end{cases}\\
&\qquad \wedge\, s_{join} = \begin{cases} \text{“”} & \text{if } n_{len} = 0\\ s_0 + \text{“},\text{”} + s_1 + \text{“},\text{”} + ... + \text{“},\text{”} + s_{n_{len}-1} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.concat}\text{”}, args)\rrbracket(H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, exc) = \textit{ToObject}(H, v)\\
&\qquad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.concat}\text{”}, args)\rrbracket(H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto l_{new}]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, l) = \textit{ToObject}() \wedge n = 0\\
&\qquad \wedge\, o = \textit{NewArrayObject}(0)\left[\forall i \in \{0, ...H(l)(\text{“}length\text{”}) - 1\} : \textit{toString}(n^{++}) \mapsto H(l)(\text{“}i\text{”})\right]\\
&\qquad \wedge\, v_{len} = \textit{getArgValue}(args, \text{“}length\text{”}) \wedge v_i = \textit{getArgValue}(args, \text{“}i\text{”})\\
&\qquad \wedge\, o_1 = o\left[\forall i \in \{0, v_{len} - 1\} : \begin{cases} \begin{array}{l}\forall j \in \{0, ...H(v_i)(\text{“}length\text{”}) - 1\} :\\ \qquad \textit{toString}(n^{++}) \mapsto \textit{Proto}(H, v_i, \text{“}j\text{”})\end{array} & \begin{array}{l}\text{if } v_i \in \textsf{Loc}\\ \quad \wedge H(v_i)(@class) = \text{“}Array\text{”}\end{array}\\ \textit{toString}(n^{++}) \mapsto v_i & \text{otherwise} \end{cases}\right]\\
&\qquad \wedge\, l_{new} = \textit{newLocatioin}() \wedge H_2 = H_1[l_{new} \mapsto o_1]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.join}\text{”}, args)\rrbracket(H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, exc) = \textit{ToObject}(H, v)\\
&\qquad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.join}\text{”}, args)\rrbracket(H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto s]], A)\\
&\quad \text{where } v = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H, A) \wedge (H_1, l) = \textit{ToObject}(H, v) \wedge v_0 = \textit{getArgValue}(args, \text{“0”})\\
&\qquad \wedge\, s_{sep} = \begin{cases} \text{“},\text{”} & \text{if } v_0 = \textsf{undefined}\\ \textit{ToString}(\textit{ToPrimitive}(v_o)) & \text{otherwise} \end{cases}\\
&\qquad \wedge\, n_{len} = \textit{Proto}(H_1, l, \text{“}length\text{”})\\
&\qquad \wedge\, s_i = \begin{cases} \text{“”} & \text{if } \textit{Proto}(H_1, l, \text{“}i\text{”}) \in \{\textsf{null}, \textsf{undefined}\}\\ \textit{ToString}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}i\text{”}))) & \text{otherwise} \end{cases}\\
&\qquad \wedge\, s = \begin{cases} \text{“”} & \text{if } n_{len} = 0\\ s_0 + s_{sep} + s_1 + s_{sep} + ... + s_{sep} + s_{n_{len}-1} & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.pop}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, exc) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.pop}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto v]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, l) = \textit{ToObject}(H, v_{this}) \wedge n_{len} = \textit{ToUInt32}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}length\text{”})))\\
\qquad\quad \wedge\, (H_2, v) = \begin{cases} (\textit{PropStore}(H_1, l, \text{“}length\text{”}, 0), \textsf{undefined}) & \text{if } n_{len} = 0\\ \begin{pmatrix} \textit{Delete}(\textit{PropStore}(H_1, l, \text{“}length\text{”}, n_{len}-1), l, \textit{ToString}(n_{len}-1)).1,\\ \textit{Proto}(H_1, l, \textit{ToString}(n_{len}-1)) \end{pmatrix} & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.push}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, exc) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.push}\text{”}, args)\rrbracket(H,A) = (H_3[\#temp \mapsto H_3(\#temp)[@return \mapsto n]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, l) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, n_{len} = \textit{ToUInt32}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}length\text{”}))) \wedge n_{arglen} = \textit{getArgValue}(args, \text{“}length\text{”})\\
\qquad\quad \wedge\, H_2 = H_1\,[l \mapsto H_1(l)\,[\forall i \in \{0, ..., n_{arglen}-1\} : \textit{ToString}(n_{len}+i) \mapsto \textit{getArgValue}(args, \text{“}i\text{”})]]\\
\qquad\quad \wedge\, n = n_{len} + n_{arglen} \wedge H_3 = \textit{PropStore}(H_2, l, \text{“}length\text{”}, n)
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.reverse}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, exc) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.push}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, l) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, n_{len} = \textit{ToUInt32}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}length\text{”}))) \wedge n_{last} = n_{len} - 1 \wedge\ n_{mid} = native.floor(n_{len}/2)\\
\qquad\quad \wedge\, b_i = \textit{HasProperty}(H_1, l, i) \wedge o = H(l)\\
\qquad\quad \wedge\, o_1 = o \left[\forall i \in \{0, ..., n_{mid}-1\} : \begin{cases} \begin{array}{l} \textit{ToString}(i) \mapsto \textit{Proto}(H_1, l, \textit{ToString}(n_{last}-i)),\\ \textit{ToString}(n_{last}-i) \mapsto \textit{Proto}(H_1, l, \textit{ToString}(i)) \end{array} & \text{if } b_i \wedge b_{n_{last}-i}\\ \textit{ToString}(i) \mapsto \textit{Proto}(H_1, l, \textit{ToString}(n_{last}-i) & \text{if } \neg b_i \wedge b_{n_{last}-i}\\ \textit{ToString}(n_{last}-i) \mapsto \textit{Proto}(H_1, l, \textit{ToString}(i) & \text{if } b_i \wedge \neg b_{n_{last}-i} \end{cases} \right]\\
\qquad\quad \wedge\, H_2 = H_1[l \mapsto o_1]
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.shift}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, exc) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, l_e = \textit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \textit{NewExceptionObject}(exc)]\\
\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Array.prototype.shift}\text{”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto v]], A)\\
\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this} \rrbracket(H,A) \wedge (H_1, l) = \textit{ToObject}(H, v_{this})\\
\qquad\quad \wedge\, n_{len} = \textit{ToUInt32}(\textit{ToPrimitive}(\textit{Proto}(H_1, l, \text{“}length\text{”})))\\
\qquad\quad \wedge\, (H_2, v) = \begin{cases} (H_1, \textsf{undefined}) & \text{if } n_{len} = 0\\ (H_1[l \mapsto o_1], \textit{Proto}(H_1, l, \text{“}0\text{”}) & \text{otherwise} \end{cases}\\
\qquad\quad \wedge\, o_1 = H_1(l)\,\big[\forall i \in \{1, ..., n_{len}-1\} :\ \textit{ToString}(i) \mapsto \textit{Proto}(H_1, l, \textit{ToString}(i-1)) \quad \text{if } \textit{HasProperty}(H_1, l, i)\ \big]
\end{array}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\textsf{“Array.prototype.slice”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, l_e = \mathit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\textsf{“Array.prototype.slice”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto l_{new}]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}length\text{”})))\\
&\qquad\quad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”})))\\
&\qquad\quad \wedge\, n_{start} = \begin{cases} 0 & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} \leq 0\\ n_{len} + n_{argstart} & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} > 0\\ n_{argstart} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} < n_{len}\\ n_{len} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} \geq n_{len} \end{cases}\\
&\qquad\quad \wedge\, n_{argend} = \begin{cases} n_{len} & \text{if } \mathit{getArgValue}(args, \text{“1”}) = \textsf{undefined}\\ \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\, n_{end} = \begin{cases} 0 & \text{if } n_{argend} < 0 \wedge n_{len} + n_{argend} \leq 0\\ n_{len} + n_{argend} & \text{if } n_{argend} < 0 \wedge n_{len} + n_argend > 0\\ n_{argend} & \text{if } n_{argend} \geq 0 \wedge n_{argend} < n_{len}\\ n_{len} & \text{if } n_{argend} \geq 0 \wedge n_{argend} \geq n_{len} \end{cases}\\
&\qquad\quad \wedge\, o = \mathit{NewArrayObject}(0)\\
&\qquad\quad \wedge\, o_1 = o\left[\forall i \in \{n_{start}, ..., n_{end} - 1\} :\ \mathit{ToString}(i) \mapsto \mathit{Proto}(H_1, l, \mathit{ToString}(i - 1)) \quad \text{if } \mathit{HasProperty}(H_1, l, i)\ \right]\\
&\qquad\quad \wedge\, l_{new} = \mathit{newLocation}() \wedge H_2 = H_1[l_{new} \mapsto o_1]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\textsf{“Array.prototype.splice”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, exc) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, l_e = \mathit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\textsf{“Array.prototype.splice”}, args)]\!](H, A) = (H_3[\#temp \mapsto H_3(\#temp)[@return \mapsto l_{new}]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, n_{arglen} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“}length\text{”}))) \wedge n_{arglen} = 2\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}length\text{”})))\\
&\qquad\quad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”})))\\
&\qquad\quad \wedge\, n_{start} = \begin{cases} 0 & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} \leq 0\\ n_{len} + n_{argstart} & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} > 0\\ n_{argstart} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} < n_{len}\\ n_{len} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} \geq n_{len} \end{cases}\\
&\qquad\quad \wedge\, n_{argdel} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”})))\\
&\qquad\quad \wedge\, n_{del} = \mathit{min}(\mathit{max}(n_{argdel}, 0), n_{len} - n_{start})\\
&\qquad\quad \wedge\, o = \mathit{NewArrayObject}(0)\\
&\qquad\quad \wedge\, o_1 = o\left[\forall i \in \{0, ..., n_{del} - 1\} :\ \mathit{ToString}(i) \mapsto \mathit{Proto}(H_1, l, n_{start} + i) \quad \text{if } \mathit{HasProperty}(H_1, l, i + n_{del})\ \right]\\
&\qquad\quad \wedge\, H_2 = H_1\,[l \mapsto H_1(l)\,[\forall i \in \{n_{start}, ..., n_{start} + n_{del} - 1\} :\ \mathit{ToString}(i) \not\mapsto \textit{// delete prop}]]\\
&\qquad\quad \wedge\, l_{new} = \mathit{newLocation}() \wedge H_3 = H_2[l_{new} \mapsto o_1]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\textsf{“Array.prototype.splice”}, args)]\!](H, A) = (H_2[\#temp \mapsto H_2(\#temp)[@return \mapsto l_{new}]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, n_{arglen} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“}length\text{”}))) \wedge n_{arglen} > 2 \wedge n_{replace} = n_{arglne} - 2\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}length\text{”})))\\
&\qquad\quad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”})))\\
&\qquad\quad \wedge\, n_{start} = \begin{cases} 0 & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} \leq 0\\ n_{len} + n_{argstart} & \text{if } n_{argstart} < 0 \wedge n_{len} + n_{argstart} > 0\\ n_{argstart} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} < n_{len}\\ n_{len} & \text{if } n_{argstart} \geq 0 \wedge n_{argstart} \geq n_{len} \end{cases}\\
&\qquad\quad \wedge\, n_{argdel} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”})))\\
&\qquad\quad \wedge\, n_{del} = \mathit{min}(\mathit{max}(n_{argdel}, 0), n_{len} - n_{start})\\
&\qquad\quad \wedge\, o = \mathit{NewArrayObject}(0)\\
&\qquad\quad \wedge\, o_1 = o\left[\forall i \in \{0, ..., n_{del} - 1\} : \mathit{ToString}(i) \mapsto \mathit{Proto}(H_1, l, n_{start} + i) \quad \text{if } \mathit{HasProperty}(H_1, l, i + n_{del})\ \right]\\
&\qquad\quad \wedge\, H_2 = H_1 \left[l \mapsto H_1(l) \left[\begin{array}{l} \forall i \in \{n_{start}, ..., n_{start} + n_{del} - 1\} : \mathit{ToString}(i) \not\mapsto, \textit{// delete prop}\\ \forall j \in \{0, ..., n_{del} - 1\} : \mathit{ToString}(j) \mapsto \mathit{getArgValue}(args, n_{start} + j),\\ \forall k \in \{0, ..., n_{len} - n_{start} - n_{del} - 1\} :\\ \quad \mathit{ToString}(n_{start} + n_{replace} + k) \mapsto \mathit{Proto}(H_1, l, n_{start} + n_{del} + k) \end{array} \right] \right]\\
&\qquad\quad \wedge\, l_{new} = \mathit{newLocation}() \wedge H_3 = H_2[l_{new} \mapsto o_1]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.unshift”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, exc) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, l_e = \mathit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.unshift”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto n]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, n_{arglen} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“}\mathit{length}\text{”})))\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}\mathit{length}\text{”})))\\
&\qquad\quad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”})))\\
&\qquad\quad \wedge\, H_2 = H_1\left[l \mapsto H_1(l) \left[\begin{array}{l} \forall i \in \{0, ..., n_{len}-1\} : \\ \qquad \mathit{ToString}(n_{arglen}+i) \mapsto \mathit{Proto}(H_1, l, i) \quad \text{if } \mathit{HasProperty}(H_1, l, i) \\ \forall j \in \{0, ..., n_{arglen}-1\} : \mathit{ToString}(j) \mapsto \mathit{getArgValue}(args, j) \end{array} \right]\right]\\
&\qquad\quad \wedge\, l_{new} = \mathit{newLocation}() \wedge H_3 = H_2[l_{new} \mapsto o_1] \wedge n = n_{len} + n_{arglen}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.indexOf”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, exc) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, l_e = \mathit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.indexOf”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto n_{pos}]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this}) \wedge v_{find} = \mathit{getArgValue}(args, \text{“0”})\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}\mathit{length}\text{”})))\\
&\qquad\quad \wedge\, n_{start} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) & \text{if } \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) > 1 \\ 0 & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\, n_{pos} = \begin{cases} -1 & \text{if } n_{start} > n_{len} - 1 \\ -1 & \text{if } \begin{array}{l} \neg\exists i \in \{n_{start}, ...n_{len}-1\} : \\ \quad (\mathit{HasProperty}(H_1, l, i) \wedge \mathit{Proto}(H, l, i) = v_{find}) \end{array} \\ \mathit{min}\left(\begin{array}{l} \forall i \in \{n_{start}, ...n_{len}-1\} : \\ \quad \left(\begin{array}{l} \mathit{HasProperty}(H_1, l, i) \\ \wedge \mathit{Proto}(H, l, i) = v_{find} \end{array} \right) \end{array} \right) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.lastIndexOf”}, args)\rrbracket(H,A) = (H_2[\#temp \mapsto H_2(\#temp)[@exception \mapsto l_e]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, exc) = \mathit{ToObject}(H, v_{this})\\
&\qquad\quad \wedge\, l_e = \mathit{newLocation}() \wedge H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(exc)]\\
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“Array.prototype.lastIndexOf”}, args)\rrbracket(H,A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto n_{pos}]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}\llbracket \texttt{this}\rrbracket(H,A) \wedge (H_1, l) = \mathit{ToObject}(H, v_{this}) \wedge v_{find} = \mathit{getArgValue}(args, \text{“0”})\\
&\qquad\quad \wedge\, n_{len} = \mathit{ToUInt32}(\mathit{ToPrimitive}(\mathit{Proto}(H_1, l, \text{“}\mathit{length}\text{”})))\\
&\qquad\quad \wedge\, n_{end} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) & \text{if } \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) > 1 \\ n_{len} - 1 & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\, n_{pos} = \begin{cases} -1 & \text{if } \begin{array}{l} \neg\exists i \in \{0, ...n_{end}\} : \\ \quad (\mathit{HasProperty}(H_1, l, i) \wedge \mathit{Proto}(H, l, i) = v_{find}) \end{array} \\ \mathit{max}\left(\begin{array}{l} \forall i \in \{0, ...n_{len}\} : \\ \quad \left(\begin{array}{l} \mathit{HasProperty}(H_1, l, i) \\ \wedge \mathit{Proto}(H, l, i) = v_{find} \end{array} \right) \end{array} \right) & \text{otherwise} \end{cases}
\end{aligned}$$

11.1.9 String

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“String.constructor”}, args)\rrbracket(H,A) = H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l_{new}]], A)\\
&\quad \text{where } s = \begin{cases} \text{“”} & \text{if } \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) < 1 \\ \mathit{toString}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\, l_{new} = \mathit{newLocation}() \wedge H_1 = H[l_{new} \mapsto \mathit{Newstring}(s)]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}\llbracket \textsf{BuiltintCall}(\text{“String.fromCharCode”}, args)\rrbracket(H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\
&\quad \text{where } s_{init} = \text{“”} \wedge s_i = \mathit{toString}(\mathit{native.toChar}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“}i\text{”})))) \quad \mathit{//\, java,\, scala}\\
&\qquad\quad \wedge\, n = \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) \wedge s = s_{init} + s_0 + s_i + ... + s_n
\end{aligned}$$

11.1.10 String.prototype

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toString”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \not\in \mathsf{Loc} \land \ l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toString”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \mathsf{Loc} \land H(v)(@class) \neq \text{“}\mathit{String}\text{”} \\
\qquad\qquad \land\, l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toString”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \mathsf{Loc} \land H(v)(@class) = \text{“}\mathit{String}\text{”} \land \ s = H(v)(@primitive)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.valueOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \not\in \mathsf{Loc} \land \ l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.valueOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \mathsf{Loc} \land H(v)(@class) \neq \text{“}\mathit{String}\text{”} \\
\qquad\qquad \land\, l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.valueOf”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \mathsf{Loc} \land H(v)(@class) = \text{“}\mathit{String}\text{”} \land \ s = H(v)(@primitive)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charAt”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \{\ \mathsf{undefined}, \mathsf{null}\ \} \\
\qquad\qquad \land\, l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charAt”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s_2]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \not\in \{\ \mathsf{undefined}, \mathsf{null}\ \} \land s_1 = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \land\, n_{pos} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \land n_{size} = s.length \\
\qquad\qquad \land\, s_2 = \begin{cases} \text{“”} & \text{if } \ n_{pos} < 0 \lor n_{pos} \geq n_{size} \\ s(n_{pos}) & \text{otherwise} \end{cases} \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charCodeAt”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \{\ \mathsf{undefined}, \mathsf{null}\ \} \\
\qquad\qquad \land\, l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charCodeAt”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \not\in \{\ \mathsf{undefined}, \mathsf{null}\ \} \land s_1 = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \land\, n_{pos} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \land n_{size} = s.length \\
\qquad\qquad \land\, n = \begin{cases} \mathsf{NaN} & \text{if } \ n_{pos} < 0 \lor n_{pos} \geq n_{size} \\ native.toInt(s(n_{pos})) & \text{otherwise} \quad \textit{// java, scala} \end{cases}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.concat”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \in \{\ \mathsf{undefined}, \mathsf{null}\ \} \\
\qquad\qquad \land\, l_e = \mathit{newLocation}() \land \ H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.concat”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \land v \not\in \{\ \mathsf{undefined}, \mathsf{null}\ \} \land s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \land\, s_i = \mathit{ToString}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \land n_{len} = \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) \\
\qquad\qquad \land\, s = s_{this} + s_0 + s_1 + \ldots + s_{n_{len}-1}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.indexOf}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{ \; \textsf{undefined}, \textsf{null} \; \} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.indexOf}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n_{pos}]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{ \; \textsf{undefined}, \textsf{null} \; \} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s_{find} = \mathit{ToString}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \\
\qquad\qquad \wedge\, n_{argstart} = \begin{cases} 0 & \text{if } \; \mathit{getArgValue}(args, \text{“1”}) = \textsf{undefined} \\ \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, n_{start} = min(max(n_{argstart}, 0)s_{this}.length) \\
\qquad\qquad \wedge\, n_{pos} = native.string.indexOf(s_{this}, s_{find}, n_{start}) \quad \textit{// java, scala}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.lastIndexOf}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{ \; \textsf{undefined}, \textsf{null} \; \} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.lastIndexOf}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n_{pos}]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{ \; \textsf{undefined}, \textsf{null} \; \} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s_{find} = \mathit{ToString}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \\
\qquad\qquad \wedge\, n_{argend} = \begin{cases} \textsf{Inf} & \text{if } \; \mathit{getArgValue}(args, \text{“1”}) = \textsf{undefined} \\ \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, n_{start} = min(max(n_{argstart}, 0)s_{this}.length) \\
\qquad\qquad \wedge\, n_{pos} = native.string.lastIndexOf(s_{this}, s_{find}, n_{start}) \quad \textit{// java, scala}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.localeCompare}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{ \; \textsf{undefined}, \textsf{null} \; \} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.localeCompare}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{ \; \textsf{undefined}, \textsf{null} \; \} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s_{that} = \mathit{ToString}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \\
\qquad\qquad \wedge\, n = native.string.compare(s_{this}, s_{that}) \quad \textit{// java, scala}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.slice}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{ \; \textsf{undefined}, \textsf{null} \; \} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.slice}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where } \; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{ \; \textsf{undefined}, \textsf{null} \; \} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \\
\qquad\qquad \wedge\, n_{argend} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”}))) \\
\qquad\qquad \wedge\, n_{strlen} = s_{this}.\mathit{length} \\
\qquad\qquad \wedge\, n_{start} = \begin{cases} max(n_{strlen} + n_{argstart}, 0) & \text{if } \; n_{argstart} < 0 \\ min(n_{argstart}, n_{strlen}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, n_{end} = \begin{cases} max(n_{strlen} + n_{argend}, 0) & \text{if } \; n_{argend} < 0 \\ min(n_{argend}, n_{strlen}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, s = native.string.slice(s_{this}, n_{start}, n_{end}) \quad \textit{// java, scala}
\end{array}
$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.substring}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.substring}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, n_{strlen} = s_{this}.length \\
\qquad\qquad \wedge\, n_{argstart} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“}0\text{”}))) \\
\qquad\qquad \wedge\, n_{argend} = \begin{cases} n_{strlen} & \text{if } \; \mathit{getArgValue}(args, \text{“}1\text{”}) = \textsf{undefined} \\ \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“}1\text{”}))) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, n_{start} = min(max(n_{argstart}, 0), n_{strlen}) \\
\qquad\qquad \wedge\, n_{end} = min(max(n_{argend}, 0), n_{strlen}) \\
\qquad\qquad \wedge\, s = native.string.slice(s_{this}, min(n_{start}, n_{end}), max(n_{start}, n_{end})) \quad \textit{// java, scala}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLowerCase}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLowerCase}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s = native.string.toLowerCase(s_{this}) \quad \textit{// java, scala}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLocaleLowerCase}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLocaleLowerCase}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s = native.string.toLowerCase(s_{this}) \quad \textit{// java, scala}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toUpperCase}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toUpperCase}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s = native.string.toUpperCase(s_{this}) \quad \textit{// java, scala}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLocaleUpperCase}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.toLocaleUpperCase}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s = native.string.toUpperCase(s_{this}) \quad \textit{// java, scala}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.trim}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \in \{\; \textsf{undefined}, \textsf{null} \;\} \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge\; H_1 = H[l_e \mapsto \mathit{NewExceptionObject}(\textsf{TypeError})] \\
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.trim}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where} \;\; v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \{\; \textsf{undefined}, \textsf{null} \;\} \wedge s_{this} = \mathit{ToString}(\mathit{ToPrimitive}(v)) \\
\qquad\qquad \wedge\, s = native.string.trim(s_{this}) \quad \textit{// java, scala}
\end{array}$$

11.1.11 Boolean

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Boolean.constructor}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where} \;\; v = \mathit{getArgValue}(args, \text{“}0\text{”}) \wedge l = \mathit{NewLocation}() \\
\qquad\qquad \wedge\, o = \mathit{NewBoolean}(\mathit{toBoolean}(\mathit{toPrimitive}(v))) \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.12 Boolean.prototype

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Boolean.prototype.toString”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \text{Boolean} \wedge (v \notin \text{Loc} \vee (v \in \text{Loc} \wedge H(v)(@class) \neq \text{“Boolean”})) \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\text{TypeError})] \\
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Boolean.prototype.toString”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge b = \begin{cases} v & \text{if } \ v \in \text{Boolean} \\ H(v)(@primitive) & \text{if } \ v \in \text{Loc} \wedge H(v)(@class) = \text{“Boolean”} \end{cases} \\
\qquad\qquad \wedge\, s = \begin{cases} \text{“true”} & \text{if } \ b = \text{true} \\ \text{“false”} & \text{if } \ b = \text{false} \end{cases}
\end{array}$$

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Boolean.prototype.valueOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \text{Boolean} \wedge (v \notin \text{Loc} \vee (v \in \text{Loc} \wedge H(v)(@class) \neq \text{“Boolean”})) \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\text{TypeError})] \\
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Boolean.prototype.valueOf”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto b]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge b = \begin{cases} v & \text{if } \ v \in \text{Boolean} \\ H(v)(@primitive) & \text{if } \ v \in \text{Loc} \wedge H(v)(@class) = \text{“Boolean”} \end{cases}
\end{array}$$

11.1.13 Number

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Number.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) < 1 \wedge l = \mathit{NewLocation}() \wedge o = \mathit{NewNumber}(0) \wedge H_1 = H\,[l \mapsto o] \\
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Number.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ v = \mathit{getArgValue}(args, \text{“0”}) \wedge \mathit{getArgValue}(args, \text{“}\mathit{length}\text{”}) \geq 1 \wedge l = \mathit{NewLocation}() \\
\qquad\qquad \wedge\, o = \mathit{NewNumber}(\mathit{toNumber}(\mathit{toPrimitive}(v))) \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.14 Number.prototype

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Number.prototype.valueOf”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v \notin \text{Number} \wedge (v \notin \text{Loc} \vee (v \in \text{Loc} \wedge H(v)(@class) \neq \text{“Number”})) \\
\qquad\qquad \wedge\, l_e = \mathit{newLocation}() \wedge \ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\text{TypeError})] \\
\mathcal{I}_{cp}[\![\text{BuiltintCall}(\text{“Number.prototype.valueOf”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n]], A) \\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge n = \begin{cases} v & \text{if } \ v \in \text{Number} \\ H(v)(@primitive) & \text{if } \ v \in \text{Loc} \wedge H(v)(@class) = \text{“Number”} \end{cases}
\end{array}$$

11.1.15 Math

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.abs}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 = \textsf{NaN} \\ \textsf{Inf} & \text{if } n_1 \in \{ \textsf{-Inf}, \textsf{Inf} \} \\ -n_1 & \text{if } v < 0 \\ n_1 & \text{if } 0 \leq v_1 \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.acos}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 \in \{ \textsf{NaN}, \textsf{Inf}, \textsf{-inf}, n \mid n < -1 \vee 1 < n \} \\ native.acos(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.asin}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 \in \{ \textsf{NaN}, \textsf{Inf}, \textsf{-inf}, n \mid n < -1 \vee 1 < n \} \\ native.asin(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.atan}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 = \textsf{NaN} \\ \frac{\pi}{2} & \text{if } n_1 = \textsf{Inf} \\ -\frac{\pi}{2} & \text{if } n_1 = \textsf{-Inf} \\ native.atan(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.atan2}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_3]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”}))) \wedge n_2 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“1”})))$$
$$n_3 = \begin{cases} \textsf{NaN} & \text{if } n_1 = \textsf{NaN} \vee n_2 = \textsf{NaN} \\ 0 & \text{if } n_1 = n \wedge n_2 = \textsf{Inf} \\ +0 & \text{if } n_1 > 0 \wedge n_1 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \wedge n_2 = \textsf{Inf} \\ \pi & \text{if } n_1 > 0 \wedge n_1 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \wedge n_2 = \textsf{-Inf} \\ -0 & \text{if } n_1 < 0 \wedge n_1 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \wedge n_2 = \textsf{Inf} \\ -\pi & \text{if } n_1 < 0 \wedge n_1 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \wedge n_2 = \textsf{-Inf} \\ \frac{\pi}{2} & \text{if } n_1 = \textsf{Inf} \wedge n_2 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \\ -\frac{\pi}{2} & \text{if } n_1 = \textsf{-Inf} \wedge n_2 \notin \{ \textsf{-Inf}, \textsf{Inf} \} \\ \frac{\pi}{4} & \text{if } n_1 = \textsf{Inf} \wedge n_2 = \textsf{Inf} \\ \frac{3\pi}{4} & \text{if } n_1 = \textsf{Inf} \wedge n_2 = \textsf{-Inf} \\ -\frac{\pi}{4} & \text{if } n_1 = \textsf{-Inf} \wedge n_2 = \textsf{Inf} \\ -\frac{3\pi}{4} & \text{if } n_1 = \textsf{-Inf} \wedge v_2 = \textsf{-Inf} \\ native.atan2(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.ceil}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} n_1 & \text{if } n_1 \in \{ \textsf{NaN}, \textsf{Inf}, \textsf{-Inf} \} \\ native.ceil(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.cos}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 \in \{\ \textsf{NaN}, \textsf{Inf}, \textsf{-Inf}\ \} \\ native.cos(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.exp}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} n_1 & \text{if } n_1 \in \{\ \textsf{NaN}, \textsf{Inf}\ \} \\ 0 & \text{if } n_1 = \textsf{-Inf} \\ native.exp(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.floor}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} n_1 & \text{if } n_1 \in \{\ \textsf{NaN, Inf, -Inf}\ \} \\ native.floor(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.log}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$$
$$\text{where } n_1 = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“0”})))$$
$$n_2 = \begin{cases} \textsf{NaN} & \text{if } n_1 \in \{\ \textsf{NaN}, \textsf{-Inf}, n \mid n < 0\ \} \\ \textsf{Inf} & \text{if } n_1 = \textsf{Inf} \\ \textsf{-Inf} & \text{if } n_1 = 0 \\ native.log(n_1) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.max}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n]], A)$$
$$\text{where } v_{len} = \textit{getArgValue}(args, \text{“}length\text{”}) \wedge n_i = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“}i\text{”})))$$
$$n = \begin{cases} \textsf{-Inf} & \text{if } v_{len} = 0 \\ \textsf{NaN} & \text{if } \exists i \in \{0, ..., v_{len}\} : n_i = \textsf{NaN} \\ n_{i_1} & \text{if } \exists i_1 \in \{0, ..., v_{len}\} : \forall i_2 \in \{0, ..., v_{len}\} : n_{i_1} \geq n_{i_2} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.min}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n]], A)$$
$$\text{where } v_{len} = \textit{getArgValue}(args, \text{“}length\text{”}) \wedge n_i = \textit{toNumber}(\textit{toPrimitive}(\textit{getArgValue}(args, \text{“}i\text{”})))$$
$$n = \begin{cases} \textsf{-Inf} & \text{if } v_{len} = 0 \\ \textsf{NaN} & \text{if } \exists i \in \{0, ..., v_{len}\} : n_i = \textsf{NaN} \\ n_{i_1} & \text{if } \exists i_1 \in \{0, ..., v_{len}\} : \forall i_2 \in \{0, ..., v_{len}\} : n_{i_1} \leq n_{i_2} \end{cases}$$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.pow}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_3]], A)$
where $n_1 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”}))) \wedge n_2 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“1”})))$

$$n_3 = \begin{cases} \mathsf{NaN} & \text{if } n_2 = \mathsf{NaN} \\ 1 & \text{if } n_2 = 0 \\ \mathsf{NaN} & \text{if } n_1 = \mathsf{NaN} \wedge n_2 \neq 0 \\ \mathsf{Inf} & \text{if } (n_1 > 1 \vee -n_1 > 1) \wedge n_2 = \mathsf{Inf} \\ 0 & \text{if } (n_1 > 1 \vee -n_1 > 1) \wedge n_2 = \mathsf{\text{-}Inf} \\ \mathsf{NaN} & \text{if } (n_1 == 1 \vee -n_1 == 1) \wedge v_{y1} \in \{\ \mathsf{Inf, \text{-}Inf}\ \} \\ 0 & \text{if } (n_1 < 1 \vee -n_1 < 1) \wedge v_{y1} = \mathsf{Inf} \\ \mathsf{Inf} & \text{if } (n_1 < 1 \vee -n_1 < 1) \wedge v_{y1} = \mathsf{\text{-}Inf} \\ \mathsf{Inf} & \text{if } n_1 = \mathsf{Inf} \wedge n_2 > 0 \\ 0 & \text{if } n_1 = \mathsf{Inf} \wedge n_2 < 0 \\ \mathsf{\text{-}Inf} & \text{if } n_1 = \mathsf{\text{-}Inf} \wedge n_2 > 0 \wedge n_2 \neq \mathsf{Inf} \wedge n_2\%2 = 1 \\ \mathsf{Inf} & \text{if } n_1 = \mathsf{\text{-}Inf} \wedge n_2 > 0 \wedge n_2 \neq \mathsf{Inf} \wedge n_2\%2 = 0 \\ 0 & \text{if } n_1 = \mathsf{\text{-}Inf} \wedge v_{y1} < 0 \\ 0 & \text{if } n_1 = 0 \wedge n_2 > 0 \\ \mathsf{Inf} & \text{if } n_1 = 0 \wedge n_2 < 0 \\ \mathsf{NaN} & \text{if } v_{x1} < 0 \wedge v_{x1} \neq \mathsf{\text{-}Inf} \wedge v_{y1} \neq \{\ \mathsf{Inf, \text{-}Inf}\ \} \wedge \neg isInt(v_{y1}) \\ native.pow(n_1) & \text{otherwise} \end{cases}$$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.random}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n]], A)$
where $n = native.random()$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.round}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$
where $n_1 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”})))$

$$n_2 = \begin{cases} n_1 & \text{if } n_1 \in \{\ \mathsf{NaN, Inf, \text{-}Inf}\ \} \\ native.round(n_1) & \text{otherwise} \end{cases}$$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.sin}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$
where $n_1 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”})))$

$$n_2 = \begin{cases} \mathsf{NaN} & \text{if } n_1 \in \{\ \mathsf{NaN, Inf, \text{-}Inf}\ \} \\ native.sin(n_1) & \text{otherwise} \end{cases}$$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.sqrt}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$
where $n_1 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”})))$

$$n_2 = \begin{cases} n_1 & \text{if } n_1 \in \{\ \mathsf{NaN, Inf}\ \} \\ \mathsf{NaN} & \text{if } n_1 < 0 \vee n_1 = \mathsf{\text{-}Inf} \\ native.sqrt(n_1) & \text{otherwise} \end{cases}$$

$\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.tan}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto n_2]], A)$
where $n_1 = \mathit{toNumber}(\mathit{toPrimitive}(\mathit{getArgValue}(args, \text{“0”})))$

$$n_2 = \begin{cases} \mathsf{NaN} & \text{if } n_1 \in \{\ \mathsf{NaN, Inf, \text{-}Inf}\ \} \\ native.tan(n_1) & \text{otherwise} \end{cases}$$

11.1.16 Date

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) = 0\\
&\qquad\qquad \wedge\, n = native.Calendar.getInstance().getTimeInMillis() \quad \textit{// java, scala}\\
&\qquad\qquad \wedge\, o = \mathit{NewDateObject}(n) \wedge l = \mathit{newLocation}() \wedge H_1 = H[l \mapsto o]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) = 1 \wedge v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \in \mathsf{String}\\
&\qquad\qquad \wedge\, o = \mathit{NewDateObject}(v) \wedge l = \mathit{newLocation}() \wedge H_1 = H[l \mapsto o]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) = 1 \wedge v = \mathit{getArgValue}(args, \text{“0”}) \wedge v \notin \mathsf{String}\\
&\qquad\qquad \wedge\, n = \mathit{ToNumber}(\mathit{ToPrimitive}(v)) \wedge o = \mathit{NewDateObject}(\mathit{TimeClip}(n)) \wedge l = \mathit{newLocation}() \wedge H_1 = H[l \mapsto o]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
&\quad \text{where } \ n_{arglen} = \mathit{getArgValue}(args, \text{“}length\text{”}) \wedge n_{arglen} > 1\\
&\qquad\qquad \wedge\, n_1 = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“0”})))\\
&\qquad\qquad \wedge\, n_{year} = \begin{cases} n_1 + 1900 & \text{if } \ n_1 \neq \mathsf{NaN} \wedge 0 \leq n_1 \wedge n_1 \leq 99 \\ n_1 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n_{month} = \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“1”})))\\
&\qquad\qquad \wedge\, n_{date} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“2”}))) & \text{if } \ n_{arglen} > 2 \\ 1 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n_{hour} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“3”}))) & \text{if } \ n_{arglen} > 3 \\ 0 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n_{min} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“4”}))) & \text{if } \ n_{arglen} > 4 \\ 0 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n_{sec} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“5”}))) & \text{if } \ n_{arglen} > 5 \\ 0 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n_{ms} = \begin{cases} \mathit{ToNumber}(\mathit{ToPrimitive}(\mathit{getArgValue}(args, \text{“6”}))) & \text{if } \ n_{arglen} > 6 \\ 0 & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\, n = \mathit{MakeDate}(\mathit{MakeDay}(n_{year}, n_{month}, n_{date}), \mathit{MakeTime}(n_{hour}, n_{min}, n_{sec}, n_{ms}))\\
&\qquad\qquad \wedge\, o = \mathit{NewDateObject}(v) \wedge l = \mathit{newLocation}() \wedge H_1 = H[l \mapsto o]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.now”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto n]], A)\\
&\quad \text{where } \ n = native.Calendar.getInstance().getTimeInMillis() \quad \textit{// java, scala}
\end{aligned}$$

11.1.17 Date.prototype

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toDateString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toTimeString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleDateString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleTimeString}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive) \land s = (native.util.Date(v_2)).toString()\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.valueOf}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto 1(\#temp)[@return \mapsto v_2]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive)\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getTime}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v_2]], A)\\&\quad \text{where } \ v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_2 = H(v_1)(@primitive)\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getFullYear}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\&\quad \text{where } \ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_{time} = H(v_{this})(@primitive)\\&\qquad\qquad v = \begin{cases}\mathsf{NaN} & \text{if } \ v_{time} = \mathsf{NaN}\\ native.Calendar(v_{time}).get(YEAR) & \text{otherwise}\end{cases}\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCFullYear}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\&\quad \text{where } \ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_{time} = H(v_{this})(@primitive)\\&\qquad\qquad v = \begin{cases}\mathsf{NaN} & \text{if } \ v_{time} = \mathsf{NaN}\\ native.Calendar(v_{time}).get(YEAR) & \text{otherwise}\end{cases}\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getMonth}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\&\quad \text{where } \ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_{time} = H(v_{this})(@primitive)\\&\qquad\qquad v = \begin{cases}\mathsf{NaN} & \text{if } \ v_{time} = \mathsf{NaN}\\ native.Calendar(v_{time}).get(MONTH) & \text{otherwise}\end{cases}\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCMonth}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\&\quad \text{where } \ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \land v_{time} = H(v_{this})(@primitive)\\&\qquad\qquad v = \begin{cases}\mathsf{NaN} & \text{if } \ v_{time} = \mathsf{NaN}\\ native.Calendar(v_{time}).get(MONTH) & \text{otherwise}\end{cases}\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getDate}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{DAY\_OF\_MONTH}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCDate}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{DAY\_OF\_MONTH}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getDay}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{DAY\_OF\_WEEK}) - 1 & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCDay}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{DAY\_OF\_WEEK}) - 1 & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getHours}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{HOURS}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCHours}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{HOURS}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getMinutes}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{MINUTE}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCMinutes}\text{”}, args)]\!](H,A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H,A) \wedge v_{time} = H(v_{this})(@primitive)\\
&\qquad\quad v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{MINUTE}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getMinutes}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_time).get(\mathit{MINUTE}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCMinutes}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{MINUTE}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getSeconds}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_time).get(\mathit{SECOND}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCSeconds}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{SECOND}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getMilliseconds}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_time).get(\mathit{MILLISECOND}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCMilliseconds}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{MILLISECOND}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getTimeZoneOffset}\text{”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive)$

$$v = \begin{cases} \textsf{NaN} & \text{if } v_{time} = \textsf{NaN} \\ native.Calendar(v_{time}).get(\mathit{ZONE\_OFFSET}) & \text{otherwise} \end{cases}$$

$$\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setTime}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)$$

where $v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))$

$\wedge\ H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMilliseconds}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{MILLISECOND}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMilliseconds}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{MILLISECOND}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setSeconds}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{SECOND}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCSeconds}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{SECOND}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMinutes}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{MINUTE}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMinutes}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{MINUTE}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setHours}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{HOURS}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCHours}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\&\quad\text{where }\ v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\&\qquad\qquad \wedge\, v = native.Calendar(v_{time}).set(\mathit{HOURS}, v_{arg}).getTimeInMillis()\\&\qquad\qquad \wedge\, H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setDate}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{DAY\_OF\_MONTH}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCDate}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{DAY\_OF\_MONTH}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMonth}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{MONTH}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMonth}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{MONTH}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setFullYear}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{YEAR}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCFullYear}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto v]], A)\\
&\quad \text{where } v_{this} = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_{time} = H(v_{this})(@primitive) \wedge v_{arg} = \mathit{ToNumber}(\mathit{getArgValue}(args, \text{“0”}))\\
&\qquad\qquad \wedge v = native.Calendar(v_{time}).set(\mathit{YEAR}, v_{arg}).getTimeInMillis()\\
&\qquad\qquad \wedge H_1 = H[v_{this} \mapsto H(v_{this})[@ \mapsto v]]
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.toUTCString}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto s]], A)\\
&\quad \text{where } v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_2 = H(v_1)(@primitive) \wedge s = (\mathit{native.util.Date}(v_2)).\mathit{toString}()
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.toISOString}\text{”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto s]], A)\\
&\quad \text{where } v_1 = \mathcal{V}_{cp}[\![\texttt{this}]\!](H, A) \wedge v_2 = H(v_1)(@primitive) \wedge s = (\mathit{native.util.Date}(v_2)).\mathit{toString}()
\end{aligned}$$

11.1.18 RegExp

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp\text{”}, args)]\!](H, A) = (H_1, A) \\
\quad \text{if } \ v_1 \in \mathsf{Loc} \ \wedge \ H(v_1)(@class) = \text{“}RegExp\text{”} \ \wedge \ v_2 = \mathsf{undefined} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(args, \text{“}0\text{”}) \ \wedge \ v_2 = \mathit{getArgValue}(args, \text{“}1\text{”}) \\
\qquad\qquad \wedge\, H_1 = H[\#temp \mapsto H(\#temp)[@return \mapsto v_1]] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp\text{”}, args)]\!](H, A) = \mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp.constructor\text{”}, args)]\!](H, A)
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp.constructor\text{”}, args)]\!](H, A) = (H_2, A) \\
\quad \text{if } \ v_1 \in Loc \ \wedge \ H(v_1)(@class) = \text{“}RegExp\text{”} \ \wedge \ v_2 = \mathsf{undefined} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(args, \text{“}0\text{”}) \ \wedge \ v_2 = \mathit{getArgValue}(args, \text{“}1\text{”}) \\
\qquad\qquad \wedge\, v_{source} = H(v_1)(\text{“}source\text{”}) \\
\qquad\qquad \wedge\, b_g = H(v_1)(\text{“}global\text{”}) \\
\qquad\qquad \wedge\, b_i = H(v_1)(\text{“}ignoreCase\text{”}) \\
\qquad\qquad \wedge\, b_m = H(v_1)(\text{“}multiline\text{”}) \\
\qquad\qquad \wedge\, mid = H(v_1)(@matcher) \\
\qquad\qquad \wedge\, o = \underline{\mathsf{NewRegExp}}(v_{source}, b_g, b_i, b_m, mid) \\
\qquad\qquad \wedge\, l = \mathit{NewLocation}() \\
\qquad\qquad \wedge\, H_1 = H[l \mapsto o] \\
\qquad\qquad \wedge\, H_2 = H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp.constructor\text{”}, args)]\!](H, A) = (H_1, A) \\
\quad \text{if } \ v_1 \in Loc \ \wedge \ H(v_1)(@class) = \text{“}RegExp\text{”} \ \wedge \ v_2 \neq \mathsf{undefined} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(args, \text{“}0\text{”}) \ \wedge \ v_2 = \mathit{getArgValue}(args, \text{“}1\text{”}) \\
\qquad\qquad \wedge\, H_1 = \underline{\mathsf{RaiseException}}(H, \mathsf{TypeError}) \\
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp.constructor\text{”}, args)]\!](H, A) = (H_1, A) \\
\quad \text{if } \ v_1 \notin Loc \ \wedge \ v_{match} \in \mathsf{MatcherId} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(args, \text{“}0\text{”}) \ \wedge \ v_2 = \mathit{getArgValue}(args, \text{“}1\text{”}) \\
\qquad\qquad \wedge\, v_P = \begin{cases} \text{“”} & \text{if } \ v_1 = \mathsf{undefined} \\ \underline{\mathsf{toString}}(v_1) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, v_F = \begin{cases} \text{“”} & \text{if } \ v_2 = \mathsf{undefined} \\ \underline{\mathsf{toString}}(v_2) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, v_{match} = \underline{\mathsf{RegExpParser}}(v_P, v_F) \\
\qquad\qquad \wedge\, o = \underline{\mathsf{NewRegExp}}(v_{source}, b_g, b_i, b_m, v_{match}) \\
\qquad\qquad \wedge\, l = \mathit{NewLocation}() \\
\qquad\qquad \wedge\, H_1 = H[l \mapsto o] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}RegExp.constructor\text{”}, args)]\!](H, A) = (H_1, A) \\
\quad \text{if } \ v_1 \notin Loc \ \wedge \ exc \in \mathsf{Exception} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(args, \text{“}0\text{”}) \ \wedge \ v_2 = \mathit{getArgValue}(args, \text{“}1\text{”}) \\
\qquad\qquad \wedge\, v_P = \begin{cases} \text{“”} & \text{if } \ v_1 = \mathsf{undefined} \\ \underline{\mathsf{toString}}(v_1) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, v_F = \begin{cases} \text{“”} & \text{if } \ v_2 = \mathsf{undefined} \\ \underline{\mathsf{toString}}(v_2) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, exc = \underline{\mathsf{RegExpParser}}(v_P, v_F) \\
\qquad\qquad \wedge\, H_1 = \underline{\mathsf{RaiseException}}(H, exc)
\end{array}
$$

11.1.19 RegExp.prototype

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}\mathit{RegExp.prototype.exec}\text{”}, \mathit{args})]\!](H, A) = (H_3, A) \\
\quad \text{if } \ v_{exec} \in \mathit{Object} \times \mathsf{Int} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(\mathit{args}, \text{“}0\text{”}) \\
\qquad\qquad \wedge\, v_R = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \ \wedge \ v_S = \underline{\mathsf{toString}}(v_1) \\
\qquad\qquad \wedge\, v_{lastIndex} = H(v_R)(\text{“}\mathit{lastIndex}\text{”}) \ \wedge \ i = \underline{\mathsf{ToInteger}}(v_{lastIndex}) \\
\qquad\qquad \wedge\, v_g = H(v_R)(\text{“}\mathit{global}\text{”}) \\
\qquad\qquad \wedge\, v_i = H(v_R)(\text{“}\mathit{ignoreCase}\text{”}) \\
\qquad\qquad \wedge\, v_m = H(v_R)(\text{“}\mathit{multiline}\text{”}) \\
\qquad\qquad \wedge\, \mathit{mid} = H(v_R)(@\mathit{matcher}) \\
\qquad\qquad \wedge\, v_{exec} = \underline{\mathsf{exec}}(\mathit{mid}, v_S, i, v_g, v_i, v_m) \\
\qquad\qquad \wedge\, l = \mathit{NewLocation}() \\
\qquad\qquad \wedge\, H_1 = H[l \mapsto v_{exec}.1] \\
\qquad\qquad \wedge\, H_2 = \begin{cases} H_1[v_R \mapsto H_1(v_R)[\text{“}\mathit{lastIndex}\text{”} \mapsto v_{exec}.2]] & \text{if } \ v_g = \mathsf{true} \\ H_1 & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, H_3 = H_2[\#\mathit{temp} \mapsto H_1(\#\mathit{temp})[@\mathit{return} \mapsto l]]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}\mathit{RegExp.prototype.exec}\text{”}, \mathit{args})]\!](H, A) = (H_1, A) \\
\quad \text{if } \ v_{exec} = \mathsf{null} \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(\mathit{args}, \text{“}0\text{”}) \\
\qquad\qquad \wedge\, v_R = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \ \wedge \ v_S = \underline{\mathsf{toString}}(v_1) \\
\qquad\qquad \wedge\, v_{lastIndex} = H(v_R)(\text{“}\mathit{lastIndex}\text{”}) \ \wedge \ i = \underline{\mathsf{ToInteger}}(v_{lastIndex}) \\
\qquad\qquad \wedge\, v_g = H(v_R)(\text{“}\mathit{global}\text{”}) \\
\qquad\qquad \wedge\, v_i = H(v_R)(\text{“}\mathit{ignoreCase}\text{”}) \\
\qquad\qquad \wedge\, v_m = H(v_R)(\text{“}\mathit{multiline}\text{”}) \\
\qquad\qquad \wedge\, \mathit{mid} = H(v_R)(@\mathit{matcher}) \\
\qquad\qquad \wedge\, v_{exec} = \underline{\mathsf{exec}}(\mathit{mid}, v_S, i, v_g, v_i, v_m) \\
\qquad\qquad \wedge\, H_1 = H[\#\mathit{temp} \mapsto H_1(\#\mathit{temp})[@\mathit{return} \mapsto \mathsf{null}]]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}\mathit{RegExp.prototype.exec}\text{”}, \mathit{args})]\!](H, A) = (H_1, A) \\
\quad \text{where } \ v_1 = \mathit{getArgValue}(\mathit{args}, \text{“}0\text{”}) \\
\qquad\qquad \wedge\, v_R = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \ \wedge \ v_S = \underline{\mathsf{toString}}(v_1) \\
\qquad\qquad \wedge\, v_{lastIndex} = H(v_R)(\text{“}\mathit{lastIndex}\text{”}) \ \wedge \ i = \underline{\mathsf{ToInteger}}(v_{lastIndex}) \\
\qquad\qquad \wedge\, v_g = H(v_R)(\text{“}\mathit{global}\text{”}) \\
\qquad\qquad \wedge\, v_i = H(v_R)(\text{“}\mathit{ignoreCase}\text{”}) \\
\qquad\qquad \wedge\, v_m = H(v_R)(\text{“}\mathit{multiline}\text{”}) \\
\qquad\qquad \wedge\, \mathit{mid} = H(v_R)(@\mathit{matcher}) \\
\qquad\qquad \wedge\, v_{exec} = \underline{\mathsf{exec}}(\mathit{mid}, v_S, i, v_g, v_i, v_m) \\
\qquad\qquad \wedge\, v_{return} = \begin{cases} \mathsf{true} & \text{if } \ v_{exec} \neq \mathsf{null} \\ \mathsf{false} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, H_1 = H[\#\mathit{temp} \mapsto H_1(\#\mathit{temp})[@\mathit{return} \mapsto v_{return}]]
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltinCall}(\text{“}\mathit{RegExp.prototype.toString}\text{”}, \mathit{args})]\!](H, A) = (H_1, A) \\
\quad \text{where } \ v_R = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \\
\qquad\qquad \wedge\, v_{src} = H(v_R)(\text{“}\mathit{source}\text{”}) \\
\qquad\qquad \wedge\, v_g = H(v_R)(\text{“}\mathit{global}\text{”}) \\
\qquad\qquad \wedge\, v_i = H(v_R)(\text{“}\mathit{ignoreCase}\text{”}) \\
\qquad\qquad \wedge\, v_m = H(v_R)(\text{“}\mathit{multiline}\text{”}) \\
\qquad\qquad \wedge\, s_1 = \begin{cases} \text{“}g\text{”} & \text{if } \ v_g = \mathsf{true} \\ \text{“”} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, s_2 = \begin{cases} \text{“}i\text{”} & \text{if } \ v_i = \mathsf{true} \\ \text{“”} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, s_3 = \begin{cases} \text{“}m\text{”} & \text{if } \ v_m = \mathsf{true} \\ \text{“”} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, v_{return} = \text{“}/\text{”} + v_{src} + \text{“}/\text{”} + s_1 + s_2 + s_3 \\
\qquad\qquad \wedge\, H_1 = H[\#\mathit{temp} \mapsto H_1(\#\mathit{temp})[@\mathit{return} \mapsto v_{return}]]
\end{array}
$$

11.1.20 Error

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) < 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#ErrorProto,\\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o]\\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ v = \mathit{getArgValue}(args, \text{“0”}) \wedge \mathit{getArgValue}(args, \text{“}length\text{”}) \geq 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#ErrorProto,\\ @extensible \mapsto \mathsf{true}, & message \mapsto \mathit{toString}(\mathit{toPrimitive}(v)) \end{array} \right\}\\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.21 Error.prototype

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.prototype.toString”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@exception \mapsto l_e]], A)\\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \wedge v \notin \mathsf{Loc} \wedge l_e = \mathit{newLocation}() \wedge \ H_2 = H_1[l_e \mapsto \mathit{NewExceptionObject}(\mathsf{TypeError})]\\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.prototype.toString”}, args)]\!](H, A) = (H[\#temp \mapsto H(\#temp)[@return \mapsto s_3]], A)\\
\quad \text{where } \ v = \mathcal{V}_{cp}[\![\mathtt{this}]\!](H, A) \wedge v \in \mathsf{Loc}\\
\qquad\quad s_1 = \left\{ \begin{array}{ll} H(v)(\text{“}name\text{”}) & \text{if } \ H(v)(\text{“}name\text{”}) \neq \mathsf{undefined}\\ \text{“}Error\text{”} & \text{otherwise} \end{array} \right.\\
\qquad\quad s_2 = \left\{ \begin{array}{ll} \mathit{toString}(\mathit{toPrimitive}(H(v)(\text{“}message\text{”}))) & \text{if } \ H(v)(\text{“}message\text{”}) \neq \mathsf{undefined}\\ \text{“”} & \text{otherwise} \end{array} \right.\\
\qquad\quad s_3 = \left\{ \begin{array}{ll} s_2 & \text{if } \ s_1 = \text{“”}\\ s_1 & \text{if } \ s_2 = \text{“”}\\ s_1 + \text{“ : ”} + s_2 & \text{otherwise} \end{array} \right.
\end{array}$$

11.1.22 EvalError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) < 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#EvalErrorProto,\\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o]\\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ v = \mathit{getArgValue}(args, \text{“0”}) \wedge \mathit{getArgValue}(args, \text{“}length\text{”}) \geq 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#EvalErrorProto,\\ @extensible \mapsto \mathsf{true}, & message \mapsto \mathit{toString}(\mathit{toPrimitive}(v)) \end{array} \right\}\\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.23 RangeError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ \mathit{getArgValue}(args, \text{“}length\text{”}) < 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#RangeErrorProto,\\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o]\\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A)\\
\quad \text{where } \ v = \mathit{getArgValue}(args, \text{“0”}) \wedge \mathit{getArgValue}(args, \text{“}length\text{”}) \geq 1 \wedge l = \mathit{NewLocation}()\\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#RangeErrorProto,\\ @extensible \mapsto \mathsf{true}, & message \mapsto \mathit{toString}(\mathit{toPrimitive}(v)) \end{array} \right\}\\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.24 ReferenceError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) < 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#ReferenceErrorProto, \\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ v = \textit{getArgValue}(args, \text{“0”}) \wedge \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) \geq 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#ReferenceErrorProto, \\ @extensible \mapsto \mathsf{true}, & message \mapsto \textit{toString}(\textit{toPrimitive}(v)) \end{array} \right\} \\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.25 SyntaxError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) < 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#SyntaxErrorProto, \\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ v = \textit{getArgValue}(args, \text{“0”}) \wedge \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) \geq 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#SyntaxErrorProto, \\ @extensible \mapsto \mathsf{true}, & message \mapsto \textit{toString}(\textit{toPrimitive}(v)) \end{array} \right\} \\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.26 TypeError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) < 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#TypeErrorProto, \\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ v = \textit{getArgValue}(args, \text{“0”}) \wedge \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) \geq 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#TypeErrorProto, \\ @extensible \mapsto \mathsf{true}, & message \mapsto \textit{toString}(\textit{toPrimitive}(v)) \end{array} \right\} \\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.1.27 URIError

$$\begin{array}{l}
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) < 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#URIErrorProto, \\ @extensible \mapsto \mathsf{true} & \end{array} \right\} \wedge H_1 = H\,[l \mapsto o] \\
\mathcal{I}_{cp}[\![\mathsf{BuiltintCall}(\text{“Error.constructor”}, args)]\!](H, A) = (H_1[\#temp \mapsto H_1(\#temp)[@return \mapsto l]], A) \\
\quad \text{where } \ v = \textit{getArgValue}(args, \text{“0”}) \wedge \textit{getArgValue}(args, \text{“}\textit{length}\text{”}) \geq 1 \wedge l = \textit{NewLocation}() \\
\qquad\quad \wedge\, o = \left\{ \begin{array}{ll} @class \mapsto \text{“}Error\text{”}, & @proto \mapsto \#URIErrorProto, \\ @extensible \mapsto \mathsf{true}, & message \mapsto \textit{toString}(\textit{toPrimitive}(v)) \end{array} \right\} \\
\qquad\quad \wedge H_1 = H\,[l \mapsto o]
\end{array}$$

11.2 Abstract Semantics

11.2.1 Helper

$$
\begin{array}{ll}
\underline{\widehat{\textsf{DefineProperty}}} & : \widehat{\textsf{Heap}} \times \widehat{\textsf{Loc}} \times \widehat{\textsf{String}} \times \widehat{\textsf{Loc}} \rightarrow \widehat{\textsf{Heap}} \\
& \underline{\widehat{\textsf{ToPropertyDescriptor}}}(\hat{H}, \hat{l}_1, \hat{s}, \hat{l}_2) = \hat{H}_1 \\
& \quad \text{where} \;\; \hat{H}_1 = \hat{H}[\hat{l}_1 \mapsto \hat{H}(\hat{l}_1)[\hat{s} \mapsto \langle \hat{v}_{val}, \hat{b}_w, \hat{b}_e, \hat{b}_c \rangle]] \\
& \qquad\qquad \textit{// skip when AccessorDescriptor} \\
& \qquad\qquad \wedge\, \hat{v}_{val} = \underline{\widehat{\textsf{Proto}}}(\hat{H}, \hat{l}_2, \text{“}\widehat{value}\text{”}) \\
& \qquad\qquad \wedge\, \hat{b}_w = \underline{\widehat{\textsf{toBoolean}}}(\underline{\widehat{\textsf{Proto}}}(\hat{H}, \hat{l}_2, \text{“}\widehat{writable}\text{”})) \\
& \qquad\qquad \wedge\, \hat{b}_e = \underline{\widehat{\textsf{toBoolean}}}(\underline{\widehat{\textsf{Proto}}}(\hat{H}, \hat{l}_2, \text{“}\widehat{enumerable}\text{”})) \\
& \qquad\qquad \wedge\, \hat{b}_c = \underline{\widehat{\textsf{toBoolean}}}(\underline{\widehat{\textsf{Proto}}}(\hat{H}, \hat{l}_2, \text{“}\widehat{configurable}\text{”}))
\end{array}
$$

$$
\begin{array}{ll}
\underline{\widehat{\textsf{TimeClip}}} & : \widehat{\textsf{Number}} \rightarrow \widehat{\textsf{Number}} \\
& \underline{\widehat{\textsf{TimeClip}}}(\hat{n}) = \begin{cases}
\top_{Number} & \text{if } \; \textsf{UInt} \sqsubseteq \hat{n} \vee \textsf{NUInt} \sqsubseteq \hat{n} \\
\textsf{NaN} & \text{if } \; \textsf{-inf} = \hat{n} \vee \textsf{inf} = \hat{n} \vee \textsf{NaN} = \hat{n} \\
\textsf{NaN} & \text{if } \; \textsf{NUIntSingle}(n) = \hat{n} \wedge abs(n) > 8.64 \times 10^{15} \\
\underline{\widehat{\textsf{toInteger}}}(\hat{n}) & \text{if } \; \textsf{NUIntSingle}(n) = \hat{n} \wedge abs(n) \leq 8.64 \times 10^{15} \\
\bot_{Number} & \text{if } \; \hat{n} \sqsubseteq \bot_{Number}
\end{cases}
\end{array}
$$

11.2.2 Global

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{isNaN}\text{”}, args)]\!]((\hat{H}_1, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}), \hat{S}) \\
\quad \text{where} \;\; \hat{n} = \underline{\widehat{\textsf{toNumber}}}(\underline{\widehat{\textsf{toPrimitive}}}(\underline{\widehat{\textsf{getArgValue}}}(\text{“}0\text{”}))) \\
\qquad\qquad \wedge\, \hat{b} = \begin{cases}
\widehat{\textsf{true}} & \text{if } \; \widehat{\textsf{NaN}} = \hat{n} \\
\widehat{\textsf{false}} & \text{if } \; \widehat{\textsf{NaN}} \neq \hat{n} \wedge \widehat{\textsf{NaN}} \not\sqsubseteq \hat{n} \\
\top_{bool} & \text{if } \; \widehat{\textsf{NaN}} \neq \hat{n} \wedge \widehat{\textsf{NaN}} \sqsubseteq \hat{n} \\
\bot_{bool} & \text{otherwise}
\end{cases} \\
\qquad\qquad \wedge\, \hat{H}_1 = \underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{b}))
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{isFinite}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}), \hat{S}) \\
\quad \text{where} \;\; \hat{n} = \underline{\widehat{\textsf{toNumber}}}(\underline{\widehat{\textsf{toPrimitive}}}(\underline{\widehat{\textsf{getArgValue}}}(\text{“}0\text{”}))) \\
\qquad\qquad \wedge\, \hat{b} = \begin{cases}
\widehat{\textsf{false}} & \text{if } \; \widehat{\textsf{NaN}} = \hat{n} \vee \widehat{\textsf{inf}} = \hat{n} \vee \widehat{\textsf{-inf}} = \hat{n} \\
\top_{bool} & \text{if } \; \widehat{\textsf{NaN}} \sqsubseteq \hat{n} \vee \widehat{\textsf{inf}} \sqsubseteq \hat{n} \vee \widehat{\textsf{-inf}} \sqsubseteq \hat{n} \\
\widehat{\textsf{true}} & \text{if } \; \widehat{\textsf{NaN}} \not\sqsubseteq \hat{n} \wedge \widehat{\textsf{inf}} \not\sqsubseteq \hat{n} \wedge \widehat{\textsf{-inf}} \not\sqsubseteq \hat{n} \\
\bot_{bool} & \text{otherwise}
\end{cases} \\
\qquad\qquad \wedge\, \hat{H}_1 = \underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{b}))
\end{array}
$$

11.2.3 Object

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.constructor}\text{”}, args)_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_8,\hat{C}_8),\hat{S}_1) \\
\quad \text{where } \hat{v} = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}0\text{”}) \\
\qquad \wedge (\hat{v}_1,\hat{H}_1,\hat{C}_1) = \begin{cases} (Value(\hat{v}.2),\hat{H},\hat{C}) & \text{if } \hat{v}.2 \neq \{\} \\ (\bot_{value},\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
\qquad \wedge (\hat{v}_2,\hat{H}_2,\hat{C}_2,\hat{es}_2) = \begin{cases} (\hat{v}_4,\hat{H}_4,\hat{C}_4,\hat{es}) & \text{if } \hat{v}.1.3 \not\sqsubseteq \bot_{bool} \vee \hat{v}.1.4 \not\sqsubseteq \bot_{num} \vee \hat{v}.1.5 \not\sqsubseteq \bot_{string} \\ (\bot_{value},\bot_{heap},\bot_{context},\{\}) & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{v}_{prim} = Value(PValue(\bot_{undef},\bot_{null},\hat{v}.1.3,\hat{v}.1.4,\hat{v}.1.5)) \\
\qquad \wedge (\hat{v}_4,\hat{H}_4,\hat{C}_4,\hat{es}) = \underline{\widehat{\mathsf{toObject}}}(\hat{H},\hat{C},\hat{v}_{prim},\hat{a}_{new}) \\
\qquad \wedge (\hat{v}_3,\hat{H}_3,\hat{C}_3) = \begin{cases} (Value(\{\hat{l}_R\}),\hat{H}_6,\hat{C}_5) & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{null} \\ (\bot_{value},\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_5,\hat{C}_5) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction} \\
\qquad \wedge \hat{H}_6 = \underline{\widehat{\mathsf{allocObject}}}(\hat{H}_5,\{\ \widehat{\#ObjProto}_R\ \},\hat{l}_R) \\
\qquad \wedge (\hat{v}_7,\hat{H}_7,\hat{C}_7) = (\hat{v}_1 \sqcup \hat{v}_2 \sqcup \hat{v}_3, \hat{H}_1 \sqcup \hat{H}_2 \sqcup \hat{H}_3, \hat{C}_1 \sqcup \hat{C}_2 \sqcup \hat{C}_3) \\
\qquad \wedge (\hat{H}_8,\hat{C}_8) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_7,\hat{v}_7),\hat{C}_7) & \text{if } \hat{v}_7 \not\sqsubseteq \bot_{value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}_2,\hat{C}_2,\hat{es}_2)
\end{array}
$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Object.getPrototypeOf}\text{”}, args)\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \underline{\mathsf{get}\widehat{\mathsf{ArgVal}}\mathsf{ue}}(\hat{H},\hat{C},\text{“}0\text{”})\\
& \wedge \hat{v}_2 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}_1.2} \hat{H}(\hat{l})(@\hat{proto}).1.1.1\\
& \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\hat{v}_2),\hat{C}) & \text{if } \ \hat{v}_2 \not\sqsubseteq \bot_{value}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
& \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)_{\hat{a}_{new}}\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_7,\hat{C}_7),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \underline{\mathsf{get}\widehat{\mathsf{ArgVal}}\mathsf{ue}}(\hat{H},\hat{C},\text{“}0\text{”}) \wedge \hat{s}_{prop} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\underline{\mathsf{get}\widehat{\mathsf{ArgVal}}\mathsf{ue}}(\hat{H},\hat{C},\text{“}1\text{”})))\\
& \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge \hat{ov} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}_1.2} \hat{H}(\hat{l})(\hat{s}_{prop}).1.1\\
& \wedge (\hat{v}_2,\hat{H}_2,\hat{C}_2) = \begin{cases} (Value(PValue(\top_{undef})),\hat{H},\hat{C}) & \text{if } \ \top_{undef} \sqsubseteq \hat{ov}.1.1.1\\ (\bot_{value},\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
& \wedge (\hat{v}_3,\hat{H}_3,\hat{C}_3) = \begin{cases} (Value(\{\hat{l}_R\}),\hat{H}_5,\hat{C}_4) & \text{if } \ Value(Pvalue(\bot_{undef},\hat{ov}.1.1.2,\hat{ov}.1.1.3,\hat{ov}.1.1.4,\hat{ov}.1.1.5),\hat{v}.1.2)\\ & \qquad\qquad \not\sqsubseteq \bot_{value}\\ (\bot_{value},\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
& \wedge \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_4,\hat{C}_4) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
& \wedge \hat{o}_{new} = \underline{\mathsf{New}\widehat{\mathsf{Obj}}\mathsf{ect}}(\#Obj\hat{P}roto_R)\\
& \wedge \hat{o}_1 = \begin{cases} \hat{o}_{new}\left[\begin{array}{l} \text{“}va\hat{l}ue\text{”} \mapsto \langle \hat{v}_2.1.1.1, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle\\ \text{“}wri\hat{t}able\text{”} \mapsto \langle \hat{v}_2.1.1.2, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle\end{array}\right] & \text{if } \ \textit{IsDataDescriptor}(H(v),s)\\ \hat{o}_{new} & \text{otherwise}\textit{// skip when AccessorDescriptor}\end{cases}\\
& \wedge \hat{o}_2 = \hat{o}_1\left[\begin{array}{l} \text{“}enum\hat{e}rable\text{”} \mapsto \langle \hat{v}_2.1.1.3, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle\\ \text{“}confi\hat{g}urable\text{”} \mapsto \langle \hat{v}_2.1.1.4, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle\end{array}\right]\\
& \wedge \hat{H}_5 = \hat{H}_4[\hat{l}_R \mapsto \hat{o}_2]\\
& \wedge \hat{H}_6 = \hat{H}_2 \sqcup \hat{H}_3\\
& \wedge (\hat{H}_7,\hat{C}_7) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_6,\hat{v}_2 \sqcup \hat{v}_3),\hat{C}_2 \sqcup \hat{C}_3) & \text{if } \ \hat{v}_2 \not\sqsubseteq \bot_{value}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
& \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyNames}\text{”}, args)_{\hat{a}_{new}}\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \underline{\mathsf{get}\widehat{\mathsf{ArgVal}}\mathsf{ue}}(\hat{H},\hat{C},\text{“}0\text{”})\\
& \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge \hat{o} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}_1.2} \hat{H}(\hat{l}) \wedge n = 0\\
& \wedge \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
& \wedge \hat{o}_1 = \underline{\widehat{\mathsf{NewArray}}}(0)\left[\ \forall s \in dom(o) : n^{++} \mapsto \langle s, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle\ \right] \textit{// ignore @default, unsound??}\\
& \wedge \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \hat{o}_1]\\
& \wedge (\hat{H}_3,\hat{C}_3) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\{\hat{l}_R\})),\hat{C}_1)\\
& \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
\text{where }\ & \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
& \wedge \hat{v} = \underline{\mathsf{get}\widehat{\mathsf{ArgVal}}\mathsf{ue}}(\hat{H},\hat{C},\text{“}0\text{”}) \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge \hat{o}_1 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \underline{\mathsf{New}\widehat{\mathsf{ArrayO}}\mathsf{bject}}(\mathsf{UInt})[\forall \hat{s} \in dom(\hat{H}_1(\hat{l})) : \mathsf{NumStr} \mapsto \langle \hat{s}, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle]\\
& \wedge \hat{o}_2 = \begin{cases} \underline{\mathsf{New}\widehat{\mathsf{ArrayO}}\mathsf{bject}}(\mathsf{UInt})[\mathsf{NumStr} \mapsto \langle \mathsf{NumStr}, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle] & \text{if } \ \hat{H}_1(\hat{l})(\text{“}\textit{@default\_number}\text{”}) \not\sqsubseteq \bot_{PropValue}\\ \bot_{Obj} & \text{otherwise}\end{cases}\\
& \wedge \hat{o}_3 = \begin{cases} \underline{\mathsf{New}\widehat{\mathsf{ArrayO}}\mathsf{bject}}(\mathsf{UInt})[\mathsf{NumStr} \mapsto \langle \mathsf{OtherStr}, \hat{\mathsf{true}}, \hat{\mathsf{true}}, \hat{\mathsf{true}}\rangle] & \text{if } \ \hat{H}_1(\hat{l})(\text{“}\textit{@default\_other}\text{”}) \not\sqsubseteq \bot_{PropValue}\\ \bot_{Obj} & \text{otherwise}\end{cases}\\
& \wedge \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \hat{o}_1 \sqcup \hat{o}_2 \sqcup \hat{o}_3] \wedge\ (\hat{H}_3,\hat{C}_3) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)),\hat{C}) & \text{if } \ \hat{o} \not\sqsubseteq \bot_{Obj}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
& \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}_1) \\
\quad \text{where } \;\hat{l}_R = (\hat{a}_{new}, \mathit{Recent}) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction} \\
\qquad\qquad \wedge\, \hat{v}_1 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H}_1,\hat{C}_1, \text{“0”}) \wedge \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(\mathit{getArgValue}(\hat{H}_1,\hat{C}_1, \text{“}\mathit{len\hat{g}th}\text{”})) \\
\qquad\qquad \wedge\, \hat{v}_2 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H}_1,\hat{C}_1, \text{“1”}) \wedge \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \wedge \hat{n}_{arglen} = \hat{2} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \underline{\widehat{\mathsf{NewObject}}}(\hat{v}_1.2)] \\
\qquad\qquad \wedge\, \hat{H}_3 = \begin{cases} \bigsqcup_{\hat{l}_2\in\hat{v}_2.2} \bigsqcup_{s\in\underline{\widehat{\mathsf{GetProps}}}(\hat{H}_2,\hat{l}_2)} \underline{\widehat{\mathsf{DefineProperty}}}(\hat{H}_2,\hat{l}_R,\hat{s},\hat{l}_2) & \text{if } \;\hat{n}_{arglen} = \hat{2} \\ \hat{H}_2 & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\{\hat{l}_R\}),\hat{C}_1) \\
\qquad\qquad \wedge\, \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_1 \sqcup \hat{es}_2)
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.defineProperty}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}_1) \\
\quad \text{where } \;\hat{v}_1 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C}, \text{“0”}) \wedge \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{s}_{name} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C}, \text{“1”}))) \\
\qquad\qquad \wedge\, \hat{v}_2 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C}, \text{“2”}) \wedge \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{H}_1 = \bigsqcup_{\hat{l}_1\in\hat{v}_1.2} \bigsqcup_{\hat{l}_2\in\hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperty}}}(\hat{H},\hat{l}_1,\hat{s}_{name},\hat{l}_2) \\
\qquad\qquad \wedge\, (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\hat{v}_1.2)),\hat{C}) & \text{if } \;Value(\hat{v}_1.2) \not\sqsubseteq \bot_{value} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_1 \sqcup \hat{es}_2)
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.defineProperties}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}_1) \\
\quad \text{where } \;\hat{v}_1 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C}, \text{“0”}) \wedge \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{v}_2 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C}, \text{“1”}) \wedge \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \;\hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{H}_1 = \bigsqcup_{\hat{l}_1\in\hat{v}_1.2} \bigsqcup_{\hat{l}_2\in\hat{v}_2.2} \bigsqcup_{s\in\underline{\widehat{\mathsf{GetProps}}}(\hat{H},\hat{l}_2)} \underline{\widehat{\mathsf{DefineProperty}}}(\hat{H},\hat{l}_1,\hat{s},\hat{l}_2) \\
\qquad\qquad \wedge\, (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\hat{v}_1.2)),\hat{C}) & \text{if } \;Value(\hat{v}_1.2) \not\sqsubseteq \bot_{value} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_1 \sqcup \hat{es}_2)
\end{array}
$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.seal}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}_1)$$

$$\begin{aligned}
\text{where}\ \ \hat{v} &= \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\wedge \hat{H}_1 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \bigsqcup_{s\in\widehat{\mathsf{GetProps}}(\hat{H},\hat{l})} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[\begin{array}{l} s \mapsto \left\langle \begin{array}{l} \hat{H}(\hat{l})(x).1.1.1, \hat{H}(\hat{l})(x).1.1.2, \\ \hat{H}(\hat{l})(x).1.1.3, \widehat{\mathsf{false}} \end{array} \right\rangle \\ @\widehat{extensible} \mapsto \widehat{\mathsf{false}} \end{array}\right]\right] \\
&\wedge (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\hat{v}.2)),\hat{C}) & \text{if }\ Value(\hat{v}.2) \not\sqsubseteq \bot_{value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
&\wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.freeze}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}_1)$$

$$\begin{aligned}
\text{where}\ \ \hat{v} &= \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\wedge \hat{H}_1 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \bigsqcup_{s\in\hat{P}_{data}} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[\begin{array}{l} s \mapsto \left\langle \begin{array}{l} \hat{H}(\hat{l})(x).1.1.1, \widehat{\mathsf{false}}, \\ \hat{H}(\hat{l})(x).1.1.3, \widehat{\mathsf{false}} \end{array} \right\rangle \\ @\widehat{extensible} \mapsto \widehat{\mathsf{false}} \end{array}\right]\right] \\
&\wedge \hat{H}_2 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \bigsqcup_{s\in\hat{P}_{access}} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[\begin{array}{l} s \mapsto \left\langle \begin{array}{l} \hat{H}(\hat{l})(x).1.1.1, \widehat{\mathsf{false}}, \\ \hat{H}(\hat{l})(x).1.1.3, \widehat{\mathsf{false}} \end{array} \right\rangle \\ @\widehat{extensible} \mapsto \widehat{\mathsf{false}} \end{array}\right]\right] \\
&\wedge \hat{P}_{data} = \{x \mid x \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}(\hat{l})) \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsDataDescriptor}}(x)}\} \\
&\wedge \hat{P}_{access} = \{x \mid x \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}(\hat{l})) \wedge \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsDataDescriptor}}(x)}\} \\
&\wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \\
&\wedge (\hat{H}_4,\hat{C}_4) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_3, Value(\hat{v}.2)),\hat{C}) & \text{if }\ Value(\hat{v}.2) \not\sqsubseteq \bot_{value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
&\wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.preventExtensions}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}_1)$$

$$\begin{aligned}
\text{where}\ \ \hat{v} &= \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\wedge \hat{H}_1 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \hat{H}\left[\hat{l} \mapsto \hat{H}(\hat{l})\left[@\widehat{extensible} \mapsto \widehat{\mathsf{false}}\right]\right] \\
&\wedge (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\hat{v}.2)),\hat{C}) & \text{if }\ Value(\hat{v}.2) \not\sqsubseteq \bot_{value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases} \\
&\wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.isSealed}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad\text{where }\ \hat{v} = \underline{\widehat{\textsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\textsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b}_{f_{\hat{l}}} = \begin{cases} \widehat{\textsf{false}} & \text{if }\ \exists x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) : \widehat{\textsf{true}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.3\\ \bot_{bool} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b}_{t_{\hat{l}}} = \begin{cases} \top_{bool} & \text{if }\ \forall x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.3 \wedge \top_{bool} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extensible})\\ \widehat{\textsf{true}} & \text{if }\ \forall x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.3 \wedge \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extensible})\\ \widehat{\textsf{false}} & \text{if }\ \forall x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.3 \wedge \widehat{\textsf{true}} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extensible})\\ \bot_{bool} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2}(\hat{b}_{f_{\hat{l}}} \sqcup \hat{b}_{t_{\hat{l}}})\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{b})),\hat{C}) & \text{if }\ \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.isFrozen}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad\text{where }\ \hat{v} = \underline{\widehat{\textsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\textsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b}_{f_{\hat{l}}} = \begin{cases} \widehat{\textsf{false}} & \text{if }\ \exists x \in \hat{P}_{data} : \widehat{\textsf{true}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.2\\ \widehat{\textsf{false}} & \text{if }\ \exists x \in \hat{P}_{access} : \widehat{\textsf{true}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.4\\ \bot_{bool} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b}_{t_{\hat{l}}} = \begin{cases} \top_{bool} & \text{if }\ \forall x \in \hat{P}_{data} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.2 \wedge \forall x \in \hat{P}_{access} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.4 \wedge \top_{bool} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extens}\\ \widehat{\textsf{true}} & \text{if }\ \forall x \in \hat{P}_{data} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.2 \wedge \forall x \in \hat{P}_{access} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.4 \wedge \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extens}\\ \widehat{\textsf{false}} & \text{if }\ \forall x \in \hat{P}_{data} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.2 \wedge \forall x \in \hat{P}_{access} : \widehat{\textsf{false}} \sqsubseteq \hat{H}(\hat{l})(x).1.1.4 \wedge \widehat{\textsf{true}} \sqsubseteq \hat{H}(\hat{l})(@\widehat{extensi}\\ \bot_{bool} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{P}_{data} = \{x \mid x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) \wedge \widehat{\textsf{true}} \sqsubseteq \underline{\widehat{\textsf{IsDataDescriptor}}}(x)\}\\
&\qquad\quad \wedge \hat{P}_{access} = \{x \mid x \in \underline{\widehat{\textsf{GetProps}}}(\hat{H}(\hat{l})) \wedge \widehat{\textsf{false}} \sqsubseteq \underline{\widehat{\textsf{IsDataDescriptor}}}(x)\}\\
&\qquad\quad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2}(\hat{b}_{f_{\hat{l}}} \sqcup \hat{b}_{t_{\hat{l}}})\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{b})),\hat{C}) & \text{if }\ \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.isExtensible}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad\text{where }\ \hat{v} = \underline{\widehat{\textsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\textsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \hat{H}(\hat{l})(@\widehat{extensible}).1.2\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{b})),\hat{C}) & \text{if }\ \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Object.keys}\text{”}, args)_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad\text{where }\ \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\textsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad\quad \wedge \hat{v} = \underline{\widehat{\textsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}) \wedge \hat{es} = \begin{cases} \{\textsf{TypeError}\} & \text{if }\ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{o}_1 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \underline{\widehat{\textsf{NewArrayObject}}}(\textsf{UInt})[\forall \hat{s} \in \hat{P}_{enum} : \textsf{NumStr} \mapsto \langle \hat{s}, \widehat{\textsf{true}}, \widehat{\textsf{true}}, \widehat{\textsf{true}}\rangle]\\
&\qquad\quad \wedge \hat{P}_{enum} = \{s \mid s \in dom(\hat{H}_1(\hat{l})) \wedge \widehat{\textsf{true}} \sqsubseteq \hat{H}_1(\hat{l})(s).1.1.3\}\\
&\qquad\quad \wedge \hat{o}_2 = \begin{cases} \underline{\widehat{\textsf{NewArrayObject}}}(\textsf{UInt})[\textsf{NumStr} \mapsto \langle \textsf{NumStr}, \widehat{\textsf{true}}, \widehat{\textsf{true}}, \widehat{\textsf{true}}\rangle] & \text{if }\ \hat{H}_1(\hat{l})(\text{“}\textit{@default\_number}\text{”}) \not\sqsubseteq \bot_{PropValue}\\ \bot_{Obj} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{o}_3 = \begin{cases} \underline{\widehat{\textsf{NewArrayObject}}}(\textsf{UInt})[\textsf{NumStr} \mapsto \langle \textsf{OtherStr}, \widehat{\textsf{true}}, \widehat{\textsf{true}}, \widehat{\textsf{true}}\rangle] & \text{if }\ \hat{H}_1(\hat{l})(\text{“}\textit{@default\_other}\text{”}) \not\sqsubseteq \bot_{PropValue}\\ \bot_{Obj} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \hat{o}_1 \sqcup \hat{o}_2 \sqcup \hat{o}_3] \wedge\ (\hat{H}_3,\hat{C}_3) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)),\hat{C}) & \text{if }\ \hat{o} \not\sqsubseteq \bot_{Obj}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})
\end{aligned}$$

11.2.4 Object.prototype

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.toString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad\text{where } \hat{s} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \alpha(\text{“}[object\text{”} + s + \text{“}]\text{”}) & \text{if } \hat{H}(\hat{l})(@\widehat{class}) = \mathsf{NumStrSingle}(s)\\ \alpha(\text{“}[object\text{”} + s + \text{“}]\text{”}) & \text{if } \hat{H}(\hat{l})(@\widehat{class}) = \mathsf{OtherStrSingle}(s)\\ \bot_{string} & \text{if } \hat{H}(\hat{l})(@\widehat{class}) = \bot_{string}\\ \mathsf{OtherStr} & \text{otherwise} \end{cases}\\
&\qquad\wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s})),\hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.toLocaleString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S})\\
&= \hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.toString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.ValueOf}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad\text{where } (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{C}.2)),\hat{C}) & \text{if } Value(\hat{C}.2) \not\sqsubseteq \bot_{Value}\\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.hasOwnProperty}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad\text{where } \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}0\text{”})))\\
&\qquad\wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{HasOwnProperty}}}(\hat{H},\hat{l},\hat{s})\\
&\qquad\wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{b})),\hat{C}) & \text{if } \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.isPrototypeOf}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad\text{where } \hat{v} = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}0\text{”})\\
&\qquad\wedge \hat{b}_1 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \hat{v}.1 \sqsubseteq \bot_{PValue}\\ \bot_{bool} & \text{otherwise} \end{cases} \quad \wedge \hat{b}_2 = \textstyle\bigsqcup_{\hat{l}\in\hat{v}.2} \hat{b}_{3_{\hat{l}}} \sqcup \hat{b}_{4_{\hat{l}}}\\
&\qquad\wedge \hat{v}_{proto_{\hat{l}}} = \hat{H}(\hat{l})(@\widehat{proto}).1.2\\
&\qquad\wedge \hat{b}_{3_{\hat{l}}} = \begin{cases} \widehat{\mathsf{false}} & \text{if } \top_{null} \sqsubseteq \hat{v}_{proto_{\hat{l}}}.1.2\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\wedge \hat{b}_{4_{\hat{l}}} = (Value(\hat{v}.2)\mathrel{\widehat{==}}Value(\hat{v}_{proto_{\hat{l}}}.2)).1.3\\
&\qquad\wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{b}_1 \sqcup \hat{b}_2)),\hat{C}) & \text{if } \hat{b}_1 \sqcup \hat{b}_2 \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.prototype.propertyIsEnumerable}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad\text{where } \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}0\text{”})))\\
&\qquad\wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{b}_{1_{\hat{l}}} \sqcup \hat{b}_{2_{\hat{l}}}\\
&\qquad\wedge \hat{b}_{1_{\hat{l}}} = \begin{cases} \widehat{\mathsf{false}} & \text{if } \top_{undef} \sqsubseteq \hat{H}(\hat{l})(\hat{s}).1.1.1.1.1\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\wedge \hat{b}_{2_{\hat{l}}} = \begin{cases} \hat{H}(\hat{l}).1.1.3 & \text{if } \hat{H}(\hat{l})(\hat{s}).1.1.1.1.1 \sqsubseteq \bot_{undef}\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{b})),\hat{C}) & \text{if } \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

11.2.5 Function.prototype

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype.toString}\text{”}, args))_{\hat{a}_{new}}\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}) \\
&\quad \text{where } \ \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{C}.2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad\qquad \wedge\ \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \exists \hat{l} \in \hat{C}.2.2 : \hat{H}(\hat{l})(@\hat{class}) \neq \text{“}\widehat{Function}\text{”} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad\qquad \wedge\ \hat{L}_{fun} = \{\hat{l} \mid \hat{l} \in \hat{C}.2.2 \wedge \hat{H}(\hat{l})(@\hat{class}) = \text{“}\widehat{Function}\text{”}\} \\
&\qquad\qquad \wedge\ \hat{s} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{fun}} \underline{\widehat{\mathsf{fid2String}}}(\hat{H}(\hat{l})(@\hat{\hat{function}})) \\
&\qquad\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{s}]], \hat{C}) \\
&\qquad\qquad \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_1 \sqcup \hat{es}_2)
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype.apply}\text{”}, args))_{\hat{a}_{new}}\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S}) \\
&\quad \text{where } \ \hat{v}_1 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}0\text{”}) \\
&\qquad\qquad \wedge\ \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad\qquad \wedge\ \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq \bigsqcup_{\hat{l}\in\hat{v}_1.2} : \underline{\widehat{\mathsf{IsCallable}}}(\hat{H},\hat{l}) \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad\qquad \wedge\ \hat{L}_f = \{\hat{l} \mid \hat{l} \in \hat{v}_1.2 \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H},\hat{l})\} \\
&\qquad\qquad \wedge\ \hat{v}_2 = \underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“}1\text{”}) \\
&\qquad\qquad \wedge\ \hat{es}_3 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \langle \bot_{undef}, \bot_{null}, \hat{v}_2.1.3, \hat{v}_2.1.4, \hat{v}_2.1.5 \rangle \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad\qquad \wedge
\end{aligned}
$$

11.2.6 Array

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H}, \hat{C}), \hat{S})\\
&= \hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.constructor}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H}, \hat{C}), \hat{S})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.constructor}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_3, \hat{C}_3), \hat{S}_1)\\
&\quad \text{where } \ (\hat{H}_1, \hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H}, \hat{C}, \hat{a}_{new}) \wedge \ \hat{l}_R = (\hat{a}_{new}, Recent) \quad \textit{// Recency Abstraction}\\
&\qquad\qquad \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(\textit{getArgValue}(\hat{H}_1, \hat{C}_1, \text{“}\textit{length}\text{”}))\\
&\qquad\qquad \wedge\ \hat{o}_1 = \begin{cases} \hat{o}_{arg1_1} \sqcup \hat{o}_{arg1_2} & \text{if } \ \mathsf{UIntSingle}(1) = \hat{n}_{arglen}\\ \hat{o}_{argn} & \text{if } \ \mathsf{UIntSingle}(n_{arglen}) = \hat{n}_{arglen} \wedge n > 1\\ \hat{o}_{arg1_1} \sqcup \hat{o}_{arg1_2} \sqcup \hat{o}_{uint} & \text{if } \ \mathsf{UInt} \sqsubseteq \hat{n}_{arglen}\\ \bot_{Obj} & \text{if } \ \hat{n}_{arglen} \sqsubseteq \bot_{number} \end{cases}\\
&\qquad\qquad \wedge\ \hat{v}_i = \textit{getArgValue}(\hat{H}_1, \hat{C}_1, \text{“}i\text{”})\\
&\qquad\qquad \wedge\ \hat{v}0_{notNum} = Value(\langle \hat{v}_0.1.1, \hat{v}_0.1.2, \hat{v}_0.1.3, \bot_{Number}, \hat{v}_0.1.5\rangle, \hat{v}_0.2)\\
&\qquad\qquad \wedge\ \hat{o}_{arg1_1} = \begin{cases} \underline{\widehat{\mathsf{NewArrayObject}}}(1)[0 \mapsto \hat{v}0_{notNum}] & \text{if } \ \hat{v}0_{notNum} \not\sqsubseteq \bot_{Value}\\ \bot_{Obj} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{o}_{arg1_2} = \begin{cases} \underline{\widehat{\mathsf{NewArrayObject}}}(\hat{v}_0.1.4) & \text{if } \ \hat{v}_0.1.4 \not\sqsubseteq \bot_{Number}\\ \bot_{Obj} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{es} = \begin{cases} \{\mathsf{RangeError}\} & \text{if } \ \mathsf{UIntSingle}(1) = \hat{n}_{arglen} \wedge \hat{v}_0.1.4 \not\sqsubseteq \mathsf{UInt}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{o}_{argn} = \underline{\widehat{\mathsf{NewArrayObject}}}(n_{arglen})[\forall i \in \{0, ..., n_{arglen} - 1\} : i \mapsto v_i]\\
&\qquad\qquad \wedge\ \hat{o}_{uint} = \underline{\widehat{\mathsf{NewArrayObject}}}(n_{arglen})[@defaul\hat{t}\_number \mapsto \hat{v}_{allarg}]\\
&\qquad\qquad \wedge\ \hat{v}_{allarg} = \textit{getArgValue}(\hat{H}_1, \hat{C}_1, \mathsf{UInt})\\
&\qquad\qquad \wedge\ \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \hat{o}_1]\\
&\qquad\qquad \wedge\ (\hat{H}_3, \hat{C}_3) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)), \hat{C}) & \text{if } \ \hat{o}_1 \not\sqsubseteq \bot_{Obj}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H}, \hat{C}, \hat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.isArray}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})\\
&\quad \text{where } \ \hat{v} = \textit{getArgValue}(\hat{H}, \hat{C}, \text{“}0\text{”})\\
&\qquad\qquad \wedge\ \hat{b}_1 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \ \hat{v}.1 \not\sqsubseteq \bot_{PValue}\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{b}_2 = \begin{cases} \widehat{\mathsf{true}} & \text{if } \ \exists \hat{l} \in \hat{v}.2 : \text{“}Ar\hat{r}ay\text{”} \sqsubseteq \hat{H}(\hat{l})(@c\hat{l}ass).1.2.1.5\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{b}_3 = \begin{cases} \widehat{\mathsf{false}} & \text{if } \ \exists \hat{l} \in \hat{v}.2 : \hat{H}(\hat{l})(@c\hat{l}ass).1.2.1.5 \not\sqsubseteq \text{“}Ar\hat{r}ay\text{”}\\ \bot_{bool} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge\ \hat{b} = \hat{b}_1 \sqcup \hat{b}_2 \sqcup \hat{b}_3\\
&\qquad\qquad \wedge\ (\hat{H}_1, \hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{b})), \hat{C}) & \text{if } \ \hat{b} \not\sqsubseteq \bot_{Bool}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}
$$

11.2.7 Array.prototype

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toString}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where} \quad \color{red}{\textit{// [[get]] join, [[call]]}}\\
&\qquad \hat{v}_{length} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}_1.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”})\\
&\qquad \wedge\ \hat{s} = \begin{cases} \hat{s}_0 \hat{+} \hat{s}_{sep} \hat{+} ... \hat{+} \hat{s}_{sep} \hat{+} \hat{s}_{n_{len}-1} & \text{if } \ \hat{v}_{length}.1.4 = \mathsf{UIntSingle}(n_{len})\\ \bot_{string} & \text{if } \ \hat{v}_{length}.1.4 = \bot_{number}\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{s}_i = \begin{cases} \text{“}\hat{}\text{”} & \text{if } \ \hat{v}_i \not\sqsubseteq \bot_{Undef} \vee \hat{v}_i \not\sqsubseteq \bot_{N}\\ \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(Value(\langle\bot_{Undef},\bot_{Null},\hat{v}_{arg}.1.3,\hat{v}_{arg}.1.4,\hat{v}_{arg}.1.5\rangle, v_{arg}.2))) & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{v}_i = \textstyle\bigsqcup_{\hat{l}\in\hat{C}_1.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}i\text{”}) \wedge\ \hat{v}_{arg} = \textit{getArgValue}(args, \text{“}0\text{”})\\
&\qquad \wedge\ \hat{s}_{sep} = \text{“}\hat{,}\text{”}\\
&\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#Pur\hat{e}Local_R \mapsto \hat{H}(\#Pur\hat{e}Local_R)[@return \mapsto \hat{s}]],\hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toLocaleString}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S})\\
&= \hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toString}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.concat}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where} \ \ \color{blue}{(\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \wedge\ \hat{l}_R = (\hat{a}_{new}, \textit{Recent}) \quad \textit{// Recency Abstraction}}\\
&\qquad \wedge\ \hat{n}_{len} = \textit{getArgValue}(\hat{H}_1,\hat{C}_1,\text{“}lenght\text{”}).1.4\\
&\qquad \wedge\ \hat{H}_2 = \begin{cases} \bot_{Heap} & \text{if } \ \hat{n}_{len} = \bot_{Number}\\ \hat{H}_1[\hat{l}_R \mapsto \hat{o}_1] & \text{if } \ \hat{n}_{len} = \mathsf{UIntSingle}(n_{arglen})\\ \hat{H}_1[\hat{l}_R \mapsto \hat{o}_2] & \text{if } \ \hat{n}_{len} \neq \mathsf{UIntSingle} \end{cases}\\
&\qquad \wedge\ \hat{o}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}_1.2} \hat{H}_1(\hat{l})\\
&\qquad \wedge\ \hat{o}_1 = \begin{cases} \hat{o}_3 & \text{if } \ \hat{o}_{this}(\text{“}len\hat{g}th\text{”}) = \mathsf{UIntSingle}(n_{len})\\ \hat{o}_4 & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{o}_2 = \hat{o}_{this}[@defaul\hat{t}\_number \mapsto \hat{o}_{this}(@defaul\hat{t}\_number) \sqcup \textit{Value}(\top_{PValue},\{\})]\\
&\qquad \wedge\ \hat{o}_3 = \hat{o}_{this}[\forall i \in \{0,...,n_{arglen}-1\} : i + n_{len} \mapsto \hat{v}_{arg_i}, length \mapsto \alpha(n_{len}+n_{arglen})]\\
&\qquad \wedge\ \hat{v}_{arg_i} = \textit{getArgValue}(\hat{H}_1,\hat{C}_1,\text{“}i\text{”})\\
&\qquad \wedge\ \hat{o}_4 = \hat{o}_{this}[@defaul\hat{t}\_number \mapsto \hat{o}_{this}(@defaul\hat{t}\_number) \sqcup \textstyle\bigsqcup_{\hat{i}\in\{0,...,n_{len}\}} \hat{v}_{arg_i}]\\
&\qquad \wedge\ (\hat{H}_3,\hat{C}_3) = (\hat{H}_2[\#Pur\hat{e}Local_R \mapsto \hat{H}_2(\#Pur\hat{e}Local_R)[@return \mapsto Value(\hat{l}_R)]],\hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.join}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where} \ \ \hat{v}_{length} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}_1.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”})\\
&\qquad \wedge\ \hat{s} = \begin{cases} \hat{s}_0 \hat{+} \hat{s}_{sep} \hat{+} ... \hat{+} \hat{s}_{sep} \hat{+} \hat{s}_{n_{len}-1} & \text{if } \ \hat{v}_{length}.1.4 = \mathsf{UIntSingle}(n_{len})\\ \bot_{string} & \text{if } \ \hat{v}_{length}.1.4 = \bot_{number}\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{s}_i = \begin{cases} \text{“}\hat{}\text{”} & \text{if } \ \hat{v}_i \not\sqsubseteq \bot_{Undef} \vee \hat{v}_i \not\sqsubseteq \bot_{N}\\ \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(Value(\langle\bot_{Undef},\bot_{Null},\hat{v}_{arg}.1.3,\hat{v}_{arg}.1.4,\hat{v}_{arg}.1.5\rangle, v_{arg}.2))) & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{v}_i = \textstyle\bigsqcup_{\hat{l}\in\hat{C}_1.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}i\text{”}) \wedge\ \hat{v}_{arg} = \textit{getArgValue}(args, \text{“}0\text{”})\\
&\qquad \wedge\ \hat{s}_{sep} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(Value(\langle\bot_{undef},\hat{v}_{arg}.1.2,\hat{v}_{arg}.1.3,\hat{v}_{arg}.1.4,\hat{v}_{arg}.1.5\rangle, v_{arg}.2))) \sqcup \hat{s}_{udf}\\
&\qquad \wedge\ \hat{s}_{udf} = \begin{cases} \text{“}\hat{,}\text{”} & \text{if } \ v_1.1.1 \sqsubseteq \bot_{Undef}\\ \bot_{string} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#Pur\hat{e}Local_R \mapsto \hat{H}(\#Pur\hat{e}Local_R)[@return \mapsto \hat{s}]],\hat{C})
\end{aligned}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.pop}\text{”}, args))]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_5, \hat{C}_5), \hat{S}) \\
\quad \text{where } \ (\hat{v}_{\hat{l}}, \hat{H}_{1_{\hat{l}}}) = \begin{cases} (Value(\top_{undef}), \hat{H}_{2_{\hat{l}}}) & \text{if } \ \hat{n}_{len_{\hat{l}}} = \mathsf{UIntSingle}(0) \\ (\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \widehat{n_{len} - 1}), \hat{H}_{3_{\hat{l}}}) & \text{if } \ \hat{n}_{len_{\hat{l}}} = \mathsf{UIntSingle}(n_{len}) \\ (\bot_{Value}, \bot_{Heap}) & \text{if } \ \hat{n}_{len_{\hat{l}}} = \bot_{number} \\ (\hat{H}(\hat{l})(@\widehat{default\_number}), \hat{H}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{n}_{len_{\hat{l}}} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}length\text{”})) \\
\qquad\qquad \wedge\, \hat{H}_{2_{\hat{l}}} = \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\text{“}\widehat{length}\text{”} \mapsto \hat{0}]] \\
\qquad\qquad \wedge\, \hat{H}_{3_{\hat{l}}} = (\underline{\widehat{\mathsf{Delete}}}(\hat{H}, \hat{l}, \widehat{n_{len} - 1}).1)[\hat{l} \mapsto \hat{H}(\hat{l})[\text{“}\widehat{length}\text{”} \mapsto \widehat{n_{len} - 1}]] \\
\qquad\qquad \wedge\, (\hat{v}, \hat{H}_4) = \bigsqcup_{\hat{l} \in \hat{C}.2} (\hat{v}_{\hat{l}}, \hat{H}_{1_{\hat{l}}}) \\
\qquad\qquad \wedge\, (\hat{H}_5, \hat{C}_5) = (\hat{H}_4[\#\widehat{PureLocal}_R \mapsto \hat{H}_4(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]], \hat{C})
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.push}\text{”}, args))]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_3, \hat{C}_3), \hat{S}) \\
\quad \text{where } \ \hat{n}_{len_{\hat{l}}} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}length\text{”})) \\
\qquad\qquad \wedge\, \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(\mathit{getArgValue}(args, \text{“}length\text{”})) \\
\qquad\qquad \wedge\, (\hat{H}_1, \hat{v}_1) = \bigsqcup_{\hat{l} \in \hat{C}.2} \begin{cases} \bot_{Heap}, \bot_{value} & \text{if } \ \hat{n}_{arglen} = \bot_{Number} \\ \hat{H}_1[\hat{l} \mapsto \hat{o}_1], \hat{n}_{len_{\hat{l}}} + \hat{n}_{arglen} & \text{if } \ \hat{n}_{len_{\hat{l}}} = \mathsf{UIntSingle}(n_{arglen}) \\ \hat{H}_1[\hat{l} \mapsto \hat{o}_2], \hat{n}_{len_{\hat{l}}} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{o}_1 = \hat{H}(\hat{l})[\forall i \in \{0, ..., n_{arglen} - 1\} : i + \hat{n}_{len} \mapsto \mathit{getArgValue}(args, i), \text{“}length\text{”} \mapsto \hat{n}_{len} + \hat{n}_{arglen}] \\
\qquad\qquad \wedge\, \hat{o}_2 = \hat{H}(\hat{l})[\hat{n}_{arglen} \mapsto \mathit{getArgValue}(args, \hat{n}_{arglen})] \\
\qquad\qquad \wedge\, (\hat{H}_2, \hat{C}_2) = (\hat{H}_1[\#\widehat{PureLocal}_R \mapsto \hat{H}_1(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}_1]], \hat{C})
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.reverse}\text{”}, args))]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S}) \\
\quad \text{where } \ \hat{H}_1 = \bigsqcup_{\hat{l} \in \hat{C}.2} \begin{cases} \hat{H}[\hat{l} \mapsto \hat{o}_1] & \text{if } \ \hat{n}_{len_{\hat{l}}} = \mathsf{UIntSingle}(n_{len_{\hat{l}}}) \\ \hat{H}[\hat{l} \mapsto \hat{o}_2] & \text{if } \ \hat{n}_{len_{\hat{l}}} = \mathsf{UInt} \end{cases} \\
\qquad\qquad \wedge\, \hat{n}_{len_{\hat{l}}} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}length\text{”})) \\
\qquad\qquad \wedge\, \hat{o}_1 = \forall i \in \{0, ..., n_{mid_{\hat{l}}} - 1\} : \hat{o}_{low1} \sqcup \hat{o}_{low2} \sqcup \hat{o}_{up1} \sqcup \hat{o}_{up2} \\
\qquad\qquad \wedge\, n_{up} = n_{len_{\hat{l}}} - i - 1 \\
\qquad\qquad \wedge\, \hat{o}_{low1} = \begin{cases} \hat{H}(\hat{l})[n_{up} \mapsto \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, i)] & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}, i) \\ \hat{H}(\hat{l}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{o}_{low2} = \begin{cases} \hat{H}(\hat{l}) - n_{up} & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}, i) \\ \hat{H}(\hat{l}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{o}_{up1} = \begin{cases} \hat{H}(\hat{l})[i \mapsto \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, n_{up})] & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}, n_{up}) \\ \hat{H}(\hat{l}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{o}_{up2} = \begin{cases} \hat{H}(\hat{l}) - i & \text{if } \ \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{l}, n_{up}) \\ \hat{H}(\hat{l}) & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\, \hat{o}_2 = \forall s \in \underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H}, \hat{l}) : \hat{o}_3 - s \\
\qquad\qquad \wedge\, \hat{o}_3 = \hat{H}(\hat{l})[@\widehat{default\_number} \mapsto \hat{H}(\hat{l})(@\widehat{default\_number}) \sqcup \bigsqcup_{s \in \underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H}, \hat{l})} \hat{H}(\hat{l})(s)] \\
\qquad\qquad \wedge\, (\hat{H}_2, \hat{C}_2) = (\hat{H}_1[\#\widehat{PureLocal}_R \mapsto \hat{H}_1(\#\widehat{PureLocal}_R)[@return \mapsto \hat{C}.2]], \hat{C})
\end{array}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.shift}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})\\
&\quad\text{where}\ \ (\hat{H}_1,\hat{v}_1) = \bigsqcup_{\hat{l}\in\hat{C}.2}\begin{cases}(\hat{H}[\hat{l}\mapsto\hat{o}_1],\hat{v}_{head}) & \text{if}\ \ \hat{n}_{len_{\hat{l}}} = \mathsf{UIntSingle}(n_{len_{\hat{l}}})\\ (\hat{H}[\hat{l}\mapsto\hat{o}_2],\hat{v}_{uint}) & \text{if}\ \ \hat{n}_{len_{\hat{l}}} = \mathsf{UInt}\end{cases}\\
&\qquad\qquad\wedge\ \hat{n}_{len_{\hat{l}}} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”}))\\
&\qquad\qquad\wedge\ \hat{o}_2 = (\forall i\in\{1,...,n_{len_{\hat{l}}}-1\}:\hat{H}(\hat{l})[i-1\mapsto\hat{H}(\hat{l})(i)]) - n_{len_{\hat{l}}}\\
&\qquad\qquad\wedge\ \hat{v}_{head} = \begin{cases}\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}0\text{”}) & \text{if}\ \ n_{len_{\hat{l}}}\neq 0\\ Value(\top_{undef}) & \text{if}\ \ n_{len_{\hat{l}}} = 0\end{cases}\\
&\qquad\qquad\wedge\ \hat{o}_2 = \bigsqcup_{i\in\underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H},\hat{l})}(\hat{H}(\hat{l})[i-1\mapsto\hat{H}(\hat{l})(i)]-i)\\
&\qquad\qquad\wedge\ \hat{v}_{uint} = Value(\top_{undef})\sqcup\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}0\text{”})\\
&\qquad\qquad\wedge\ (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\#Pur\hat{e}Local_R\mapsto\hat{H}_1(\#Pur\hat{e}Local_R)[@return\mapsto\hat{v}_1]],\hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.slice}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})\\
&\quad\text{where}\ \ (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new})\wedge\ \hat{l}_R = (\hat{a}_{new}, Recent)\quad \textit{// Recency Abstraction}\\
&\qquad\qquad\wedge\ \hat{o}_{new} = \underline{\widehat{\mathsf{NewArrayObject}}}(0)\\
&\qquad\qquad\wedge\ \hat{v}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”}))\\
&\qquad\qquad\wedge\ \hat{v}_{start} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(args,\text{“}0\text{”}))\\
&\qquad\qquad\wedge\ \hat{n}_k = \begin{cases}max((\hat{v}_{start}+v_{len}),0)\sqcup min(\hat{v}_{start},v_{len}) & \text{if}\ \ \top_{bool} = \hat{v}_{start}<0\\ max((\hat{v}_{start}+v_{len}),0) & \text{if}\ \ \hat{\mathsf{true}} = \hat{v}_{start}<0\\ min(\hat{v}_{start},v_{len}) & \text{if}\ \ \hat{\mathsf{false}} = \hat{v}_{start}<0\end{cases}\\
&\qquad\qquad\wedge\ \hat{v}_{end} = \begin{cases}\underline{\widehat{\mathsf{toInteger}}}(getArgValue(args,\text{“}0\text{”}))\sqcup\hat{v}_{len} & \text{if}\ \ \top_{undef}\sqsubseteq getArgValue(args,\text{“}1\text{”}).1.1\\ \underline{\widehat{\mathsf{toInteger}}}(getArgValue(args,\text{“}0\text{”})) & \text{otherwise}\end{cases}\\
&\qquad\qquad\wedge\ \hat{n}_{final} = \begin{cases}max((\hat{v}_{end}+v_{len}),0)\sqcup min(\hat{v}_{end},v_{len}) & \text{if}\ \ \top_{bool} = \hat{v}_{end}<0\\ max((\hat{v}_{end}+v_{len}),0) & \text{if}\ \ \hat{\mathsf{true}} = \hat{v}_{end}<0\\ min(\hat{v}_{end},v_{len}) & \text{if}\ \ \hat{\mathsf{false}} = \hat{v}_{end}<0\end{cases}\\
&\qquad\qquad\wedge\ \hat{o} = \begin{cases}\hat{o}_{slice} & \text{if}\ \ \hat{n}_k = \mathsf{UIntSingle}(n_k)\wedge\hat{n}_{final} = \mathsf{UIntSingle}(n_{final})\\ \hat{o}_{uint} & \text{otherwise}\end{cases}\\
&\qquad\qquad\wedge\ \hat{o}_{slice} = \bigsqcup_{i\in\{n_k,...,n_{final}-1\}]}\hat{o}_{slice1}\sqcup\hat{o}_{slice2}\\
&\qquad\qquad\wedge\ \hat{o}_{slice1} = \begin{cases}\hat{o}_{new}[i-n_k\mapsto\hat{H}(\hat{l})(i)] & \text{if}\ \ \hat{\mathsf{true}}\sqsubseteq\underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},i)\\ \bot_{obj} & \text{otherwise}\end{cases}\\
&\qquad\qquad\wedge\ \hat{o}_{slice2} = \begin{cases}\hat{o}_{new} & \text{if}\ \ \hat{\mathsf{false}}\sqsubseteq\underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},i)\\ \bot_{obj} & \text{otherwise}\end{cases}\\
&\qquad\qquad\wedge\ \hat{o}_{uint} = \hat{o}_{new}[@defaul\hat{t}\_number\mapsto\hat{H}(\hat{l})(@defaul\hat{t}\_number)\sqcup Value(\top_{undef})\sqcup\bigsqcup_{i\in\underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H},\hat{l})}\hat{H}(\hat{l})(i)]\\
&\qquad\qquad\wedge\ \hat{H}_1 = \hat{H}[\hat{l}_R\mapsto\bigsqcup_{\hat{l}in\hat{C}.2}\hat{o}]\\
&\qquad\qquad\wedge\ (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\#Pur\hat{e}Local_R\mapsto\hat{H}_1(\#Pur\hat{e}Local_R)[@return\mapsto\hat{l}_R]],\hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.splice}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})\\
&\quad \text{where } (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \wedge \hat{l}_R = (\hat{a}_{new}, Recent) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge \hat{o}_{new} = \underline{\widehat{\mathsf{NewArrayObject}}}(0)\\
&\qquad \wedge \hat{v}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”}))\\
&\qquad \wedge \hat{n}_{argstat} = getArgValue(args,\text{“}0\text{”})),0)\\
&\qquad \wedge \hat{n}_{start} = \begin{cases} max((\hat{v}_{argstat}+v_{len}),0) \sqcup min(\hat{v}_{argstat},v_{len}) & \text{if } \top_{bool} = \hat{v}_{argstat} < 0\\ max((\hat{v}_{argstat}+v_{len}),0) & \text{if } \widehat{\mathsf{true}} = \hat{v}_{argstat} < 0\\ min(\hat{v}_{argstat},v_{len}) & \text{if } \widehat{\mathsf{false}} = \hat{v}_{argstat} < 0 \end{cases}\\
&\qquad \wedge \hat{v}_{count} = min(max(\underline{\widehat{\mathsf{toInteger}}}(getArgValue(args,\text{“}1\text{”})),0),\hat{v}_{len}-\hat{n}_{start})\\
&\qquad \wedge \hat{o} = \begin{cases} \hat{o}_{splice}[length \mapsto n_{final}-n_k] & \text{if } \hat{n}_{start} = \mathsf{UIntSingle}(n_{start}) \wedge \hat{n}_{count} = \mathsf{UIntSingle}(n_{count})\\ \hat{o}_{uint}[length \mapsto \mathsf{UInt}] & \text{otherwise} \end{cases}\\
&\qquad \wedge \hat{o}_{this_{\hat{l}}} = \begin{cases} \hat{o}_{this_{del}}[length \mapsto n_{final}-n_k] & \text{if } \hat{n}_{start} = \mathsf{UIntSingle}(n_{start}) \wedge \hat{n}_{count} = \mathsf{UIntSingle}(n_{count})\\ \hat{o}_{this_{uint}}[length \mapsto \mathsf{UInt}] & \text{otherwise} \end{cases}\\
&\qquad \wedge \hat{o}_{slice} = \textstyle\bigsqcup_{i\in\{n_k,\ldots,n_{final}-1\}]} \hat{o}_{slice1} \sqcup \hat{o}_{slice2}\\
&\qquad \wedge \hat{o}_{slice1} = \begin{cases} \hat{o}_{new}[i-n_k \mapsto \hat{H}(\hat{l})(i)] & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},i)\\ \bot_{obj} & \text{otherwise} \end{cases}\\
&\qquad \wedge \hat{o}_{slice2} = \begin{cases} \hat{o}_{new} & \text{if } \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},i)\\ \bot_{obj} & \text{otherwise} \end{cases}\\
&\qquad \wedge \hat{o}_{uint} = \hat{o}_{new}[@defaul\hat{t}\_number \mapsto \hat{H}(\hat{l})(@defaul\hat{t}\_number) \sqcup Value(\top_{undef}) \sqcup \textstyle\bigsqcup_{i\in\underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H},\hat{l})} \hat{H}(\hat{l})(i)]\\
&\qquad \wedge \hat{o}_{this_{uint}} = \hat{H}(\hat{l})[@defaul\hat{t}\_number \mapsto \hat{H}(\hat{l})(@defaul\hat{t}\_number) \sqcup Value(\top_{undef}) \sqcup \textstyle\bigsqcup_{i\in\underline{\widehat{\mathsf{GetUIntProps}}}(\hat{H},\hat{l})} \hat{H}(\hat{l})(i)]\\
&\qquad \wedge \hat{H}_1 = \hat{H}[\hat{l}_R \mapsto \hat{o}]\\
&\qquad \wedge (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\#Pur\hat{e}Local_R \mapsto \hat{H}_1(\#Pur\hat{e}Local_R)[@return \mapsto \hat{l}_R]],\hat{C})
\end{aligned}
$$

11.2.8 String

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{n}_{arglen} = \widehat{\underline{\mathsf{toUInt32}}}(\mathit{getArgValue}(\hat{H},\hat{C},\text{“}\mathit{length}\text{”}))\\
&\qquad \wedge\ \hat{s}_1 = \begin{cases} \text{“}\hat{}\text{”} & \text{if } \mathsf{UIntSingle}(0) \sqsubseteq \hat{n}_{arglen}\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ \hat{s}_2 = \begin{cases} \widehat{\underline{\mathsf{toString}}}(\widehat{\underline{\mathsf{toPrimitive}}}(\mathit{getArgValue}(\hat{H},\hat{C},\text{“}0\text{”}))) & \text{if } \mathsf{UIntSingle}(n) \sqsubseteq \hat{n}_{arglen} \wedge n > 0\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ \hat{s} = \hat{s}_1 \sqcup \hat{s}_2\\
&\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s})),\hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.constructor}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where } (\hat{H}_1,\hat{C}_1) = \widehat{\underline{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \wedge\ \hat{l}_R = (\hat{a}_{new}, \mathit{Recent}) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge\ \hat{n}_{arglen} = \widehat{\underline{\mathsf{toUInt32}}}(\mathit{getArgValue}(\hat{H}_1,\hat{C}_1,\text{“}\mathit{length}\text{”}))\\
&\qquad \wedge\ \hat{s}_1 = \begin{cases} \text{“}\hat{}\text{”} & \text{if } \mathsf{UIntSingle}(0) \sqsubseteq \hat{n}_{arglen}\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ \hat{s}_2 = \begin{cases} \widehat{\underline{\mathsf{toString}}}(\widehat{\underline{\mathsf{toPrimitive}}}(\mathit{getArgValue}(\hat{H}_1,\hat{C}_1,\text{“}0\text{”}))) & \text{if } \mathsf{UIntSingle}(n) \sqsubseteq \hat{n}_{arglen} \wedge n > 0\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ \hat{s} = \hat{s}_1 \sqcup \hat{s}_2\\
&\qquad \wedge\ \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \widehat{\underline{\mathsf{NewString}}}(\hat{s})]\\
&\qquad \wedge\ (\hat{H}_3,\hat{C}_3) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)),\hat{C}_1) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.fromCharCode}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{n}_{arglen} = \widehat{\underline{\mathsf{toUInt32}}}(\mathit{getArgValue}(\hat{H},\hat{C},\text{“}\mathit{length}\text{”}))\\
&\qquad \wedge\ \hat{s}_1 = \begin{cases} \text{“}\hat{}\text{”} & \text{if } \mathsf{UIntSingle}(0) \sqsubseteq \hat{n}_{arglen}\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ \hat{s}_2 = \begin{cases} \top_{String} & \text{if } \top_{UIntSingle}(n) \neq \hat{n}_{arglen} \wedge \hat{n}_{arglen} \not\sqsubseteq \bot_{Number}\\ \text{“}\hat{}\text{”}\,\hat{+}\,\hat{s}_0\,\hat{+}\ldots\hat{+}\,\hat{s}_{n-1} & \text{if } \mathsf{UIntSingle}(n) \sqsubseteq \hat{n}_{arglen} \wedge n > 0\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge\ s_i = \widehat{\underline{\mathsf{toChar}}}(\widehat{\underline{\mathsf{toUInt16}}}(\mathit{getArgValue}(\hat{H},\hat{C},\text{“}i\text{”}))) \wedge\ \hat{s} = \hat{s}_1 \sqcup \hat{s}_2\\
&\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s})),\hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

11.2.9 String.prototype

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toString”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad \text{where } \ \widehat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{string}\text{”} \\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{L}_{string} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{string}\text{”}\}\\
&\qquad\quad \wedge \hat{v} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{string}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\hat{v})),\hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\widehat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.valueOf”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad \text{where } \ \widehat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{string}\text{”} \\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{L}_{string} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{string}\text{”}\}\\
&\qquad\quad \wedge \hat{v} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{string}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\hat{v})),\hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\widehat{es})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charAt”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \ \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad\quad \wedge \hat{v}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad\quad \wedge \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad\quad \wedge \hat{n}_{size} = \mid \hat{s}_{this} \mid\\
&\qquad\quad \wedge \hat{n}_{pos} = \underline{\widehat{\mathsf{toInteger}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}))\\
&\qquad\quad \wedge \hat{v}_1 = \begin{cases} \widehat{\text{“”}} & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{pos} < \hat{0}) \\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{v}_2 = \begin{cases} \widehat{\text{“”}} & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{size} < \hat{n}_{pos}) \vee \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{size} = \hat{n}_{pos}) \\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{v}_3 = \wedge \hat{v} = \hat{v}_1 \sqcup \hat{v}_2 \sqcup \alpha(\mathit{native.charAt}(\gamma(\hat{s}_{this}),\gamma(n_{pos}))) \quad \textit{// java, scala}\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\hat{v})),\hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charCodeAt”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \ \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad\quad \wedge \hat{v}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad\quad \wedge \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad\quad \wedge \hat{n}_{size} = \mid \hat{s}_{this} \mid\\
&\qquad\quad \wedge \hat{n}_{pos} = \underline{\widehat{\mathsf{toInteger}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H},\hat{C},\text{“0”}))\\
&\qquad\quad \wedge \hat{v}_1 = \begin{cases} \widehat{\mathsf{NaN}} & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{pos} < \hat{0}) \\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{v}_2 = \begin{cases} \widehat{\mathsf{NaN}} & \text{if } \ \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{size} < \hat{n}_{pos}) \vee \widehat{\mathsf{true}} \sqsubseteq (\hat{n}_{size} = \hat{n}_{pos}) \\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad\quad \wedge \hat{v} = \hat{v}_1 \sqcup \hat{v}_2 \sqcup \alpha(\mathit{native.charAt}(\gamma(\hat{s}_{this}),\gamma(n_{pos})).toInt) \quad \textit{// java, scala}\\
&\qquad\quad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\hat{v})),\hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value} \\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.concat}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}) \\
\quad \text{where } \; \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@pri\hat{m}itive\text{”} \dot{\in} \hat{H}(\hat{l})\} \\
\qquad\qquad \wedge\ \hat{v}_{this} = \bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \quad \textit{// [[DefaultValue]]??} \\
\qquad\qquad \wedge\ \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this})) \\
\qquad\qquad \wedge\ \hat{n}_{argalen} = \textit{getArgValue}(args, \text{“}\textit{length}\text{”}).1.4 \\
\qquad\qquad \wedge\ \hat{s} = \begin{cases} \bot_{string} & \text{if } \; \hat{n}_{arglen} = \bot_{number} \\ \top_{string} & \text{if } \; \hat{n}_{arglen} \neq \mathsf{UIntSingle} \\ \hat{s}_{this}\hat{+}\hat{s}_0\hat{+}...\hat{+}\hat{s}_{n-1} & \text{if } \; \hat{n}_{arglen} = \mathsf{UIntSingle}(n) \end{cases} \\
\qquad\qquad \wedge\ \hat{s}_i = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“}i\text{”}))) \\
\qquad\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \; \hat{s} \not\sqsubseteq \bot_{String} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.indexOf}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}) \\
\quad \text{where } \; \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@pri\hat{m}itive\text{”} \dot{\in} \hat{H}(\hat{l})\} \\
\qquad\qquad \wedge\ \hat{v}_{this} = \bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \quad \textit{// [[DefaultValue]]??} \\
\qquad\qquad \wedge\ \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this})) \\
\qquad\qquad \wedge\ \hat{n} = \begin{cases} \bot_{number} & \text{if } \; \hat{s}_{this} = \bot_{string} \\ \hat{n}' & \text{if } \; \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ \hat{s}_{search} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“}0\text{”}))) \\
\qquad\qquad \wedge\ \hat{n}' = \begin{cases} \bot_{number} & \text{if } \; \hat{s}_{search} = \bot_{string} \\ \hat{n}'' & \text{if } \; \hat{s}_{search} = \mathsf{NumStrSingle}(s_{search}) \vee \hat{s}_{search} = \mathsf{OtherStrSingle}(s_{search}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ \hat{n}_{pos} = \underline{\widehat{\mathsf{toInteger}}}(\textit{getArgValue}(args, \text{“}1\text{”})) \\
\qquad\qquad \wedge\ \hat{n}'' = \begin{cases} \bot_{number} & \text{if } \; \hat{n}_{pos} = \bot_{number} \\ \hat{n}''' & \text{if } \; \hat{n}_{pos} = \mathsf{UIntSingle}(n_{pos}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ n_{start} = min(max(n_{pos}, 0), s_{this}.length) \\
\qquad\qquad \wedge\ \hat{n}''' = \alpha(native.string.indexOf(s_{this}, s_{search}, n_{start})) \quad \textit{// java, scala} \\
\qquad\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{n}))), \hat{C}) & \text{if } \; \hat{n} \not\sqsubseteq \bot_{Number} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.lastIndexOf}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}) \\
\quad \text{where } \; \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@pri\hat{m}itive\text{”} \dot{\in} \hat{H}(\hat{l})\} \\
\qquad\qquad \wedge\ \hat{v}_{this} = \bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \quad \textit{// [[DefaultValue]]??} \\
\qquad\qquad \wedge\ \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this})) \\
\qquad\qquad \wedge\ \hat{n} = \begin{cases} \bot_{number} & \text{if } \; \hat{s}_{this} = \bot_{string} \\ \hat{n}' & \text{if } \; \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ \hat{s}_{search} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“}0\text{”}))) \\
\qquad\qquad \wedge\ \hat{n}' = \begin{cases} \bot_{number} & \text{if } \; \hat{s}_{search} = \bot_{string} \\ \hat{n}'' & \text{if } \; \hat{s}_{search} = \mathsf{NumStrSingle}(s_{search}) \vee \hat{s}_{search} = \mathsf{OtherStrSingle}(s_{search}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ \hat{n}_{pos} = \underline{\widehat{\mathsf{toInteger}}}(\textit{getArgValue}(args, \text{“}1\text{”})) \\
\qquad\qquad \wedge\ \hat{n}'' = \begin{cases} \bot_{number} & \text{if } \; \hat{n}_{pos} = \bot_{number} \\ \hat{n}''' & \text{if } \; \hat{n}_{pos} = \mathsf{UIntSingle}(n_{pos}) \\ \top_{number} & \text{otherwise} \end{cases} \\
\qquad\qquad \wedge\ n_{start} = min(max(n_{pos}, 0), s_{this}.length) \\
\qquad\qquad \wedge\ \hat{n}''' = \alpha(native.string.lastIndexOf(s_{this}, s_{search}, n_{start})) \quad \textit{// java, scala} \\
\qquad\qquad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{n}))), \hat{C}) & \text{if } \; \hat{n} \not\sqsubseteq \bot_{Number} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.localeCompare}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@pri\hat{m}itive\text{”}\dot{\in}\hat{H}(\hat{l})\} \\
& \wedge\ \hat{v}_{this} = \bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \quad \textit{// [[DefaultValue]]??} \\
& \wedge\ \hat{s}_{this} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{this})) \\
& \wedge\ \hat{n} = \begin{cases} \bot_{number} & \text{if } \ \hat{s}_{this} = \bot_{string} \\ \hat{n}' & \text{if } \ \hat{s}_{this} = \textsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \textsf{OtherStrSingle}(s_{this}) \\ \top_{number} & \text{otherwise} \end{cases} \\
& \wedge\ \hat{s}_{that} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“0”}))) \\
& \wedge\ \hat{n}' = \begin{cases} \bot_{number} & \text{if } \ \hat{s}_{that} = \bot_{string} \\ \hat{n}'' & \text{if } \ \hat{s}_{that} = \textsf{NumStrSingle}(s_{that}) \vee \hat{s}_{that} = \textsf{OtherStrSingle}(s_{that}) \\ \top_{number} & \text{otherwise} \end{cases} \\
& \wedge\ \hat{n}'' = \alpha(native.string.compare(s_{this}, s_{that})) \quad \textit{// java, scala} \\
& \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{n}))), \hat{C}) & \text{if } \ \hat{n} \not\sqsubseteq \bot_{Number} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{String.prototype.slice}\text{”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@pri\hat{m}itive\text{”}\dot{\in}\hat{H}(\hat{l})\} \\
& \wedge\ \hat{v}_{this} = \bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \quad \textit{// [[DefaultValue]]??} \\
& \wedge\ \hat{s}_{this} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{this})) \\
& \wedge\ \hat{s} = \begin{cases} \bot_{string} & \text{if } \ \hat{s}_{this} = \bot_{string} \\ \hat{s}' & \text{if } \ \hat{s}_{this} = \textsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \textsf{OtherStrSingle}(s_{this}) \\ \top_{string} & \text{otherwise} \end{cases} \\
& \wedge\ \hat{n}_{start} = \underline{\widehat{\textsf{toInteger}}}(\textit{getArgValue}(args, \text{“0”})) \\
& \wedge\ \hat{s}' = \begin{cases} \bot_{string} & \text{if } \ \hat{n}_{start} = \bot_{number} \\ \hat{s}'' & \text{if } \ \hat{n}_{start} = \textsf{UIntSingle}(n_{start}) \vee \hat{n}_{start} = \textsf{NUIntSingle}(n_{start}) \\ \top_{string} & \text{otherwise} \end{cases} \\
& \wedge\ \hat{n}_{end} = \underline{\widehat{\textsf{toInteger}}}(\textit{getArgValue}(args, \text{“1”})) \\
& \wedge\ \hat{s}'' = \begin{cases} \bot_{string} & \text{if } \ \hat{n}_{end} = \bot_{number} \\ \hat{s}''' & \text{if } \ \hat{n}_{end} = \textsf{UIntSingle}(n_{end}) \vee \hat{n}_{end} = \textsf{NUIntSingle}(n_{end}) \\ \top_{string} & \text{otherwise} \end{cases} \\
& \wedge\ n'_{start} = \begin{cases} max(n_{start} + s_{this}.length, 0) & \text{if } \ n_{start} < 0 \\ min(n_{start}, s_{this}.length) & \text{otherwise} \end{cases} \\
& \wedge\ n'_{end} = \begin{cases} max(n_{end} + s_{this}.length, 0) & \text{if } \ n_{end} < 0 \\ min(n_{end}, s_{this}.length) & \text{otherwise} \end{cases} \\
& \wedge\ \hat{s}''' = \alpha(native.string.slice(s_{this}, s'_{start}, s'_{end})) \quad \textit{// java, scala} \\
& \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{String} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.substring}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad\quad \wedge\ \hat{v}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad\quad \wedge\ \hat{s}_{this} = \widehat{\underline{\mathsf{toString}}}(\widehat{\underline{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad\quad \wedge\ \hat{s} = \begin{cases} \bot_{string} & \text{if } \ \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \ \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\ \hat{n}_{start} = \widehat{\underline{\mathsf{toInteger}}}(\mathit{getArgValue}(args, \text{“}0\text{”}))\\
&\qquad\quad \wedge\ \hat{s}' = \begin{cases} \bot_{string} & \text{if } \ \hat{n}_{start} = \bot_{number}\\ \hat{s}'' & \text{if } \ \hat{n}_{start} = \mathsf{UIntSingle}(n_{start}) \vee \hat{n}_{start} = \mathsf{NUIntSingle}(n_{start})\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\ \hat{n}_{end} = \widehat{\underline{\mathsf{toInteger}}}(\mathit{getArgValue}(args, \text{“}1\text{”}))\\
&\qquad\quad \wedge\ \hat{s}'' = \begin{cases} \bot_{string} & \text{if } \ \hat{n}_{end} = \bot_{number}\\ \hat{s}''' & \text{if } \ \hat{n}_{end} = \mathsf{UIntSingle}(n_{end}) \vee \hat{n}_{end} = \mathsf{NUIntSingle}(n_{end})\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\ n'_{start} = min(max(n_{start},0), s_{this}.length) \wedge n'_{end} = min(max(n_{end},0), s_{this}.length)\\
&\qquad\quad \wedge\ \hat{s}''' = \alpha(native.string.slice(s_{this}, min(n'_{start}, n'_{end}), max(n'_{start}, n'_{end}))) \quad \textit{// java, scala}\\
&\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toLowerCase}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad\quad \wedge\ \hat{v}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad\quad \wedge\ \hat{s}_{this} = \widehat{\underline{\mathsf{toString}}}(\widehat{\underline{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad\quad \wedge\ \hat{s} = \begin{cases} \bot_{string} & \text{if } \ \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \ \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\ \hat{s}' = \alpha(native.string.toLowerCase(s_{this})) \quad \textit{// java, scala}\\
&\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toLocaleLowerCase}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad\quad \wedge\ \hat{v}_{this} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad\quad \wedge\ \hat{s}_{this} = \widehat{\underline{\mathsf{toString}}}(\widehat{\underline{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad\quad \wedge\ \hat{s} = \begin{cases} \bot_{string} & \text{if } \ \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \ \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise} \end{cases}\\
&\qquad\quad \wedge\ \hat{s}' = \alpha(native.string.toLowerCase(s_{this})) \quad \textit{// java, scala}\\
&\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\widehat{\underline{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toUpperCase”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad \wedge \hat{v}_{this} = \bigsqcup\nolimits_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad \wedge \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad \wedge \hat{s} = \begin{cases} \bot_{string} & \text{if } \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{s}' = \alpha(native.string.toUpperCase(s_{this})) \quad \textit{// java, scala}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toLocaleUpperCase”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad \wedge \hat{v}_{this} = \bigsqcup\nolimits_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad \wedge \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad \wedge \hat{s} = \begin{cases} \bot_{string} & \text{if } \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{s}' = \alpha(native.string.toUpperCase(s_{this})) \quad \textit{// java, scala}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“String.prototype.trim”}, args))]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{L}_{prim} = \{\hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \text{“}@\widehat{primitive}\text{”} \dot{\in} \hat{H}(\hat{l})\}\\
&\qquad \wedge \hat{v}_{this} = \bigsqcup\nolimits_{\hat{l}\in\hat{L}_{prim}} \hat{H}(\hat{l})(@\widehat{primitive}).1.2 \quad \textit{// [[DefaultValue]]??}\\
&\qquad \wedge \hat{s}_{this} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{this}))\\
&\qquad \wedge \hat{s} = \begin{cases} \bot_{string} & \text{if } \hat{s}_{this} = \bot_{string}\\ \hat{s}' & \text{if } \hat{s}_{this} = \mathsf{NumStrSingle}(s_{this}) \vee \hat{s}_{this} = \mathsf{OtherStrSingle}(s_{this})\\ \top_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{s}' = \alpha(native.string.trim(s_{this})) \quad \textit{// java, scala}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))), \hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}
$$

11.2.10 Boolean

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Boolean.constructor”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where } \hat{b} = \underline{\widehat{\mathsf{toBoolean}}}(getArgValue(args, \text{“0”})\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge \hat{l}_R = (\hat{a}_{new}, Recent) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge \hat{o}_{new} = \underline{\widehat{\mathsf{NewBoolean}}}(\hat{b})\\
&\qquad \wedge \hat{H}_2 = \hat{H}_l[\hat{l}_R \mapsto \hat{o}_{new}]\\
&\qquad \wedge (\hat{H}_3,\hat{C}_3) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R))), \hat{C}_1)
\end{aligned}
$$

11.2.11 Boolean.prototype

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Boolean.prototype.toString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad \text{where } \hat{L}_{this} = \hat{C}.2\\
&\qquad \wedge \hat{es} = \begin{cases} \{\hat{\mathsf{TypeError}}\} & \text{if } \exists\hat{l}\in\hat{L}_{this} : \hat{H}(\hat{l})(@class).1.2.1.5 \neq \text{“}\hat{Boolean}\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{L}_{bool} = \{\ l \mid l \in \hat{L}_{this} \wedge \hat{H}(\hat{l})(@class).1.2.1.5 = \text{“}\hat{Boolean}\text{”}\ \}\\
&\qquad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{bool}} \hat{H}(\hat{l})(@primitive).1.2.1.3\\
&\qquad \wedge \hat{s} = \begin{cases} \text{“}\hat{true}\text{”} & \text{if } \hat{b} = \hat{\mathsf{true}}\\ \text{“}\hat{false}\text{”} & \text{if } \hat{b} = \hat{\mathsf{false}}\\ \hat{\mathsf{OtherStr}} & \text{if } \hat{b} \sqsubseteq \hat{\mathsf{true}} \wedge \hat{b} \sqsubseteq \hat{\mathsf{false}}\\ \bot_{string} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}))),\hat{C}) & \text{if } \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Boolean.prototype.valueOf}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)\\
&\quad \text{where } \hat{L}_{this} = \hat{C}.2\\
&\qquad \wedge \hat{es} = \begin{cases} \{\hat{\mathsf{TypeError}}\} & \text{if } \exists\hat{l}\in\hat{L}_{this} : \hat{H}(\hat{l})(@class).1.2.1.5 \neq \text{“}\hat{Boolean}\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{L}_{bool} = \{\ l \mid l \in \hat{L}_{this} \wedge \hat{H}(\hat{l})(@class).1.2.1.5 = \text{“}\hat{Boolean}\text{”}\ \}\\
&\qquad \wedge \hat{b} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{bool}} \hat{H}(\hat{l})(@primitive).1.2.1.3\\
&\qquad \wedge \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\mathsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{b}))),\hat{C}) & \text{if } \hat{b} \not\sqsubseteq \bot_{String}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

11.2.12 Number

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where } \wedge \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\mathit{getArgValue}(args, \text{“}length\text{”}))\\
&\qquad \wedge \hat{v}_{arg1} = \mathit{getArgValue}(args, \text{“0”})\\
&\qquad \wedge \hat{v}_1 = \begin{cases} Value(\alpha(0)) & \text{if } \mathsf{UIntSigle}(0) \sqsubseteq \hat{n}_{len}\\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{v}_2 = \begin{cases} Value(\underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{arg1}))) & \text{if } \mathsf{UIntSigle}(1) \neq \hat{n}_{len} \wedge \hat{n}_{len} \not\sqsubseteq \bot_{Number}\\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{v} = \hat{v}_1 \sqcup \hat{v}_2\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, v)),\hat{C}) & \text{if } \hat{v} \not\sqsubseteq \bot_{Value}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.constructor}\text{”}, args))_{\hat{a}_{new}}]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})\\
&\quad \text{where } (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \wedge\ \hat{l}_R = (\hat{a}_{new}, Recent) \quad \textit{// Recency Abstraction}\\
&\qquad \wedge \hat{n}_{len} = (\mathit{getArgValue}(args, \text{“}length\text{”}).1.4\\
&\qquad \wedge \hat{v}_1 = \begin{cases} Value(\alpha(0)) & \text{if } \mathsf{UIntSigle}(0) \sqsubseteq \hat{n}_{len}\\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{v}_2 = \begin{cases} \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“0”}))) & \text{if } \mathsf{UIntSigle}(1) \sqsubseteq \hat{n}_{len}\\ \bot_{Value} & \text{otherwise}\end{cases}\\
&\qquad \wedge \hat{H}_2 = \begin{cases} \hat{H}_1[\hat{l}_R \mapsto \underline{\widehat{\mathsf{NewNumber}}}(\hat{v}_1 \sqcup \hat{v}_2)] & \text{if } \hat{v}_1 \sqcup \hat{v}_2 \sqsubseteq \bot_{Value}\\ \bot_{Heap} & \text{otherwise}\end{cases}\\
&\qquad \wedge (\hat{H}_3,\hat{C}_3) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)),\hat{C}_1) & \text{if } \hat{v}_1 \sqcup \hat{v}_2 \not\sqsubseteq \bot_{Value}\\ (\bot_{heap},\bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

11.2.13 Number.prototype

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{n}_{arglen} = \underline{\widehat{\textsf{toUInt32}}}(\textit{getArgValue}(args, \text{“}\textit{length}\text{”})\\
& \wedge\ \hat{v}_{prim} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2\\
& \wedge\ \hat{L}_{num} = \{\ l \mid l \in \hat{L}_{this} \wedge \hat{H}(\hat{l})(@class).1.2.1.5 = \text{“}Nu\hat{m}ber\text{”}\ \}\\
& \wedge\ \hat{es}_1 = \begin{cases} \{\textsf{Type}\hat{\textsf{E}}\textsf{rror}\} & \text{if } \exists\hat{l} \in \hat{L}_{this} : \hat{H}(\hat{l})(@class).1.2.1.5 \neq \text{“}Nu\hat{m}ber\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ (\hat{v},\hat{es}_2) = \begin{cases} (Value(\underline{\widehat{\textsf{toString}}}(\hat{v}_{prim}.1)), \bot_{Exception}) & \text{if } \ \hat{n}_{arglen} = \textsf{UIntSingle}(0)\\ (Value(\top_{String}), \hat{es}_{arg}) & \text{if } \ \hat{n}_{arglen} = \textsf{UIntSingle}(n_{arglen}) \wedge n_{arglen} > 0\\ (\bot_{Value}, \bot_{Exception}) & \text{if } \ \hat{n}_{arglen} = \bot_{Number}\\ (Value(\top_{String}), \bot_{Exception}) & \text{otherwise}\end{cases}\\
& \wedge\ \hat{es}_{arg} = \begin{cases} \{\textsf{Rang}\hat{\textsf{e}}\textsf{Error}\} & \text{if } \ n_{arglen} < 1 \vee n_{arglen} > 36\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es}_1 \sqcup \hat{es}_2)\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{s})), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{String}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toLocaleString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{aligned}
\text{where }\ & \hat{n}_{arglen} = \underline{\widehat{\textsf{toUInt32}}}(\textit{getArgValue}(args, \text{“}\textit{length}\text{”})\\
& \wedge\ \hat{v}_{prim} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}(\hat{l})(@pri\hat{m}itive).1.2 \wedge \hat{v} = Value(\underline{\widehat{\textsf{toString}}}(\hat{v}_{prim}.1))\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, v), \hat{C}) & \text{if } \ \hat{v} \not\sqsubseteq \bot_{Value}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.valueOf}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{es} = \begin{cases} \{\textsf{Type}\hat{\textsf{E}}\textsf{rror}\} & \text{if } \ \exists\hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@class).1.2.1.5 \neq \text{“}Nu\hat{m}ber\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ \hat{L}_{num} = \{\ l \mid l \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@class).1.2.1.5 = \text{“}Nu\hat{m}ber\text{”}\ \}\\
& \wedge\ \hat{n} = \textstyle\bigsqcup_{\hat{l}\in\hat{L}_{num}} \hat{H}(\hat{l})(@primitive).1.2.1.4\\
& \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = \begin{cases} (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\hat{n})), \hat{C}) & \text{if } \ \hat{s} \not\sqsubseteq \bot_{Number}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise}\end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toFixed}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \textit{getArgValue}(args, \text{“}0\text{”})\\
& \wedge\ \hat{es} = \begin{cases} \{\textsf{Rang}\hat{\textsf{e}}\textsf{Error}\} & \text{if } \ \hat{v}_1 \hat{<} \hat{0} \vee \hat{v}_1 \hat{>} \hat{2}0\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\top_{String})), \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toExponential}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \textit{getArgValue}(args, \text{“}0\text{”})\\
& \wedge\ \hat{es} = \begin{cases} \{\textsf{Rang}\hat{\textsf{e}}\textsf{Error}\} & \text{if } \ \hat{v}_1 \hat{<} \hat{0} \vee \hat{v}_1 \hat{>} \hat{2}0\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\top_{String})), \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toPrecesion}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S}_1)$$

$$\begin{aligned}
\text{where }\ & \hat{v}_1 = \textit{getArgValue}(args, \text{“}0\text{”})\\
& \wedge\ \hat{es} = \begin{cases} \{\textsf{Rang}\hat{\textsf{e}}\textsf{Error}\} & \text{if } \ \hat{v}_1 \hat{<} \hat{1} \vee \hat{v}_1 \hat{>} \hat{2}1\\ \{\} & \text{otherwise}\end{cases}\\
& \wedge\ \hat{S}_1 = \hat{S} \sqcup \underline{\widehat{\textsf{RaiseException}}}(\hat{H},\hat{C},\hat{es})\\
& \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}, Value(\top_{String})), \hat{C})
\end{aligned}$$

11.2.14 Math

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.abs}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge \hat{pv} = \begin{cases}
\hat{n} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{NaN}}, \widehat{\mathsf{UInt}}, \widehat{\mathsf{NUInt}}, \bot_{Number} \right\}\\
\widehat{\mathsf{+Inf}} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}}, \widehat{\mathsf{Inf}} \right\}\\
\alpha(abs(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n_1}), \widehat{\mathsf{NUIntSingle}}(\hat{n_1}) \right\}\\
\top_{Number} & \text{otherwise}
\end{cases}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.acos}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge \hat{pv} = \begin{cases}
\bot_{Number} & \text{if } \hat{n} = \bot_{Number}\\
\widehat{\mathsf{NaN}} & \text{if } \hat{n} \in \left\{ \begin{array}{l} \widehat{\mathsf{NaN}}, \widehat{\mathsf{Inf}}, \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}},\\ \widehat{\mathsf{UIntSingle}}(\hat{n_1}), \widehat{\mathsf{NUIntSingle}}(\hat{n_1}) \mid n_1 < -1 \vee 1 < n_1 \end{array} \right\}\\
\alpha(acos(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n_1}) \mid -1 \geq \hat{n} \leq 1 \right\}\\
\top_{Number} & \text{otherwise}
\end{cases}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.asin}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge \hat{pv} = \begin{cases}
\bot_{Number} & \text{if } \hat{n} = \bot_{Number}\\
\widehat{\mathsf{NaN}} & \text{if } \hat{n} \in \left\{ \begin{array}{l} \widehat{\mathsf{NaN}}, \widehat{\mathsf{Inf}}, \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}},\\ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \mid n_1 < -1 \vee 1 < n_1 \end{array} \right\}\\
\alpha(asin(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \mid -1 \geq \hat{n} \leq 1 \right\}\\
\top_{Number} & \text{otherwise}
\end{cases}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.atan}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge \hat{pv} = \begin{cases}
\bot_{Number} & \text{if } \hat{n} = \bot_{Number}\\
\widehat{\mathsf{NaN}} & \text{if } \hat{n} = \widehat{\mathsf{NaN}}\\
\widehat{\mathsf{NUInt}} & \text{if } \hat{n} = \widehat{\mathsf{Inf}}\\
\widehat{\mathsf{NUIntSingle}}(\tfrac{\pi}{2}) & \text{if } \hat{n} = \widehat{\mathsf{+Inf}}\\
\mathsf{NUIntSingle}(\widehat{-\tfrac{\pi}{2}}) & \text{if } \hat{n} = \widehat{\mathsf{-Inf}}\\
\alpha(atan(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n_1}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \right\}\\
\top_{Number} & \text{otherwise}
\end{cases}\\
&\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Math.atan2}\text{”}, args)\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v}_x = \widehat{getArgValue}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n}_x = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v_x}))\\
&\qquad \wedge\ \hat{v}_y = \widehat{getArgValue}(args, \text{“}\hat{1}\text{”}) \wedge \hat{n}_x = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v_y}))\\
&\qquad \wedge\ \hat{pv}_1 = \begin{cases} \bot_{Number} & \text{if } \hat{n_y} = \bot_{Number} \vee \hat{n_x} = \bot_{Number}\\ \mathsf{N\hat{a}N} & \text{if } \hat{n_y} = \mathsf{N\hat{a}N} \vee \hat{n_x} = \mathsf{N\hat{a}N}\\ \top_{Number} & \text{if } \hat{n_y} = \top_{Number} \vee \hat{n_x} = \top_{Number} \end{cases}\\
&\qquad \wedge\ \hat{pv}_2 = \begin{cases} \mathsf{UInt\hat{S}Ingle}(\hat{0}) & \text{if } \hat{n_y} \in \left\{ \mathsf{U\hat{I}nt}, \mathsf{NU\hat{I}nt}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \right\}\\ & \quad \wedge \hat{n_x} = \mathsf{+\hat{I}nf}\\ \mathsf{U\hat{I}nt} & \text{if } \hat{n_y} \in \left\{ \mathsf{U\hat{I}nt}, \mathsf{NU\hat{I}nt} \right\} \wedge \hat{n_x} = \mathsf{-\hat{I}nf}\\ \mathsf{NUInt\hat{S}Ingle}(\hat{\pi}) & \text{if } \hat{n_y} \in \left\{ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} \leq \hat{n_1} \right\}\\ & \quad \wedge \hat{n_x} = \mathsf{-\hat{I}nf}\\ \mathsf{NUInt\hat{S}Ingle}(\hat{-}\pi) & \text{if } \hat{n_y} \in \left\{ \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < \hat{0} \right\}\\ & \quad \wedge \hat{n_x} = \mathsf{-\hat{I}nf}\\ \mathsf{U\hat{I}nt} & \text{if } \hat{n_y} \in \left\{ \mathsf{U\hat{I}nt}, \mathsf{NU\hat{I}nt} \right\} \wedge \hat{n_x} = \mathsf{\hat{I}nf} \end{cases}\\
&\qquad \wedge\ \hat{pv}_3 = \begin{cases} \mathsf{NUInt\hat{S}ingle}(\frac{\hat{\pi}}{2}) & \text{if } \hat{n_y} = \mathsf{+\hat{I}nf}\\ & \quad \wedge \hat{n_x} \in \left\{ \mathsf{U\hat{I}nt}, \mathsf{NU\hat{I}nt}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \right\}\\ \mathsf{NUInt\hat{S}ingle}(-\frac{\hat{\pi}}{2}) & \text{if } \hat{n_y} = \mathsf{-\hat{I}nf}\\ & \quad \wedge \hat{n_x} \in \left\{ \mathsf{U\hat{I}nt}, \mathsf{NU\hat{I}nt}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \right\}\\ \mathsf{NUInt\hat{S}ingle}(\frac{\hat{\pi}}{4}) & \text{if } \hat{n_y} = \mathsf{+\hat{I}nf} \wedge \hat{n_x} = \mathsf{+\hat{I}nf}\\ \mathsf{NUInt\hat{S}ingle}(\frac{\hat{3\pi}}{4}) & \text{if } \hat{n_y} = \mathsf{+\hat{I}nf} \wedge \hat{n_x} = \mathsf{-\hat{I}nf}\\ \mathsf{NUInt\hat{S}ingle}(\frac{\hat{-\pi}}{4}) & \text{if } \hat{n_y} = \mathsf{-\hat{I}nf} \wedge \hat{n_x} = \mathsf{+\hat{I}nf}\\ \mathsf{NUInt\hat{S}ingle}(\frac{\hat{-3\pi}}{4}) & \text{if } \hat{n_y} = \mathsf{-\hat{I}nf} \wedge \hat{n_x} = \mathsf{-\hat{I}nf}\\ \mathsf{NU\hat{I}nt} & \text{if } \hat{n_y} = \mathsf{\hat{I}nf} \wedge \hat{n_x} \in \left\{ \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf} \right\}\\ \mathsf{NU\hat{I}nt} & \text{if } \hat{n_y} \in \left\{ \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf} \right\} \wedge \hat{n_x} = \mathsf{\hat{I}nf}\\ \mathsf{NU\hat{I}nt} & \text{if } \hat{n_y} = \mathsf{\hat{I}nf} \wedge \hat{n_x} = \mathsf{\hat{I}nf} \end{cases}\\
&\qquad \wedge\ \hat{pv}_4 = \begin{cases} \alpha(atan2(\hat{n_1}, \hat{n_2})) & \text{if } \hat{n_y} \in \left\{ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \right\}\\ & \quad \wedge \hat{n_x} \in \left\{ \mathsf{UInt\hat{S}ingle}(\hat{n_2}), \mathsf{NUInt\hat{S}ingle}(\hat{n_2}) \right\}\\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ \hat{pv} = \hat{pv}_1 \sqcup \hat{pv}_2 \sqcup \hat{pv}_3 \sqcup \hat{pv}_4\\
&\qquad \wedge\ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Math.ceil}\text{”}, args)\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{getArgValue}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge\ \hat{pv} = \begin{cases} \hat{n} & \text{if } \hat{n} \in \left\{ \bot_{Number}, \mathsf{N\hat{a}N}, \mathsf{\hat{I}nf}, \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf}, \mathsf{U\hat{I}nt}, \mathsf{UInt\hat{S}ingle}(\hat{n}) \right\}\\ \alpha(ceil(\hat{n})) & \text{if } \hat{n} = \mathsf{NUInt\hat{S}ingle}(\hat{n})\\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{Math.cos}\text{”}, args)\rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{v} = \widehat{getArgValue}(args, \text{“}0\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad \wedge\ \hat{pv} = \begin{cases} \bot_{Number} & \text{if } \hat{n} = \bot_{Number}\\ \mathsf{N\hat{a}N} & \text{if } \hat{n} \in \left\{ \mathsf{N\hat{a}N}, \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf}, \mathsf{\hat{I}nf} \right\}\\ \alpha(cos(\hat{n})) & \text{if } \hat{n} \in \left\{ \mathsf{UInt\hat{S}ingle}(\hat{n}), \mathsf{NUInt\hat{S}ingle}(\hat{n}) \right\}\\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad \wedge\ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.exp}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where }\ \hat{v} = \widehat{\mathit{getArgValue}}(\widehat{args}, \text{“}0\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad\qquad \wedge \hat{pv} = \begin{cases} \hat{n} & \text{if }\ \hat{n} \in \left\{\ \bot_{Number}, \mathsf{N\hat{a}N}, \mathsf{+\hat{I}nf}, \mathsf{N\hat{U}Int}\ \right\} \\ \mathsf{UInt\hat{S}ignle}(\hat{0}) & \text{if }\ \hat{n} = \mathsf{-\hat{I}nf} \\ \alpha(exp(\hat{n})) & \text{if }\ \hat{n} \in \left\{\ \mathsf{UInt\hat{S}ingle}(\hat{n}), \mathsf{NUInt\hat{S}ingle}(\hat{n})\ \right\} \\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.floor}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where }\ \hat{v} = \widehat{\mathit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad\qquad \wedge \hat{pv} = \begin{cases} \hat{n} & \text{if }\ \hat{n} \in \left\{\ \bot_{Number}, \mathsf{\hat{I}nf}, \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf}, \mathsf{N\hat{a}N}, \mathsf{U\hat{I}nt}, \mathsf{UInt\hat{S}ingle}(\hat{n})\ \right\} \\ \alpha(floor(\hat{n})) & \text{if }\ \hat{n} = \mathsf{NUInt\hat{S}ingle}(\hat{n}) \\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.log}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where }\ \hat{v} = \widehat{\mathit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\
&\qquad\qquad \wedge \hat{pv} = \begin{cases} \bot_{Number} & \text{if }\ \hat{n} = \bot_{Number} \\ \mathsf{N\hat{a}N} & \text{if }\ \hat{n} \in \left\{\ \mathsf{N\hat{a}N}, \mathsf{-\hat{I}nf}, \mathsf{NUInt\hat{S}ingle}(\hat{n}) \mid n < \hat{0}\ \right\} \\ \mathsf{+\hat{I}nf} & \text{if }\ \hat{n} = \mathsf{+\hat{I}nf} \\ \mathsf{-\hat{I}nf} & \text{if }\ \hat{n} = \mathsf{UInt\hat{S}ingle}(\hat{0}) \\ \alpha(log(\hat{n})) & \text{if }\ \hat{n} \in \left\{\ \mathsf{UInt\hat{S}ingle}(\hat{n}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n}\ \right\} \\ \top_{Number} & \text{otherwise} \end{cases}\\
&\qquad\qquad \wedge (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}
$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.max}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{n}_{arglen} = \textit{getArgValue}(args, \text{“}\widehat{length}\text{”}).1.4\\
&\qquad \wedge\, \hat{n}_{arg_i} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“}\hat{i}\text{”})))\\
&\qquad \wedge\, \hat{n}_1 = \begin{cases} \top_{number} & \text{if } \begin{array}{c} \widehat{\mathsf{NUIntSingle}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{inf}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{\text{-}inf}} \sqsubseteq n_{arglen} \\ \vee\, \widehat{\mathsf{NaN}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{UInt}} \sqsubseteq n_{arglen} \end{array} \\ \hat{n}_2 & \text{if } \hat{n}_{arglen} = \widehat{\mathsf{UIntSingle}} \\ \bot_{number} & \text{if } \hat{n}_{arglen} = \bot_{number} \end{cases}\\
&\qquad \wedge\, \hat{n}_2 = \begin{cases} \widehat{\mathsf{\text{-}Inf}} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(0) \\ \hat{n}_3 \sqcup \bigsqcup \hat{N} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge n > 0 \end{cases}\\
&\qquad \wedge\, \hat{n}_3 = \begin{cases} \widehat{\mathsf{NaN}} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge n > 0 \wedge \exists i \in \{1,...,n-1\} : \widehat{\mathsf{NaN}} \sqsubseteq n_{arg_i} \\ \bot_{number} & \text{otherwise} \end{cases}\\
&\qquad \wedge\, \hat{N} = \left\{\; n_{arg_i} \mid n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge i \in \{0,...,n-1\} \wedge \forall j \in \{0,...,n-1\} : n_{arg_i} \hat{\geq} n_{arg_j} \;\right\}\\
&\qquad \wedge\, (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{n}_1)), \hat{C})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.min}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\
&\quad \text{where } \hat{n}_{arglen} = \textit{getArgValue}(args, \text{“}\widehat{length}\text{”}).1.4\\
&\qquad \wedge\, \hat{n}_{arg_i} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\textit{getArgValue}(args, \text{“}\hat{i}\text{”})))\\
&\qquad \wedge\, \hat{n}_1 = \begin{cases} \top_{number} & \text{if } \begin{array}{c} \widehat{\mathsf{NUIntSingle}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{inf}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{\text{-}inf}} \sqsubseteq n_{arglen} \\ \vee\, \widehat{\mathsf{NaN}} \sqsubseteq n_{arglen} \vee \widehat{\mathsf{UInt}} \sqsubseteq n_{arglen} \end{array} \\ \hat{n}_2 & \text{if } \hat{n}_{arglen} = \widehat{\mathsf{UIntSingle}} \\ \bot_{number} & \text{if } \hat{n}_{arglen} = \bot_{number} \end{cases}\\
&\qquad \wedge\, \hat{n}_2 = \begin{cases} \widehat{\mathsf{Inf}} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(0) \\ \hat{n}_3 \sqcup \bigsqcup \hat{N} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge n > 0 \end{cases}\\
&\qquad \wedge\, \hat{n}_3 = \begin{cases} \widehat{\mathsf{NaN}} & \text{if } n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge n > 0 \wedge \exists i \in \{1,...,n-1\} : \widehat{\mathsf{NaN}} \sqsubseteq n_{arg_i} \\ \bot_{number} & \text{otherwise} \end{cases}\\
&\qquad \wedge\, \hat{N} = \left\{\; n_{arg_i} \mid n_{arglen} = \widehat{\mathsf{UIntSingle}}(n) \wedge i \in \{0,...,n-1\} \wedge \forall j \in \{0,...,n-1\} : n_{arg_i} \hat{\leq} n_{arg_j} \;\right\}\\
&\qquad \wedge\, (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{n}_1)), \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.pow}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{aligned}
\text{where } & \hat{v}_x = \mathit{getAr\hat{g}Value}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n}_x = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v_x})) \\
& \wedge \hat{v}_y = \mathit{getAr\hat{g}Value}(args, \text{“}\hat{1}\text{”}) \wedge \hat{n}_y = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v_y}))
\end{aligned}$$

$$\wedge\ \hat{pv}_1 = \begin{cases}
\bot_{Number} & \text{if } \hat{n_y} = \bot_{Number} \vee \hat{n_x} = \bot_{Number} \\
\mathsf{UInt\hat{S}ingle}(\hat{1}) & \text{if } \hat{n_y} = \mathsf{UInt\hat{S}ingle}(\hat{0}) \\
\mathsf{N\hat{a}N} & \text{if } \hat{n_y} = \mathsf{N\hat{a}N} \\
\mathsf{N\hat{a}N} & \text{if } \hat{n_x} = \mathsf{N\hat{a}N} \wedge \hat{n_y} \neq \mathsf{UInt\hat{S}ingle}(\hat{0}) \\
\top_{Number} & \text{if } \hat{n_x} = \top_{Number} \vee \hat{n_y} = \top_{Number} \\
\mathsf{N\hat{a}N} & \text{if } \hat{n_x} \in \left\{ \ \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < 0 \ \right\} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \neg isInt(\hat{n_1}) \ \right\}
\end{cases}$$

$$\wedge\ \hat{pv}_2 = \begin{cases}
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} = \mathsf{UInt\hat{S}ingle}(\hat{0}) \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{+\hat{I}nf}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n_1} \ \right\} \\
\mathsf{+\hat{I}nf} & \text{if } \hat{n_x} = \mathsf{UInt\hat{S}ingle}(\hat{0}) \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{-\hat{I}nf}, \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < \hat{0} \ \right\}
\end{cases}$$

$$\wedge\ \hat{pv}_3 = \begin{cases}
\mathsf{+\hat{I}nf} & \text{if } \hat{n_x} = \mathsf{+\hat{I}nf} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{+\hat{I}nf}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n_1} \ \right\} \\
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} = \mathsf{+\hat{I}nf} \wedge \hat{n_y} \in \left\{ \ \mathsf{-\hat{I}nf}, \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < \hat{0} \ \right\} \\
\mathsf{-\hat{I}nf} & \text{if } \hat{n_x} = \mathsf{-\hat{I}nf} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n_1} \wedge isOdd(\hat{n_1}) \ \right\} \\
\mathsf{+\hat{I}nf} & \text{if } \hat{n_x} = \mathsf{-\hat{I}nf} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{+\hat{I}nf}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n_1} \wedge \neg isOdd(\hat{n_1}) \ \right\} \\
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} = \mathsf{-\hat{I}nf} \wedge \hat{n_y} \in \left\{ \ \mathsf{-\hat{I}nf}, \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < \hat{0} \ \right\} \\
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} = \mathsf{\hat{I}nf} \wedge \hat{n_y} \in \left\{ \ \mathsf{-\hat{I}nf}, \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < \hat{0} \ \right\} \\
\mathsf{+\hat{I}nf} & \text{if } \hat{n_x} = \mathsf{\hat{I}nf} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{+\hat{I}nf}, \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{0} < \hat{n_1} \wedge \neg isOdd(\hat{n_1}) \ \right\}
\end{cases}$$

$$\wedge\ \hat{pv}_4 = \begin{cases}
\mathsf{+\hat{I}nf} & \text{if } \hat{v_{xa}}.1.4 \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < -1 \vee 1 < \hat{n_1} \ \right\} \\
& \quad \wedge \hat{n_y} = \mathsf{+\hat{I}nf} \\
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid -1 < \hat{n_1} < 1 \ \right\} \\
& \quad \wedge \hat{n_y} = \mathsf{+\hat{I}nf} \\
\mathsf{UInt\hat{S}ingle}(\hat{0}) & \text{if } \hat{n_x} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid \hat{n_1} < -1 \vee 1 < \hat{n_1} \ \right\} \\
& \quad \wedge \hat{n_y} = \mathsf{-\hat{I}nf} \\
\mathsf{+\hat{I}nf} & \text{if } \hat{n_x} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \mid -1 < \hat{n_1} < 1 \ \right\} \\
& \quad \wedge \hat{n_y} = \mathsf{-\hat{I}nf} \\
\mathsf{N\hat{a}N} & \text{if } \hat{n_x} = \mathsf{UInt\hat{S}ingle}(\hat{1}) \wedge \hat{n_y} \in \left\{ \ \mathsf{\hat{I}nf}, \mathsf{+\hat{I}nf}, \mathsf{-\hat{I}nf} \ \right\}
\end{cases}$$

$$\wedge\ \hat{pv}_5 = \begin{cases}
\alpha(pow(\hat{n_1}, \hat{n_2})) & \text{if } \hat{n_x} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_1}), \mathsf{NUInt\hat{S}ingle}(\hat{n_1}) \ \right\} \\
& \quad \wedge \hat{n_y} \in \left\{ \ \mathsf{UInt\hat{S}ingle}(\hat{n_2}), \mathsf{NUInt\hat{S}ingle}(\hat{n_2}) \ \right\} \\
\top_{Number} & \text{otherwise}
\end{cases}$$

$$\begin{aligned}
& \wedge\ \hat{pv} = \hat{pv}_1 \sqcup \hat{pv}_2 \sqcup \hat{pv}_3 \sqcup \hat{pv}_4 \sqcup \hat{pv}_5 \\
& \wedge\ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})), \hat{C})
\end{aligned}$$

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.random}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\top_{Number})),\hat{C})\end{aligned}$$

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.round}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\ &\qquad\quad \wedge\ \hat{pv} = \begin{cases} \hat{n} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{NaN}}, \widehat{\mathsf{Inf}}, \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}}, \widehat{\mathsf{UInt}}, \widehat{\mathsf{NUInt}}, \bot_{Number} \right\}\\ \alpha(round(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\widehat{n_1}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \right\}\\ \top_{Number} & \text{otherwise}\end{cases}\\ &\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})),\hat{C})\end{aligned}$$

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.sin}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\ &\qquad\quad \wedge\ \hat{pv} = \begin{cases} \bot_{Number} & \text{if } \hat{n} = \bot_{Number}\\ \widehat{\mathsf{NaN}} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{NaN}}, \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}}, \widehat{\mathsf{Inf}} \right\}\\ \alpha(sin(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \right\}\\ \top_{Number} & \text{otherwise}\end{cases}\\ &\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})),\hat{C})\end{aligned}$$

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.sqrt}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\ &\qquad\quad \wedge\ \hat{pv} = \begin{cases} \hat{n} & \text{if } \hat{n} \in \left\{ \bot_{Number}, \widehat{\mathsf{NaN}}, \widehat{\mathsf{+Inf}} \right\}\\ \widehat{\mathsf{NaN}} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{-Inf}}, \widehat{\mathsf{NUIntSingle}}(\hat{n}) \mid \hat{n} < 0 \right\}\\ \alpha(sqrt(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \mid 0 \leq \hat{n} \right\}\\ \top_{Number} & \text{otherwise}\end{cases}\\ &\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})),\hat{C})\end{aligned}$$

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Math.tan}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } \hat{v} = \widehat{\textit{getArgValue}}(args, \text{“}\hat{0}\text{”}) \wedge \hat{n} = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}))\\ &\qquad\quad \wedge\ \hat{pv} = \begin{cases} \hat{n} & \text{if } \hat{n} \in \left\{ \bot_{Number}, \widehat{\mathsf{NaN}} \right\}\\ \widehat{\mathsf{NaN}} & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{Inf}}, \widehat{\mathsf{+Inf}}, \widehat{\mathsf{-Inf}} \right\}\\ \alpha(sqrt(\hat{n})) & \text{if } \hat{n} \in \left\{ \widehat{\mathsf{UIntSingle}}(\hat{n}), \widehat{\mathsf{NUIntSingle}}(\hat{n}) \right\}\\ \top_{Number} & \text{otherwise}\end{cases}\\ &\qquad\quad \wedge\ (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{pv})),\hat{C})\end{aligned}$$

11.2.15 Date

$$\begin{aligned}&\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})\\ &\quad\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \top_{String}),\hat{C})\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Date.constructor”}, args)_{\hat{a}_{new}} \rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_3,\hat{C}_3),\hat{S})$$

$$\begin{aligned}
\text{where } & \hat{l}_R = (\hat{a}_{new}, Recent) \wedge (\hat{H}_1,\hat{C}_1) = \underline{\widehat{\mathsf{Oldify}}}(\hat{H},\hat{C},\hat{a}_{new}) \quad \textit{// Recency Abstraction}\\
& \wedge\, \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H}_1,\hat{C}_1,\text{“}length\text{”}))\\
& \wedge\, \hat{pv}_1 = \underline{\widehat{\mathsf{toPrimitive}}}(\underline{\widehat{\mathsf{getArgValue}}}(\hat{H}_1,\hat{C}_1,\text{“0”}))\\
& \wedge\, \hat{n}_{prim_1} = \begin{cases} \hat{n}_{prim_3} \sqcup \hat{n}_{prim_4} & \text{if } \mathsf{UIntSingle}(1) \sqsubseteq \hat{n}_{arglen}\\ \bot_{Number} & \text{otherwise} \end{cases}\\
& \wedge\, \hat{n}_{prim_2} = \begin{cases} \top_{Number} & \text{if } \mathsf{UIntSingle}(n) \sqsubseteq \hat{n}_{arglen} \wedge n \neq 1\\ \bot_{Number} & \text{otherwise} \end{cases}\\
& \wedge\, \hat{n}_{prim_3} = \begin{cases} \top_{Number} & \text{if } \hat{pv}_1.5 \not\sqsubseteq \bot_{String} \quad \textit{// parse}\\ \bot_{Number} & \text{otherwise} \end{cases}\\
& \wedge\, \hat{pv}_{nonstr} = PValue(\hat{v}_1.1, \hat{v}_1.2, \hat{v}_1.3, \hat{v}_1.4, \bot_{String})\\
& \wedge\, \hat{n}_{prim_4} = \begin{cases} \underline{\widehat{\mathsf{toNumber}}}(\hat{v}_{nonstr}) & \text{if } \hat{pv}_{nonstr} \not\sqsubseteq \bot_{PValue}\\ \bot_{Number} & \text{otherwise} \end{cases}\\
& \wedge\, \hat{n}_{prim} = \hat{n}_{prim_1} \sqcup \hat{n}_{prim_2}\\
& \wedge\, \hat{H}_2 = \hat{H}_1[\hat{l}_R \mapsto \underline{\widehat{\mathsf{NewDate}}}(\hat{n}_{prim})]\\
& \wedge\, (\hat{H}_3,\hat{C}_3) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_2, Value(\hat{l}_R)), \hat{C}_1) & \text{if } \hat{n}_{prim} \not\sqsubseteq \bot_{Number}\\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Date.parse”}, args) \rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \top_{Number}), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}\llbracket \mathsf{BuiltintCall}(\text{“Date.now”}, args) \rrbracket((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \top_{Number}), \hat{C})$$

11.2.16 Date.prototype

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toDateString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toTimeString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleDateString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleTimeString}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H},\top_{String}),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.valueOf}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\begin{aligned}\text{where } & \hat{v} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}(\hat{l})(\widehat{@primitive}) \\ & (\hat{H}_1,\hat{C}_1) = (\hat{H}[\widehat{\#PureLocal}_R \mapsto \hat{H}(\widehat{\#PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getTime}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\begin{aligned}\text{where } & \hat{v} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}(\hat{l})(\widehat{@primitive}) \\ & (\hat{H}_1,\hat{C}_1) = (\hat{H}[\widehat{\#PureLocal}_R \mapsto \hat{H}(\widehat{\#PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getFullYear}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1,Value(\top_{Number})),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCFullYear}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\begin{aligned}\text{where } & \hat{n}_{time} = \hat{H}(\hat{l})(\widehat{@primitive}).1.4 \\ & \hat{v} = \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \mathsf{UIntSingle}(n_{time}) \vee \hat{n}_{time} = \mathsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases} \\ & \hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(YEAR)) \quad \textit{// java, scala} \\ & (\hat{H}_1,\hat{C}_1) = (\hat{H}[\widehat{\#PureLocal}_R \mapsto \hat{H}(\widehat{\#PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getMonth}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCMonth}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{aligned}
\text{where }\ & \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4 \\
& \hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \textsf{UIntSingle}(n_{time}) \vee \hat{n}_{time} = \textsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases} \\
& \hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(MONTH)) \quad \textit{// java, scala} \\
& (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getDate}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCDate}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{aligned}
\text{where }\ & \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4 \\
& \hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \textsf{UIntSingle}(n_{time}) \vee \hat{n}_{time} = \textsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases} \\
& \hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(DAY\_OF\_MONTH)) \quad \textit{// java, scala} \\
& (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getDay}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getUTCDay}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$

$$\begin{aligned}
\text{where }\ & \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4 \\
& \hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \textsf{UIntSingle}(n_{time}) \vee \hat{n}_{time} = \textsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases} \\
& \hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(DAY\_OF\_WEEK)) \quad \textit{// java, scala} \\
& (\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getHours”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getUTCHours”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4$$
$$\hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \mathsf{UIntSingle}(n_{time}) \lor \hat{n}_{time} = \mathsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases}$$
$$\hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(HOURS)) \quad \textit{// java, scala}$$
$$(\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getMinutes”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getUTCMinutes”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4$$
$$\hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \mathsf{UIntSingle}(n_{time}) \lor \hat{n}_{time} = \mathsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases}$$
$$\hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(MINUTE)) \quad \textit{// java, scala}$$
$$(\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getSeconds”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } (\hat{H}_1,\hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})),\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getUTCSeconds”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_1,\hat{C}_1),\hat{S})$$
$$\text{where } \hat{n}_{time} = \hat{H}(\hat{l})(@\widehat{primitive}).1.4$$
$$\hat{v} = \bigsqcup_{\hat{l}\in\hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \mathsf{UIntSingle}(n_{time}) \lor \hat{n}_{time} = \mathsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases}$$
$$\hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(SECOND)) \quad \textit{// java, scala}$$
$$(\hat{H}_1,\hat{C}_1) = (\hat{H}[\#\widehat{PureLocal}_R \mapsto \hat{H}(\#\widehat{PureLocal}_R)[@return \mapsto \hat{v}]],\hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getMilliseconds”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$
$$\text{where } (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getUTCMilliseconds”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$
$$\begin{aligned}
\text{where } & \hat{n}_{time} = \hat{H}(\hat{l})(\hat{@primitive}).1.4 \\
& \hat{v} = \bigsqcup_{\hat{l} \in \hat{C}.2} \begin{cases} \bot_{Value} & \text{if } \hat{n}_{time} \sqsubseteq \bot_{number} \\ \hat{v}_{get} & \text{if } \hat{n}_{time} = \mathsf{UIntSingle}(n_{time}) \vee \hat{n}_{time} = \mathsf{NUIntSingle}(n_{time}) \\ Value(\top_{Number}) & \text{otherwise} \end{cases} \\
& \hat{v}_{get} = \alpha(native.Calendar(n_{time}).get(\mathit{MILLISECOND})) \quad \textit{// java, scala} \\
& (\hat{H}_1, \hat{C}_1) = (\hat{H}[\hat{\#PureLocal}_R \mapsto \hat{H}(\hat{\#PureLocal}_R)[@return \mapsto \hat{v}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.getTimezoneOffset”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$
$$\text{where } (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, Value(\top_{Number})), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.setTime”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S})$$
$$\begin{aligned}
\text{where } & \hat{n}_{time} = \underline{\widehat{\mathsf{TimeClip}}}(\underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“0”})))) \\
& \hat{H}_1 = \bigsqcup_{\hat{l} \in \hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\hat{@primitive} \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2, \hat{C}_2) = (\hat{H}_1[\hat{\#PureLocal}_R \mapsto \hat{H}_1(\hat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.setMilliseconds”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S})$$
$$\begin{aligned}
\text{where } & \hat{H}_1 = \bigsqcup_{\hat{l} \in \hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\hat{@primitive} \mapsto \top_{Number}]] \\
& \wedge (\hat{H}_2, \hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, \top_{Number}), \hat{C}) & \text{if } \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“Date.prototype.setUTCMilliseconds”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S})$$
$$\begin{aligned}
\text{where } & \hat{n}_1 = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“0”}))) \\
& \hat{n}_{time} = \begin{cases} \bot_{Value} & \text{if } \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
& \hat{n}_{naitve} = \alpha(native.Calendar(n_{time}).set(\mathit{MILLISECOND}, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
& \hat{H}_1 = \bigsqcup_{\hat{l} \in \hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\hat{@primitive} \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2, \hat{C}_2) = (\hat{H}_1[\hat{\#PureLocal}_R \mapsto \hat{H}_1(\hat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setSeconds}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{aligned}
\text{where } \hat{H}_1 &= \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \top_{Number}]] \\
\wedge\ (\hat{H}_2,\hat{C}_2) &= \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, \top_{Number}), \hat{C}) & \text{if } \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCSeconds}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{aligned}
\text{where } \hat{n}_1 &= \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“0”}))) \\
\hat{n}_{time} &= \begin{cases} \bot_{Value} & \text{if } \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
\hat{n}_{naitve} &= \alpha(native.Calendar(n_{time}).set(\mathit{SECOND}, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
\hat{H}_1 &= \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \hat{n}_{time}]] \\
(\hat{H}_2,\hat{C}_2) &= (\hat{H}_1[\widehat{\#PureLocal}_R \mapsto \hat{H}_1(\widehat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setMinutes}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{aligned}
\text{where } \hat{H}_1 &= \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \top_{Number}]] \\
\wedge\ (\hat{H}_2,\hat{C}_2) &= \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, \top_{Number}), \hat{C}) & \text{if } \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCMinutes}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{aligned}
\text{where } \hat{n}_1 &= \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“0”}))) \\
\hat{n}_{time} &= \begin{cases} \bot_{Value} & \text{if } \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
\hat{n}_{naitve} &= \alpha(native.Calendar(n_{time}).set(\mathit{MINUTE}, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
\hat{H}_1 &= \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \hat{n}_{time}]] \\
(\hat{H}_2,\hat{C}_2) &= (\hat{H}_1[\widehat{\#PureLocal}_R \mapsto \hat{H}_1(\widehat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setHours}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{aligned}
\text{where } \hat{H}_1 &= \textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \top_{Number}]] \\
\wedge\ (\hat{H}_2,\hat{C}_2) &= \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, \top_{Number}), \hat{C}) & \text{if } \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCHours}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{n}_1 = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“}0\text{”}))) \\
& \hat{n}_{time} = \begin{cases} \bot_{Value} & \text{if } \ \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \ \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
& \hat{n}_{naitve} = \alpha(native.Calendar(n_{time}).set(HOURS, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
& \hat{H}_1 = \bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\widehat{\#PureLocal}_R \mapsto \hat{H}_1(\widehat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]],\hat{C})
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setDate}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{H}_1 = \bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \top_{Number}]] \\
& \wedge\, (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1,\top_{Number}),\hat{C}) & \text{if } \ \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCDate}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{n}_1 = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“}0\text{”}))) \\
& \hat{n}_{time} = \begin{cases} \bot_{Value} & \text{if } \ \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \ \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
& \hat{n}_{naitve} = \alpha(native.Calendar(n_{time}).set(DAY\_OF\_MONTH, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
& \hat{H}_1 = \bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\widehat{\#PureLocal}_R \mapsto \hat{H}_1(\widehat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]],\hat{C})
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setMonth}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{H}_1 = \bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \top_{Number}]] \\
& \wedge\, (\hat{H}_2,\hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1,\top_{Number}),\hat{C}) & \text{if } \ \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap},\bot_{context}) & \text{otherwise} \end{cases}
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCMonth}\text{”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_2,\hat{C}_2),\hat{S})$$

$$\begin{array}{ll}
\text{where} & \hat{n}_1 = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“}0\text{”}))) \\
& \hat{n}_{time} = \begin{cases} \bot_{Value} & \text{if } \ \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if } \ \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
& \hat{n}_{naitve} = \alpha(native.Calendar(n_{time}).set(MONTH, n_1).getTimeInMills()) \quad \textit{// java, scala} \\
& \hat{H}_1 = \bigsqcup_{\hat{l}\in\hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[\widehat{@primitive} \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2,\hat{C}_2) = (\hat{H}_1[\widehat{\#PureLocal}_R \mapsto \hat{H}_1(\widehat{\#PureLocal}_R)[@return \mapsto \hat{n}_{time}]],\hat{C})
\end{array}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setFullYear}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S})$$

$$\begin{aligned}
\text{where}\ \ & \hat{H}_1 = \textstyle\bigsqcup_{\hat{l} \in \hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[@pri\hat{m}itive \mapsto \top_{Number}]] \\
& \wedge (\hat{H}_2, \hat{C}_2) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_1, \top_{Number}), \hat{C}) & \text{if}\ \ \hat{H}_1 \not\sqsubseteq \bot_{Heap} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.setUTCFullYear}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_2, \hat{C}_2), \hat{S})$$

$$\begin{aligned}
\text{where}\ \ & \hat{n}_1 = \underline{\widehat{\mathsf{toNumber}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\mathit{getArgValue}(args, \text{“}0\text{”}))) \\
& \hat{n}_{time} = \begin{cases} \bot_{Value} & \text{if}\ \ \hat{n}_1 \sqsubseteq \bot_{number} \\ \underline{\widehat{\mathsf{TimeClip}}}(\hat{n}_{native}) & \text{if}\ \ \hat{n}_1 = \mathsf{UIntSingle}(n_1) \vee \hat{n}_1 = \mathsf{NUIntSingle}(n_1) \\ \top_{Number} & \text{otherwise} \end{cases} \\
& \hat{n}_{naitve} = \alpha(native.Calendar(n_{time}).set(\mathit{YEAR}, n_1).getTimeInMills()) \quad \mathit{//\,java,\ scala} \\
& \hat{H}_1 = \textstyle\bigsqcup_{\hat{l} \in \hat{C}.2} \hat{H}[\hat{l} \mapsto \hat{H}(\hat{l})[@pri\hat{m}itive \mapsto \hat{n}_{time}]] \\
& (\hat{H}_2, \hat{C}_2) = (\hat{H}_1[\#Pur\hat{e}Local_R \mapsto \hat{H}_1(\#Pur\hat{e}Local_R)[@return \mapsto \hat{n}_{time}]], \hat{C})
\end{aligned}$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toUTCString}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$

$$\text{where}\ \ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \top_{String}), \hat{C})$$

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toISOString}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$

$$\text{where}\ \ (\hat{H}_1, \hat{C}_1) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, \top_{String}), \hat{C})$$

11.2.17 Error

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Error.constructor}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_4, \widehat{CC}_4), \hat{S})$$

$$\begin{aligned}
\text{where}\ \ & \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“}0\text{”}) \wedge \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\hat{E}rr_O \\
& \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if}\ \ \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if}\ \ \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
& \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4, \hat{C}_4) = (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)), \hat{C})
\end{aligned}$$

11.2.18 Error.prototype

$$\hat{\mathcal{I}}_{cp}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Error.prototype.toString}\text{”}, args)]\!]((\hat{H}, \hat{C}), \hat{S}) = ((\hat{H}_1, \hat{C}_1), \hat{S})$$

$$\begin{aligned}
\text{where}\ \ & \hat{L}_{this} = \hat{C}.2 \wedge \hat{v}_{name} = \textstyle\bigsqcup_{l \in L_{this}} \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}na\hat{m}e\text{”}) \wedge \hat{v}_{msg} = \textstyle\bigsqcup_{l \in L_{this}} \underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}mes\hat{s}age\text{”}) \\
& \wedge \hat{s}_1 = \begin{cases} \text{“}Er\hat{r}or\text{”} & \text{if}\ \ v_{name}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{string} & \text{otherwise} \end{cases} \\
& \wedge \hat{s}_2 = \underline{\widehat{\mathsf{toString}}}(\widehat{\mathsf{PValue}}(\bot_{undef}, \hat{v}_{name}.1.2, \hat{v}_{name}.1.3, \hat{v}_{name}.1.4, \hat{v}_{name}.1.5)) \\
& \wedge \hat{s}_{name} = \hat{s}_1 \sqcup \hat{s}_2 \\
& \wedge \hat{s}_3 = \begin{cases} \hat{\text{“”}} & \text{if}\ \ v_{msg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{string} & \text{otherwise} \end{cases} \\
& \wedge \hat{s}_4 = \underline{\widehat{\mathsf{toString}}}(\widehat{\mathsf{PValue}}(\bot_{undef}, \hat{v}_{msg}.1.2, \hat{v}_{msg}.1.3, \hat{v}_{msg}.1.4, \hat{v}_{msg}.1.5)) \\
& \wedge \hat{s}_{msg} = \hat{s}_3 \sqcup \hat{s}_4 \\
& \wedge \hat{s}_5 = \begin{cases} \hat{s}_{msg} & \text{if}\ \ \hat{\text{“”}} \sqsubseteq \hat{s}_{name} \\ \bot_{string} & \text{otherwise} \end{cases} \quad \wedge \hat{s}_6 = \begin{cases} \hat{s}_{name} & \text{if}\ \ \hat{\text{“”}} \sqsubseteq \hat{s}_{msg} \\ \bot_{string} & \text{otherwise} \end{cases} \quad \wedge\ \hat{s}_7 = \hat{s}_{name} + \text{“}\hat{:}\text{”} + \hat{s}_{msg} \\
& \wedge \hat{s}_{ret} = \hat{s}_5 \sqcup \hat{s}_6 \sqcup \hat{s}_7 \\
& \wedge (\hat{H}_1, \hat{C}_1) = \begin{cases} (\underline{\widehat{\mathsf{ReturnStore}}}(\hat{H}, Value(\hat{s}_{ret})), \hat{C}) & \text{if}\ \ \hat{s}_{ret} \not\sqsubseteq \bot_{String} \\ (\bot_{heap}, \bot_{context}) & \text{otherwise} \end{cases}
\end{aligned}$$

11.2.19 EvalError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“EvalError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{EvalErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.2.20 RangeError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“RangeError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{RangeErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.2.21 ReferenceError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“ReferenceError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{RefErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.2.22 SyntaxError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“SyntaxError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{SyntaxErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.2.23 TypeError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“TypeError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{TypeErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.2.24 URIError

$$\begin{array}{l}
\hat{\mathcal{I}}_{cp}[\![\textsf{BuiltintCall}(\text{“URIError.constructor”}, args)]\!]((\hat{H},\hat{C}),\hat{S}) = ((\hat{H}_4,\hat{C}_4),\hat{S}) \\
\quad \text{where } \hat{v}_{arg} = \mathit{getArgValue}(args, \text{“0”}) \wedge \hat{s} = \underline{\widehat{\textsf{toString}}}(\underline{\widehat{\textsf{toPrimitive}}}(\hat{v}_{arg})) \wedge \hat{l}_e = \#\widehat{URIErr}_O \\
\qquad \wedge \hat{H}_1 = \begin{cases} \hat{H}[\hat{l}_e \mapsto \hat{H}(\hat{l}_e)[message \mapsto \hat{s}]] & \text{if } \hat{v}_{arg}.1.1 \sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \wedge \hat{H}_2 = \begin{cases} \hat{H} & \text{if } \hat{v}_{arg}.1.1 \not\sqsubseteq \bot_{undef} \\ \bot_{Heap} & \text{otherwise} \end{cases} \\
\qquad \wedge \hat{H}_3 = \hat{H}_1 \sqcup \hat{H}_2 \wedge (\hat{H}_4,\hat{C}_4) = (\underline{\widehat{\textsf{ReturnStore}}}(\hat{H}_3, Value(\hat{l}_e)),\hat{C})
\end{array}$$

11.3 Access Analysis

11.3.1 Global

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Global.parseInt}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } \ LP_1 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Global.endcodeURIComponent}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \ \hat{es} = \{\mathsf{URIError}\}\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{URIError.isNaN}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } \ LP_1 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{URIError.isFinite}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } \ LP_1 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Global.parseInt}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \ LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{”}0\text{”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C},\text{”}1\text{”})\\
&\qquad\qquad LP_3 = getArgValue_{use}(\hat{H},\hat{C},\text{”}length\text{”})
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Global.endcodeURIComponent}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \ \hat{es} = \{\mathsf{URIError}\}\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{URIError.isNaN}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \ LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{”}0\text{”})\\
&\qquad\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{URIError.isFinite}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \ LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{”}0\text{”})\\
&\qquad\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

11.3.2 Object

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \ \hat{v} = getArgValue(\hat{H},\hat{C},\text{“}0\text{”})\\
&\qquad\qquad (LP_1,\hat{es}) = \begin{cases} (\underline{\widehat{\mathsf{toObject}}}_{def}(\hat{H},\hat{C},\hat{v}_{new},\hat{a}_1),\hat{es}'_1) & \text{if } \ \hat{v}.1.3 \not\sqsubseteq \bot_{Bool} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{Number} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{String}\\ (\{\},\{\}) & \text{otherwise} \end{cases}\\
&\qquad\qquad \hat{v}_{new} = Value(PValue(\bot_{Undef},\bot_{Null},\hat{v}.1.3,\hat{v}.1.4,\hat{v}.1.5),\hat{v}.2)\\
&\qquad\qquad \hat{es}' = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \begin{cases} LP'_2 \cup \bigcup_{s \in \underline{\widehat{\mathsf{NewObject}}}_{def}} \{\ \langle \hat{l}_R, s\rangle\ \} & \text{if } \ \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP'_2 = \underline{\widehat{\mathsf{Oldify}}}_{def}((\hat{H},\hat{C},\hat{v}_{new},\hat{a}_1)\\
&\qquad\qquad \hat{l}_R = (\hat{a}_1,\mathsf{Recent})\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_4 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

where $\hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})$

$$(LP_1, \hat{es}) = \begin{cases} (\bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\hat{P}_1\cup\hat{P}_2\cup\hat{P}_3}\{\ \langle\hat{l}, s\rangle\ \}, \hat{es}') & \text{if } \ \hat{v}.1.3 \not\sqsubseteq \bot_{Bool} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{Number} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{String} \\ (\{\},\{\}) & \text{otherwise} \end{cases}$$

$$\hat{v}_{new} = Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}.1.3, \hat{v}.1.4, \hat{v}.1.5), \hat{v}.2)$$

$$\hat{P}_1 = \begin{cases} \underline{\widehat{\mathsf{NewBool}}}_{def} & \text{if } \ \hat{v}_{new}.1.3 \not\sqsubseteq \bot_{Bool} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{P}_2 = \begin{cases} \underline{\widehat{\mathsf{NewNumber}}}_{def} & \text{if } \ \hat{v}_{new}.1.4 \not\sqsubseteq \bot_{Number} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{P}_3 = \begin{cases} \underline{\widehat{\mathsf{NewString}}}_{def}(\hat{v}_{new}.1.5) & \text{if } \ \hat{v}_{new}.1.5 \not\sqsubseteq \bot_{String} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{es}' = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null} \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_2 = \begin{cases} \bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\underline{\widehat{\mathsf{NewObject}}}_{def}}\{\ \langle\hat{l}, s\rangle\ \} & \text{if } \ \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{l}_R = (\hat{a}_1, \mathsf{Recent})$$

$$LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})$$

$$LP_4 = \{\ \langle\#\widehat{PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getPrototypeOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

where $\hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})$

$$\hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})$$

$$LP_2 = \{\ \langle\#\widehat{PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

where $\hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})$

$$\hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{l}_R = (\hat{a}_1, \mathsf{Recent})$$

$$LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1)$$

$$LP_2 = \{\ \langle\hat{l}_R, \text{“}value\text{”}\rangle, \langle\hat{l}_R, \text{“}writable\text{”}\rangle, \langle\hat{l}_R, \text{“}enumerable\text{”}\rangle, \langle\hat{l}_R, \text{“}configurable\text{”}\rangle\ \}$$

$$LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})$$

$$LP_4 = \{\ \langle\#\widehat{PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyNames}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

where $\hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})$

$$\hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{l}_R = (\hat{a}_1, \mathsf{Recent})$$

$$LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1)$$

$$LP_2 = \bigcup_{\hat{l}\in\hat{v}.2}\bigcup_{i\in\{0,\ldots,|\widehat{\mathsf{GetProps}}(\hat{H},\hat{l})|-1\}}\{\ \langle\hat{l}, i\rangle\ \}$$

$$LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})$$

$$LP_4 = \{\ \langle\#\widehat{PureLocal}_R, @return\rangle\ \}$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \\
\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1) \\
\qquad \hat{v}_1 = getArgValue(\hat{H},\hat{C}, \text{“0”}) \\
\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C}, \text{“}length\text{”})) \\
\qquad \hat{v}_2 = getArgValue(\hat{H},\hat{C}, \text{“1”}) \\
\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{n}_{len} = \hat{2} \wedge \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_2 = \bigcup_{s \in \underline{\widehat{\mathsf{NewObject}}}_{def}} \{\ \langle \hat{l}_R, s \rangle\ \} \\
\qquad LP_3 = \begin{cases} \bigcup_{\hat{l} \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperties}}}_{def}(\hat{H}, \hat{l}_R, \hat{l}) & \text{if } \hat{n}_{len} = \hat{2} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2) \\
\qquad LP_5 = \{\ \langle \hat{\#PureLocal}_R, @return \rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.defineProperty}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v}_1 = getArgValue(\hat{H},\hat{C}, \text{“0”}) \\
\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad \hat{s}_{name} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H},\hat{C}, \text{“1”}))) \\
\qquad \hat{v}_2 = getArgValue(\hat{H},\hat{C}, \text{“2”}) \\
\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{n}_{len} = \hat{2} \wedge \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_1 = \bigcup_{\hat{l}_1 \in \hat{v}_1.2} \bigcup_{\hat{l}_2 \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperty}}}_{def}(\hat{H}, \hat{l}_1, \hat{s}_{name}, \hat{l}_2) LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2) \\
\qquad LP_3 = \{\ \langle \hat{\#PureLocal}_R, @return \rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.definePropeties}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v}_1 = getArgValue(\hat{H},\hat{C}, \text{“0”}) \\
\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad \hat{v}_2 = getArgValue(\hat{H},\hat{C}, \text{“1”}) \\
\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{n}_{len} = \hat{2} \wedge \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_1 = \bigcup_{\hat{l}_1 \in \hat{v}_1.2} \bigcup_{\hat{l}_2 \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperties}}}_{def}(\hat{H}, \hat{l}_1, \hat{l}_2) LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2) \\
\qquad LP_3 = \{\ \langle \hat{\#PureLocal}_R, @return \rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.seal}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.freeze}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”}) \\
\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_1 = \bigcup_{\hat{l}_1 \in \hat{v}.2} \{\ \langle \hat{l}, @extensible \rangle\ \} \cup \bigcup_{s \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H},\hat{l})} \{\ \langle \hat{l}, s \rangle\ \} \\
\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}) \\
\qquad LP_3 = \{\ \langle \hat{\#PureLocal}_R, @return \rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.preventExtensions}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”}) \\
\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_1 = \bigcup_{\hat{l}_1 \in \hat{v}.2} \{\ \langle \hat{l}, @extensible \rangle\ \} \\
\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}) \\
\qquad LP_3 = \{\ \langle \hat{\#PureLocal}_R, @return \rangle\ \}
\end{array}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isSealed}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isFrozen}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isExtensible}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.keys}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1)\\
&\qquad\qquad \hat{v} = getArgValue(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{v}.2}\bigcup_{s\in\underline{\widehat{\mathsf{NewArrayObject}}}_{def}}\{\ \langle \hat{l}_R, s\rangle\ \}\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_4 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad (LP_2, \hat{es}) = \begin{cases} (\underline{\widehat{\mathsf{toObject}}}_{use}(\hat{H},\hat{C},\hat{v}_{new},\hat{a}_1), \hat{es}'_1) & \text{if } \hat{v}.1.3 \not\sqsubseteq \bot_{Bool} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{Number} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{String}\\ (\{\},\{\}) & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{v}_{new} = Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}.1.3, \hat{v}.1.4, \hat{v}.1.5), \hat{v}.2)\\
&\qquad\qquad \hat{es}' = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_3 = \begin{cases} \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{v}_{new},\hat{a}_1) & \\ & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_5 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad \hat{v} = getArgValue(\hat{H},\hat{C},\text{“0”})\\
&\qquad\qquad (LP_2, \hat{es}) = \begin{cases} (\bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\hat{P}_1\cup\hat{P}_2\cup\hat{P}_3}\{\ \langle \hat{l}, s\rangle\ \}, \hat{es}') & \text{if } \hat{v}.1.3 \not\sqsubseteq \bot_{Bool} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{Number} \vee \hat{v}.1.3 \not\sqsubseteq \bot_{String}\\ (\{\},\{\}) & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{v}_{new} = Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}.1.3, \hat{v}.1.4, \hat{v}.1.5), \hat{v}.2)\\
&\qquad\qquad \hat{P}_1 = \begin{cases} \underline{\widehat{\mathsf{NewBool}}}_{def} & \text{if } \hat{v}_{new}.1.3 \not\sqsubseteq \bot_{Bool}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{P}_2 = \begin{cases} \underline{\widehat{\mathsf{NewNumber}}}_{def} & \text{if } \hat{v}_{new}.1.4 \not\sqsubseteq \bot_{Number}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{P}_3 = \begin{cases} \underline{\widehat{\mathsf{NewString}}}_{def}(\hat{v}_{new}.1.5) & \text{if } \hat{v}_{new}.1.5 \not\sqsubseteq \bot_{String}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{es}' = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_3 = \begin{cases} \bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\underline{\widehat{\mathsf{NewObject}}}_{def}}\{\ \langle \hat{l}, s\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \vee \hat{v}.1.2 \not\sqsubseteq \bot_{Null}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_5 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getPrototypeOf}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad LP_2 = \bigcup_{\hat{l} \in \hat{C}.2} \{ \ \langle \hat{l}, @proto \rangle \ \} \\
&\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}) \\
&\qquad LP_4 = \{ \ \langle \# \hat{PureLocal}_R, @return \rangle \ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyDescriptor}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H}, \hat{C}, \text{“1”}))) \\
&\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“1”}) \\
&\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
&\qquad LP_2 = \bigcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\mathsf{absPair}}}(\hat{H}, \hat{l}, \hat{s}) \\
&\qquad LP_3 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_1) \\
&\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}) \\
&\qquad LP_5 = \{ \ \langle \# \hat{PureLocal}_R, @return \rangle \ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.getOwnPropertyNames}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
&\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_1) \\
&\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad LP_3 = \bigcup_{\hat{l} \in \hat{v}.2} \bigcup_{s \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}, \hat{l})} \{ \ \langle \hat{l}, s \rangle, \langle \hat{l}, \text{“}\textit{@default\_number}\text{”} \rangle, \langle \hat{l}, \text{“}\textit{@default\_other}\text{”} \rangle \ \} \\
&\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}) \\
&\qquad LP_5 = \{ \ \langle \# \hat{PureLocal}_R, @return \rangle \ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.create}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
&\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_1) \\
&\qquad \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad \hat{v}_2 = getArgValue(\hat{H}, \hat{C}, \text{“1”}) \\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“1”})) \\
&\qquad LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“1”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“}length\text{”}) \\
&\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”})) \\
&\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{n}_{len} = \hat{2} \land \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad LP_3 = \begin{cases} \bigcup_{\hat{l} \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperties}}}_{use}(\hat{H}, \hat{l}_R, \hat{l}) & \text{if } \hat{n}_{len} = \hat{2} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2) \\
&\qquad LP_5 = \{ \ \langle \# \hat{PureLocal}_R, @return \rangle \ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.defineProperty}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
&\quad \text{where } \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
&\qquad \hat{s}_{name} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H}, \hat{C}, \text{“1”}))) \\
&\qquad \hat{v}_2 = getArgValue(\hat{H}, \hat{C}, \text{“2”}) \\
&\qquad LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“1”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“2”}) \\
&\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \hat{n}_{len} = \hat{2} \land \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases} \\
&\qquad LP_1 = \bigcup_{\hat{l}_1 \in \hat{v}_1.2} \bigcup_{\hat{l}_2 \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperty}}}_{use}(\hat{H}, \hat{l}_1, \hat{s}_{name}, \hat{l}_2) LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2) \\
&\qquad LP_3 = \{ \ \langle \# \hat{PureLocal}_R, @return \rangle \ \}
\end{aligned}$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.definePropeties}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \ \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{v}_2 = getArgValue(\hat{H}, \hat{C}, \text{“1”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \cup getArgValue_{use}(\hat{H}, \hat{C}, \text{“1”})\\
&\qquad\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad \hat{es}_2 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{n}_{len} = \hat{2} \wedge \hat{v}_2.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_1 = \textstyle\bigcup_{\hat{l}_1 \in \hat{v}_1.2} \bigcup_{\hat{l}_2 \in \hat{v}_2.2} \underline{\widehat{\mathsf{DefineProperties}}}_{use}(\hat{H}, \hat{l}_1, \hat{l}_2) LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.seal}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.freeze}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \ \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \textstyle\bigcup_{\hat{l}_1 \in \hat{v}.2} \bigcup_{s \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}, \hat{l})} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.preventExtensions}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \ \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isSealed}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isFrozen}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \ \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \textstyle\bigcup_{\hat{l}_1 \in \hat{v}.2} \{\ \langle \hat{l}, @extensible\rangle\ \} \cup \bigcup_{s \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}, \hat{l})} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.isExtensible}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \ \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \textstyle\bigcup_{\hat{l}_1 \in \hat{v}.2} \{\ \langle \hat{l}, @extensible\rangle\ \}\\
&\qquad\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Object.keys}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } \ \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_1)\\
&\qquad\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \ \hat{v}_1.1 \not\sqsubseteq \bot_{PValue} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_3 = \textstyle\bigcup_{\hat{l} \in \hat{v}.2} \bigcup_{s \in \underline{\widehat{\mathsf{GetProps}}}(\hat{H}, \hat{l})} \{\ \langle \hat{l}, s\rangle, \langle \hat{l}, \text{“}\textit{@default\_number}\text{”}\rangle, \langle \hat{l}, \text{“}\textit{@default\_other}\text{”}\rangle\ \}\\
&\qquad\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad LP_5 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

11.3.3 Object.prototype

$$
\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.toString”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.valueOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.hasOwnProperty”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.isPrototypeOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.propertyIsEnumerable”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.toString”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.toLocaleString”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\quad \text{where } LP_1 = \bigcup_{\hat{l} \in \hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \} \\
\qquad\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.valueOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \\
\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.hasOwnProperty”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad\qquad \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(\hat{v})) \\
\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad\qquad LP_2 = \bigcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\mathsf{HasOwnProperty}}}_{use}(\hat{H}, \hat{l}, \hat{s}) \\
\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.isPrototypeOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad\qquad LP_2 = \bigcup_{\hat{l} \in \hat{C}.2} \{\ \langle \hat{l}, @proto\rangle\ \} \\
\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

$$
\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Object.prototype.propertyIsEnumerable”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } \hat{s} = \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H}, \hat{C}, \text{“0”}))) \\
\qquad\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad\qquad LP_2 = \bigcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\mathsf{absPair}}}(\hat{H}, \hat{l}, \hat{s}) \\
\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}
$$

11.3.4 Function

11.3.5 Function.prototype

$$\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Function.prototype}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1$$

$$\text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Function.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$\begin{aligned}
\text{where } \hat{es} &= \begin{cases} \{\textsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Function}\text{”} \\ \{\} & \text{otherwise} \end{cases} \\
LP_1 &= \underline{\widehat{\textsf{RaiseException}}}_{def}(\hat{es}) \\
LP_2 &= \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Function.prototype.apply}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6$$

$$\begin{aligned}
\text{where } LP_1 &= \underline{\widehat{\textsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1) \cup \underline{\widehat{\textsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_2) \cup \underline{\widehat{\textsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_3) \\
\hat{es}_1 &= \begin{cases} \{\textsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \widehat{\textsf{false}} \sqsubseteq \underline{\widehat{\textsf{IsCallable}}}(\hat{H},\hat{l}) \\ \{\} & \text{otherwise} \end{cases} \\
\hat{v}_{arg} &= getArgValue(\hat{H},\hat{C}, \text{“}1\text{”}) \\
\hat{v}_{arg1} &= Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}_{arg}.1.3, \hat{v}_{arg}.1.4, \hat{v}_{arg}.1.5), \hat{v}_{arg}.2) \\
(\hat{v}_{arg2}, \hat{es}_2) &= \begin{cases} (Value(\bot_{PValue}, \hat{v}_{arg1}.2), \{\textsf{TypeError}\}) & \text{if } \hat{v}_{arg1}.1 \not\sqsubseteq \bot_{PValue} \\ (\hat{v}_{arg1}, \{\}) & \text{otherwise} \end{cases} \\
LP_2 &= \begin{cases} LP_2' & \text{if } \hat{v}_{arg2} \not\sqsubseteq \bot_{Value} \\ \{\} & \text{otherwise} \end{cases} \\
LP_2' &= \textstyle\bigcup_{\hat{l} \in \hat{v}_{arg2}.2} \begin{cases} \bigcup_{i \in \{0,\ldots,n-1\}} \{\ \langle \hat{l}_{R_3}, i\rangle\ \} & \text{if } \hat{n}_{len} = \textsf{UIntSingle}(n) \\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ \underline{\widehat{\textsf{absPair}}}(\hat{H}, \hat{l}_{R_3}, \textsf{NumStr}) & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= \underline{\widehat{\textsf{toUInt32}}}(\underline{\widehat{\textsf{Proto}}}(\hat{H},\hat{l}, \text{“}\widehat{length}\text{”})) \\
\hat{v}_{this} &= getArgValue(\hat{H},\hat{l}, \text{“}0\text{”}) \\
\hat{L}_{arg} &= \underline{\widehat{\textsf{getThis}}}(\hat{H}, \hat{v}_{this}) \\
\hat{v}_{this2} &= Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}_{this}.1.3, \hat{v}_{this}.1.4, \hat{v}_{this}.1.5), \hat{L}_{arg}) \\
LP_3 &= \underline{\widehat{\textsf{toObject}}}_{def}(\hat{H},\hat{C},\hat{v}_{this2},\hat{a}_4) \\
LP_4 &= \{\ \langle \hat{l}_{R_3}, \text{“}callee\text{”}\rangle\ \} \\
LP_5 &= \underline{\widehat{\textsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2) \\
LP_6 &= \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Function.prototype.call}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6$$

$$\begin{aligned}
\text{where } LP_1 &= \underline{\widehat{\textsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1) \cup \underline{\widehat{\textsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_2) \\
\hat{es} &= \begin{cases} \{\textsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \widehat{\textsf{false}} \sqsubseteq \underline{\widehat{\textsf{IsCallable}}}(\hat{H},\hat{l}) \\ \{\} & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= getArgValue(\hat{H},\hat{l}, \text{“}length\text{”}) \hat{-} \hat{1} \\
LP_2 &= \begin{cases} \bigcup_{i \in \{0,\ldots,n-1\}} \{\ \langle \hat{l}_{R_1}, i\rangle\ \} & \text{if } \hat{n}_{len} = \textsf{UIntSingle}(n) \vee \hat{n}_{len} = \textsf{NUIntSingle}(n) \\ \{\} & \text{otherwise} \end{cases} \\
\hat{v}_{this} &= getArgValue(\hat{H},\hat{l}, \text{“}0\text{”}) \\
\hat{L}_{arg} &= \underline{\widehat{\textsf{getThis}}}(\hat{H}, \hat{v}_{this}) \\
\hat{v}_{this2} &= Value(PValue(\bot_{Undef}, \bot_{Null}, \hat{v}_{this}.1.3, \hat{v}_{this}.1.4, \hat{v}_{this}.1.5), \hat{L}_{arg}) \\
LP_3 &= \underline{\widehat{\textsf{toObject}}}_{def}(\hat{H},\hat{C},\hat{v}_{this2},\hat{a}_4) \\
LP_4 &= \{\ \langle \hat{l}_{R_3}, \text{“}callee\text{”}\rangle\ \} \\
LP_5 &= \underline{\widehat{\textsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2) \\
LP_6 &= \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Function}\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
&\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype.apply}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7\\
&\quad \text{where } LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_1) \cup \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_2) \cup \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_3)\\
&\qquad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H},\hat{l})\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad \hat{L}_{fun} = \left\{\ \hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H},\hat{l})\ \right\}\\
&\qquad \hat{v}_{arg} = getArgValue(\hat{H},\hat{C},\text{“}1\text{”})\\
&\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”}) \cup getArgValue_{use}(\hat{H},\hat{C},\text{“}1\text{”})\\
&\qquad \hat{v}_{arg1} = Value(PValue(\bot_{Undef},\bot_{Null},\hat{v}_{arg}.1.3,\hat{v}_{arg}.1.4,\hat{v}_{arg}.1.5),\hat{v}_{arg}.2)\\
&\qquad (\hat{v}_{arg2},\hat{es}_2) = \begin{cases} (Value(\bot_{PValue},\hat{v}_{arg1}.2),\{\mathsf{TypeError}\}) & \text{if } \hat{v}_{arg1}.1 \not\sqsubseteq \bot_{PValue}\\ (\hat{v}_{arg1},\{\}) & \text{otherwise}\end{cases}\\
&\qquad LP_3 = \begin{cases} LP_3' & \text{if } \hat{v}_{arg2} \not\sqsubseteq \bot_{Value}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad LP_3' = \textstyle\bigcup_{\hat{l}\in\hat{v}_{arg2}.2} \hat{LP}_{len} \cup \begin{cases} \bigcup_{i\in\{0,\dots,n-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{i}) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n)\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise}\end{cases}\\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”}))\\
&\qquad \hat{LP}_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}\widehat{length}\text{”})\\
&\qquad \hat{v}_{this} = getArgValue(\hat{H},\hat{l},\text{“}0\text{”})\\
&\qquad \hat{L}_{arg} = \underline{\widehat{\mathsf{getThis}}}(\hat{H},\hat{v}_{this})\\
&\qquad \hat{v}_{this2} = Value(PValue(\bot_{Undef},\bot_{Null},\hat{v}_{this}.1.3,\hat{v}_{this}.1.4,\hat{v}_{this}.1.5),\hat{L}_{arg})\\
&\qquad LP_4 = \underline{\widehat{\mathsf{toObject}}}_{use}(\hat{H},\hat{C},\hat{v}_{this2},\hat{a}_4)\\
&\qquad LP_5 = \textstyle\bigcup_{\hat{l}\in\hat{L}_{fun}} \{\ \langle \hat{l}@function\rangle, \langle \hat{l}@scope\rangle\ \}\\
&\qquad LP_6 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_7 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Function.prototype.call}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6 \cup LP_7\\
&\quad \text{where } LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1) \cup \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_2)\\
&\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \widehat{\mathsf{false}} \sqsubseteq \underline{\widehat{\mathsf{IsCallable}}}(\hat{H},\hat{l})\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”}) \cup getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
&\qquad \hat{n}_{len} = getArgValue(\hat{H},\hat{l},\text{“}length\text{”})\,\hat{-}\,\hat{1}\\
&\qquad LP_3 = \begin{cases} \bigcup_{i\in\{0,\dots,n-1\}} getArgValue_{use}(\hat{H},\hat{C},i+1) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \vee \hat{n}_{len} = \mathsf{NUIntSingle}(n)\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad \hat{v}_{this} = getArgValue(\hat{H},\hat{l},\text{“}0\text{”})\\
&\qquad \hat{L}_{arg} = \underline{\widehat{\mathsf{getThis}}}(\hat{H},\hat{v}_{this})\\
&\qquad \hat{v}_{this2} = Value(PValue(\bot_{Undef},\bot_{Null},\hat{v}_{this}.1.3,\hat{v}_{this}.1.4,\hat{v}_{this}.1.5),\hat{L}_{arg})\\
&\qquad LP_4 = \underline{\widehat{\mathsf{toObject}}}_{use}(\hat{H},\hat{C},\hat{v}_{this2},\hat{a}_4)\\
&\qquad LP_5 = \textstyle\bigcup_{\hat{l}\in\hat{L}_{fun}} \{\ \langle \hat{l}@function\rangle, \langle \hat{l}@scope\rangle\ \}\\
&\qquad LP_6 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_7 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

11.3.6 Array

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}_1)\\
&\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”}))\\
&\qquad \hat{es} = \begin{cases} \hat{es}' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 1\\ \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n! = 1\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \hat{es}' & \text{otherwise} \end{cases}\\
&\qquad \hat{es}' = \begin{cases} \{\} & \text{if } \hat{v}.1.4 = \mathsf{UInt} \vee \hat{v}.1.4 = \mathsf{UIntSingle} \vee \hat{v}.1.4 = \mathsf{NumBot}\\ \{\mathsf{RangeError}\} & \text{otherwise} \end{cases}\\
&\qquad LP_2 = \bigcup_{s \in \underline{\widehat{\mathsf{NewArrayObejct}}}_{def}} \{\ \langle \hat{l}_R, s \rangle\ \}\\
&\qquad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_4 = \{\ \langle \#\hat{PureLocal}_R, @return \rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.constructor}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”}))\\
&\qquad \hat{es} = \begin{cases} \hat{es}' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 1\\ \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n! = 1\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \hat{es}' & \text{otherwise} \end{cases}\\
&\qquad \hat{es}' = \begin{cases} \{\} & \text{if } \hat{v}.1.4 = \mathsf{UInt} \vee \hat{v}.1.4 = \mathsf{UIntSingle} \vee \hat{v}.1.4 = \mathsf{NumBot}\\ \{\mathsf{RangeError}\} & \text{otherwise} \end{cases}\\
&\qquad LP_1 = \bigcup_{\hat{l} \in \hat{C}.2} \bigcup_{s \in \underline{\widehat{\mathsf{NewArrayObejct}}}_{def}} \{\ \langle \hat{l}, s \rangle\ \}\\
&\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_3 = \{\ \langle \#\hat{PureLocal}_R, @return \rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.isArray}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\hat{PureLocal}_R, @return \rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H}, \hat{C}, \hat{a}_1)\\
&\qquad \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”}))\\
&\qquad LP_2 = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \cup getArgValue(\hat{H}, \hat{C}, \text{“}legnth\text{”})\\
&\qquad \hat{es} = \begin{cases} \hat{es}' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 1\\ \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n! = 1\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \hat{es}' & \text{otherwise} \end{cases}\\
&\qquad \hat{es}' = \begin{cases} \{\} & \text{if } \hat{v}.1.4 = \mathsf{UInt} \vee \hat{v}.1.4 = \mathsf{UIntSingle} \vee \hat{v}.1.4 = \mathsf{NumBot}\\ \{\mathsf{RangeError}\} & \text{otherwise} \end{cases}\\
&\qquad LP_3 = \begin{cases} \bigcup_{i \in \{0,\ldots,n-1\}} \{\ \langle \hat{l}_R, i \rangle\ \} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n)\\ \{\} & \text{if } \hat{n}_{arglen} = \bot_{Number}\\ \\ ahfabsPair(\hat{H}, \hat{l}_R, \mathsf{NumStr} & \text{otherwise} \end{cases}\\
&\qquad LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_5 = \{\ \langle \#\hat{PureLocal}_R, @return \rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_3 \cup LP_4 \cup LP_5\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})\\
&\qquad \hat{n}_{arg} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C}, \text{“}length\text{”}))\\
&\qquad LP_1 = getArgValue(\hat{H},\hat{C}, \text{“0”}) \cup getArgValue(\hat{H},\hat{C}, \text{“}legnth\text{”})\\
&\qquad \hat{es} = \begin{cases} \hat{es}' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 1\\ \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n! = 1\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \hat{es}' & \text{otherwise} \end{cases}\\
&\qquad \hat{es}' = \begin{cases} \{\} & \text{if } \hat{v}.1.4 = \mathsf{UInt} \vee \hat{v}.1.4 = \mathsf{UIntSingle} \vee \hat{v}.1.4 = \mathsf{NumBot}\\ \{\mathsf{RangeError}\} & \text{otherwise} \end{cases}\\
&\qquad LP_2 = \begin{cases} \bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{i\in\{0,\ldots,n-1\}} \{\ \langle \hat{l}, i\rangle\ \} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n)\\ \{\} & \text{if } \hat{n}_{arglen} = \bot_{Number}\\ \bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{absPair}}}(\hat{H}, \hat{l}, \mathsf{NumStr} & \text{otherwise} \end{cases}\\
&\qquad LP_3 = \bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{s\in\underline{\widehat{\mathsf{NewArrayObejct}}}_{def}} LP_4 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad LP_5 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.isArray}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H},\hat{C}, \text{“0”})\\
&\qquad LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“0”})\\
&\qquad LP_2 = \bigcup_{\hat{l}\in\hat{v}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
&\qquad LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

11.3.7 Array.prototype

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toLocaleString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.join}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.concat}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent})\\
&\qquad LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H},\hat{C},\hat{a}_1)\\
&\qquad \hat{n}_{arglen} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C}, \text{“}length\text{”}))\\
&\qquad LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{arglen} = \bot_{Number}\\ LP_2' & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n_arglen)\\ LP_{array} & \text{otherwise} \end{cases}\\
&\qquad LP_2' = \bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ LP_{array} \cup \bigcup_{i\in\{0,\ldots,n-1\}} \{\ \langle \hat{l}_R, i\rangle\ \} & \\ & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n)\\ LP_{array} & \text{otherwise} \end{cases}\\
&\qquad \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}))\\
&\qquad LP_{array} = \bigcup_{s\in\underline{\widehat{\mathsf{NewArrayObejct}}}_{def}} \{\ \langle \hat{l}_R, s\rangle\ \}\\
&\qquad LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{``}\mathsf{Array.prototype.pop}\text{''}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5$$

$$\begin{aligned}
\text{where }\; LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} LP_{length} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} = 0 \\ LP_{length} \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},n_{len}-1) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} > 0 \\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{length} \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''})) \\
LP_{length} &= \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''}) \\
LP_2 &= \{\; \langle \#Pur\hat{e}Local_R, @return \rangle \;\}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{``}\mathsf{Array.prototype.push}\text{''}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$\begin{aligned}
\text{where }\; \hat{n}_{arg} &= \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{``}length\text{''})) \\
LP_1 &= \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number} \\ LP_1' & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ \bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases} \\
LP_1' &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''}) \cup \bigcup_{i\in\{0,\ldots n_{arg}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},i \mathbin{\hat{+}} n) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''})) \\
LP_2 &= \{\; \langle \#Pur\hat{e}Local_R, @return \rangle \;\}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{``}\mathsf{Array.prototype.reverse}\text{''}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$\begin{aligned}
\text{where }\; LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_len = \bot_{Number} \\ LP_1' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''})) \\
LP_1' &= \textstyle\bigcup_{i\in\{0,\ldots,floor(n/2)\}} LP_{swap} \cup LP_{up} \cup LP_{low} \\
LP_{swap} &= \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{low}) \cup \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{up}) & \text{if } \mathsf{tr\hat{u}e} \sqsubseteq \hat{b}_{low} \wedge \mathsf{tr\hat{u}e} \sqsubseteq \hat{b}_{up} \\ \{\} & \text{otherwise} \end{cases} \\
LP_{up} &= \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{low}) \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\hat{s}_{up}) & \text{if } \mathsf{fal\hat{s}e} \sqsubseteq \hat{b}_{low} \wedge \mathsf{tr\hat{u}e} \sqsubseteq \hat{b}_{up} \\ \{\} & \text{otherwise} \end{cases} \\
LP_{low} &= \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\hat{s}_{low}) \cup \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{up}) & \text{if } \mathsf{tr\hat{u}e} \sqsubseteq \hat{b}_{low} \wedge \mathsf{fal\hat{s}e} \sqsubseteq \hat{b}_{up} \\ \{\} & \text{otherwise} \end{cases} \\
\hat{s}_{low} &= \hat{i} \\
\hat{s}_{up} &= n - \hat{i} - 1 \\
\hat{b}_{low}\; & \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{C},\hat{s}_{low}) \\
\hat{b}_{up}\; & \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{C},\hat{s}_{up}) \\
LP_2 &= \{\; \langle \#Pur\hat{e}Local_R, @return \rangle \;\}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{``}\mathsf{Array.prototype.shift}\text{''}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$\begin{aligned}
\text{where }\; LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_len = \bot_{Number} \\ LP_1' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ LP_{store_{len}} \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases} \\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''})) \\
LP_1' &= \begin{cases} LP_{store_{len}} & \text{if } n = 0 \\ LP_{array} \cup LP_{delete} \cup LP_{store_{len}} & \text{otherwise} \end{cases} \\
LP_{array} &= \textstyle\bigcup_{i\in\{1,\ldots,n-1\}} LP_{shift1} \cup LP_{shift2} \\
LP_{shift1} &= \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{to}) & \text{if } \mathsf{tr\hat{u}e} \sqsubseteq \hat{b} \\ \{\} & \text{otherwise} \end{cases} \\
LP_{shift2} &= \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\hat{s}_{to}) & \text{if } \mathsf{fal\hat{s}e} \sqsubseteq \hat{b} \\ \{\} & \text{otherwise} \end{cases} \\
LP_{delete} &= \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},n \mathbin{\hat{-}} 1) \\
LP_{store_{len}} &= \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\text{``}len\hat{g}th\text{''}) \\
\hat{s}_{from} &= \hat{i} \\
\hat{s}_{to} &= n - \hat{i} - 1 \\
\hat{b} &= \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{C},\hat{s}_{from}) \\
LP_2 &= \{\; \langle \#Pur\hat{e}Local_R, @return \rangle \;\}
\end{aligned}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.slice}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$
\begin{array}{ll}
\text{where} & \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
& LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}_1) \\
& \hat{n}_{start} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“0”})) \\
& \hat{n}_{end} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“1”})) \\
& LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{start} = \bot_{Number} \vee \hat{n}_{end} = \bot_{Number} \\ LP_{single} & \text{if } \gamma(\hat{n}_{start}) = n_{start} \wedge \gamma(\hat{n}_{end}) = n_{end} \\ LP_{top} & \text{otherwise} \end{cases} \\
& LP_{single} = \bigcup_{\hat{l} \in \hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{array} \cup \bigcup_{i \in \{0, \ldots, n_{span}-1\}} \{ \ \langle \hat{l}_R, i \rangle \ \} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ LP_{array} & \text{otherwise} \end{cases} \\
& LP_{top} = \bigcup_{\hat{l} \in \hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{array} & \text{otherwise} \end{cases} \\
& \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}\widehat{length}\text{”})) \\
& LP_{array} = \bigcup_{s \in \underline{\widehat{\mathsf{NewArrayObject}}}_{def}} \{ \ \langle \hat{l}_R, s \rangle \ \} \\
& n_{from} = \begin{cases} max(n + n_{start}, 0) & \text{if } \hat{n}_{len} = \bot_{Number} \\ min(n_{start}, n) & \text{otherwise} \end{cases} \\
& n_{to} = \begin{cases} max(n + n_{end}, 0) & \text{if } \hat{n}_{len} = \bot_{Number} \\ min(n_{end}, n) & \text{otherwise} \end{cases} \\
& n_{span} = max(n_{to} - n_{from}, 0) \\
& LP_3 = \{ \ \langle \widehat{\#PureLocal}_R, @return \rangle \ \}
\end{array}
$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.splice}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$
\begin{array}{ll}
\text{where} & \hat{l}_R = (\hat{a}_1, \mathsf{Recent}) \\
& LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{def}(\hat{H}, \hat{C}, \hat{a}_1) \\
& \hat{n}_{arg} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”})) \\
& \hat{n}_{start} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“0”})) \\
& \hat{n}_{count} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“1”})) \\
& LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{start} = \bot_{Number} \vee \hat{n}_{count} = \bot_{Number} \\ LP_{single} & \text{if } \gamma(\hat{n}_{start}) = n_{start} \wedge \gamma(\hat{n}_{count}) = n_{count} \\ LP_{top} & \text{otherwise} \end{cases} \\
& LP_{single} = \bigcup_{\hat{l} \in \hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{array} \cup LP_{single_1} \cup LP_{single_2} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \\ LP_{array} \cup LP_{top_{store}} \cup LP_{top_{delet}} & \text{otherwise} \end{cases} \\
& LP_{single_1} = \bigcup_{i \in \{0, \ldots, n_{delCount}-1\}} \{ \ \langle \hat{l}_R, i \rangle \ \} \\
& n_{delCount} = min(max(n_{count}, 0), n_{len} - n_{start}) \\
& LP_{single_2} = \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number} \\ LP_{single_3} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ \cup LP_{top_{store}} \cup LP_{top_{delete}} & \text{otherwise} \end{cases} \\
& LP_{single_3} = \begin{cases} LP_{single_{move1}} \cup LP_{single_{add}} \cup LP_{single_{delete}} \cup LP_{single_{length}} & \text{if } n_{addCount} < n_{count} \\ LP_{single_{move2}} \cup LP_{single_{add}} \cup LP_{single_{length}} & \text{otherwise} \end{cases} \\
& n_{addCount} = n_{arg} - 2 \\
& LP_{single_{move1}} = \bigcup_{i \in \{n_{moveStart}, \ldots, n_{len}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \hat{s}_{to_1}) \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \hat{s}_{to_1}) \\
& LP_{single_{move2}} = \bigcup_{i \in \{0, \ldots, n_{len}-n_{moveStart}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \hat{s}_{to_2}) \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \hat{s}_{to_2}) \\
& \hat{s}_{to_1} = i - n_{count} \hat{+} n_{addCount} \\
& \hat{s}_{to_2} = n_{len} - 1 - i - n_{\hat{c}ount} + n_{addCount} \\
& LP_{single_{add}} = \bigcup_{i \in \{0, \ldots, n_{addCount}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \widehat{n_{start} + i}) \\
& LP_{single_{delete}} = \bigcup_{i \in \{n_{newLen}, \ldots, n_{len}-1\}} \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \hat{i}) \\
& n_{newLen} = n_{len} + n_{addCount} - n_{count} LP_{single_{length}} = \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \text{“}\widehat{length}\text{”}) \\
& LP_{top} = \bigcup_{\hat{l} \in \hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{array} \cup LP_{top_{store}} \cup LP_{top_{delet}} & \text{otherwise} \end{cases} \\
& \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}\widehat{length}\text{”})) \\
& LP_{array} = \bigcup_{s \in \underline{\widehat{\mathsf{NewArrayObject}}}_{def}} \{ \ \langle \hat{l}_R, s \rangle \ \} \\
& LP_{top_{store}} = \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H}, \hat{l}, \mathsf{NumStr}) \\
& LP_{top_{delete}} = \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \mathsf{NumStr}) \\
& LP_3 = \{ \ \langle \widehat{\#PureLocal}_R, @return \rangle \ \}
\end{array}
$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.unshift}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$
\begin{aligned}
\text{where } \hat{n}_{arg} &= \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\widehat{\text{“}length\text{”}}))\\
LP_1 &= \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number} \\ LP_{single} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ LP_{top} & \text{otherwise} \end{cases}\\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}))\\
LP_{single} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{unshift} \cup LP_{add} \cup LP_{single_{length}} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \\ LP_{store} \cup LP_{delete} & \text{otherwise} \end{cases}\\
LP_{unshift} &= \textstyle\bigcup_{i\in\{0,\ldots n_{len}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\hat{s}_{to}) \cup \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\hat{s}_{to})\\
\hat{s}_{to} &= n_{len} - 1 \mathbin{\hat{-}} i + n_{arg}\\
LP_{add} &= \textstyle\bigcup_{i\in\{0,\ldots n_{arg}-1\}} \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},i)\\
LP_{delete} &= \underline{\widehat{\mathsf{Delete}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr})\\
LP_{store} &= \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\mathsf{NumStr})\\
LP_{single_{length}} &= \underline{\widehat{\mathsf{PropStore}}}_{def}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}))\\
LP_{top} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{store} \cup LP_{delete} & \text{otherwise} \end{cases}\\
LP_2 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.indexOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.lastIndexOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1$$

$$\text{where } LP_1 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.toLocaleString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$
\begin{aligned}
\text{where } \hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”}))\\
LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}length\text{”}))\\
LP_2 &= \begin{cases} LP_2' \cup LP_2'' & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} > 0 \\ \{\} & \text{otherwise} \end{cases}\\
LP_2' &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}0\text{”}))\\
LP_2'' &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{i\in\{1,\ldots,n_{len}-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}i\text{”}))\\
LP_3 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.concat}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

$$
\begin{aligned}
\text{where } \hat{l}_R &= (\hat{a}_1, \mathsf{Recent})\\
LP_1 &= \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_1)\\
\hat{n}_{arglen} &= \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
LP_2 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
LP_3 &= \begin{cases} \{\} & \text{if } \hat{n}_{arglen} = \bot_{Number} \\ LP_{single} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n_arglen) \\ LP_{top} & \text{otherwise} \end{cases}\\
LP_{single} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{len} \cup LP_{array} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n) \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases}\\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}}))\\
LP_{len} &= \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\widehat{\text{“}length\text{”}})\\
LP_{array} &= \textstyle\bigcup_{i\in\{0,\ldots,n-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{i}) \cup \bigcup_{i\in\{0,\ldots,n-1\}} getArgValue_{use}(\hat{H},\hat{l},i)\\
LP_{top} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) \cup getArgValue_{use}(\hat{H},\hat{C},\mathsf{NumStr})\\
LP_4 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.join}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

$$
\begin{aligned}
\text{where } LP_1 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
\hat{n}_{len} &= \underline{\widehat{\mathsf{toUInt32}}}(\textstyle\bigsqcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}length\text{”}))\\
LP_2 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}length\text{”}))\\
LP_3 &= \begin{cases} LP_{first} \cup LP_{remain} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n_{arglen}) \\ \{\} & \text{otherwise} \end{cases}\\
LP_{first} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}\hat{0}\text{”})\\
LP_{remain} &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{i\in\{1,\ldots,n_{arglen}-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{i})\\
LP_4 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.pop}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{n}_{len} = \widehat{\underline{\mathsf{toUInt32}}}(\textstyle\bigsqcup_{\hat{l} \in \hat{C}.2} \widehat{\underline{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}length\text{”}))\\
&\qquad LP_1 = \textstyle\bigcup_{\hat{l} \in \hat{C}.2} \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}length\text{”}))\\
&\qquad LP_2 = \textstyle\bigcup_{\hat{l} \in \hat{C}.2} \begin{cases} LP_{length} \cup LP_{store} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} = 0\\ LP_{length} \cup LP_{single} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} > 0\\ \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ LP_{length} \cup LP_{top} & \text{otherwise} \end{cases}\\
&\qquad \hat{n}_{len} = \widehat{\underline{\mathsf{toUInt32}}}(\widehat{\underline{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}))\\
&\qquad LP_{single} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \widehat{n_{len} - 1}) \cup \widehat{\underline{\mathsf{Delete}}}_{use}(\hat{H}, \hat{l}, \widehat{n_{len} - 1}) \cup \widehat{\underline{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \widehat{n_{len} - 1}) \cup LP_{store}\\
&\qquad LP_{top} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup \widehat{\underline{\mathsf{Delete}}}_{use}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup \cup \widehat{\underline{\mathsf{Delete}}}_{def}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup LP_{store}\\
&\qquad LP_{store} = \underline{\widehat{\mathsf{PropStore}}_{use}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}) \cup \underline{\widehat{\mathsf{PropStore}}_{def}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”})\\
&\qquad LP_{length} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”})\\
&\qquad LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.push}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{n}_{arg} = \widehat{\underline{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”}))\\
&\qquad LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}length\text{”})\\
&\qquad LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number}\\ LP_{single} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg})\\ LP_{top} & \text{otherwise} \end{cases}\\
&\qquad LP_{single} = \textstyle\bigcup_{\hat{l} \in \hat{C}.2} LP_{length} \cup \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number}\\ LP_{single1} \cup LP_{single2} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n)\\ LP_{top} & \text{otherwise} \end{cases}\\
&\qquad \hat{n}_{len} = \widehat{\underline{\mathsf{toUInt32}}}(\widehat{\underline{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}))\\
&\qquad LP_{single1} = \textstyle\bigcup_{i \in \{0, \ldots, n-1\}} \underline{\widehat{\mathsf{PropStore}}_{use}}(\hat{H}, \hat{l}, \hat{i}) \cup \underline{\widehat{\mathsf{PropStore}}_{def}}(\hat{H}, \hat{l}, \hat{i}) \cup getArgValue_{use}(\hat{H}, \hat{C}, i)\\
&\qquad LP_{single2} = \underline{\widehat{\mathsf{PropStore}}_{use}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}) \cup \underline{\widehat{\mathsf{PropStore}}_{def}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”})\\
&\qquad LP_{length} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”})\\
&\qquad LP_{top} = getArgValue_{use}(\hat{H}, \hat{C}, \mathsf{NumStr}) \cup \textstyle\bigcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\mathsf{PropStore}}_{use}}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup \underline{\widehat{\mathsf{PropStore}}_{def}}(\hat{H}, \hat{l}, \mathsf{NumStr})\\
&\qquad LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.reverse}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = \textstyle\bigcup_{\hat{l} \in \hat{C}.2} LP_{len} LP_1'\\
&\qquad \hat{n}_{len} = \widehat{\underline{\mathsf{toUInt32}}}(\widehat{\underline{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”}))\\
&\qquad LP_{len} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}len\hat{g}th\text{”})\\
&\qquad LP_1' = \begin{cases} \{\} & \text{if } \hat{n}_l en = \bot_{Number}\\ LP_{single} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n)\\ LP_{top} & \text{otherwise} \end{cases}\\
&\qquad LP_{single} = \textstyle\bigcup_{i \in \{0, \ldots, floor(n/2)\}} \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \hat{s}_{low}) \cup \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \hat{s}_{up}) \cup LP_{swap} \cup LP_{up} \cup LP_{low}\\
&\qquad LP_{swap} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}_{use/def}}(\hat{H}, \hat{l}, \hat{s}_{low}) \cup \underline{\widehat{\mathsf{PropStore}}_{use/def}}(\hat{H}, \hat{l}, \hat{s}_{up}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{b}_{low} \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{b}_{up}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_{up} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}_{use/def}}(\hat{H}, \hat{l}, \hat{s}_{low}) \cup \widehat{\underline{\mathsf{Delete}}}_{use/def}(\hat{H}, \hat{l}, \hat{s}_{up}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq \hat{b}_{low} \wedge \widehat{\mathsf{true}} \sqsubseteq \hat{b}_{up}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_{low} = \begin{cases} \widehat{\underline{\mathsf{Delete}}}_{use/def}(\hat{H}, \hat{l}, \hat{s}_{low}) \cup \underline{\widehat{\mathsf{PropStore}}_{use/def}}(\hat{H}, \hat{l}, \hat{s}_{up}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{b}_{low} \wedge \widehat{\mathsf{false}} \sqsubseteq \hat{b}_{up}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_{top} = \widehat{\underline{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup \underline{\widehat{\mathsf{PropStore}}_{use/def}}(\hat{H}, \hat{l}, \mathsf{NumStr}) \cup \widehat{\underline{\mathsf{Delete}}}_{use/def}(\hat{H}, \hat{l}, \mathsf{NumStr})\\
&\qquad \hat{s}_{low} = \hat{\hat{i}}\\
&\qquad \hat{s}_{up} = n - \hat{\hat{i}} - 1\\
&\qquad \hat{b}_{low}\ \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{C}, \hat{s}_{low})\\
&\qquad \hat{b}_{up}\ \underline{\widehat{\mathsf{HasProperty}}}(\hat{H}, \hat{C}, \hat{s}_{up})\\
&\qquad LP_2 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.shift}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2$$

$$\text{where } LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup LP'_1 \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}}))$$

$$LP_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}})$$

$$LP'_1 = \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{single} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ LP_{top} & \text{otherwise} \end{cases}$$

$$LP_{single} = \begin{cases} LP_{store_{len}} & \text{if } n = 0 \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{0}) \cup LP_{array} \cup LP_{delete} \cup LP_{store_{len}} & \text{otherwise} \end{cases}$$

$$LP_{array} = \textstyle\bigcup_{i\in\{1,\ldots,n-1\}} LP_{shift1} \cup LP_{shift2}$$

$$LP_{shift1} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to}) \cup \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{s}_{from}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{b} \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{shift2} = \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to}) & \text{if } \widehat{\mathsf{false}} \sqsubseteq \hat{b} \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{delete} = \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l}, n \mathbin{\hat{-}} 1)$$

$$LP_{store_{len}} = \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}})$$

$$LP_{top} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) \cup \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}}) \cup \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$\hat{s}_{from} = \hat{i}$$

$$\hat{s}_{to} = n - \hat{i} - 1$$

$$\hat{b} = \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{C},\hat{s}_{from})$$

$$LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.slice}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

$$\text{where } \hat{l}_R = (\hat{a}_1, \mathsf{Recent})$$

$$LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_1)$$

$$\hat{n}_{start} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C}, \text{“0”}))$$

$$\hat{n}_{end} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C}, \text{“1”}))$$

$$LP_2 = getArgValue_{use}(\hat{H},\hat{C}, \text{“0”}) \sqcup getArgValue_{use}(\hat{H},\hat{C}, \text{“1”})$$

$$LP_3 = \begin{cases} \{\} & \text{if } \hat{n}_{start} = \bot_{Number} \vee \hat{n}_{end} = \bot_{Number} \\ LP_{single} & \text{if } \gamma(\hat{n}_{start}) = n_{start} \wedge \gamma(\hat{n}_{end}) = n_{end} \\ LP_{top} & \text{otherwise} \end{cases}$$

$$LP_{single} = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{array} \cup \bigcup_{i\in\{0,\ldots,n_{span}-1\}} LP_{slice} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases}$$

$$LP_{slice} = \begin{cases} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l}, \widehat{from} + i) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{i}) \\ \{\} \cup LP_{array} & \text{otherwise} \end{cases}$$

$$LP_{top} = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{length} \cup LP_{array} & \text{otherwise} \end{cases}$$

$$\hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}}))$$

$$LP_{array} = \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) & \text{otherwise} \end{cases}$$

$$LP_{length} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l}, \widehat{\text{“}length\text{”}})$$

$$n_{from} = \begin{cases} max(n + n_{start}, 0) & \text{if } \hat{n}_{len} = \bot_{Number} \\ min(n_{start}, n) & \text{otherwise} \end{cases}$$

$$n_{to} = \begin{cases} max(n + n_{end}, 0) & \text{if } \hat{n}_{len} = \bot_{Number} \\ min(n_{end}, n) & \text{otherwise} \end{cases}$$

$$n_{span} = max(n_{to} - n_{from}, 0)$$

$$LP_4 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.splice}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

$$\text{where } LP_1 = \underline{\widehat{\mathsf{Oldify}}}_{use}(\hat{H},\hat{C},\hat{a}_1)$$

$$\hat{n}_{arg} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))$$

$$\hat{n}_{start} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C},\text{“}0\text{”}))$$

$$\hat{n}_{count} = \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C},\text{“}1\text{”}))$$

$$LP_2 = getArgValue_{use}(\hat{H},\hat{C},\text{“}\hat{0}\text{”}) \cup getArgValue_{use}(\hat{H},\hat{C},\text{“}\hat{1}\text{”}) \cup getArgValue_{use}(\hat{H},\hat{C},\text{“}len\hat{g}th\text{”})$$

$$LP_3 = \begin{cases} \{\} & \text{if } \hat{n}_{start} = \bot_{Number} \vee \hat{n}_{count} = \bot_{Number} \\ LP_{single} & \text{if } \gamma(\hat{n}_{start}) = n_{start} \wedge \gamma(\hat{n}_{count}) = n_{count} \\ LP_{top} & \text{otherwise} \end{cases}$$

$$LP_{single} = \bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{single_1} \cup LP_{single_2} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \\ LP_{top_{proto}} \cup LP_{top_{store}} \cup LP_{top_{delete}} \cup LP_{top_{get}} & \text{otherwise} \end{cases}$$

$$LP_{single_1} = \bigcup_{i\in\{0,\ldots,n_{delCount}-1\}} \begin{cases} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},i+\hat{f}rom) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{i}) \\ \{\} & \text{otherwise} \end{cases}$$

$$n_{delCount} = min(max(n_{count},0), n_{len} - n_{start})$$

$$LP_{single_2} = \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number} \\ LP_{single_3} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ LP_{top_{store}} \cup LP_{top_{delete}} \cup LP_{top_{get}} & \text{otherwise} \end{cases}$$

$$LP_{single_3} = \begin{cases} LP_{single_{move1}} \cup LP_{single_{add}} \cup LP_{single_{delete}} \cup LP_{single_{length}} & \text{if } n_{addCount} < n_{count} \\ LP_{single_{move2}} \cup LP_{single_{add}} \cup LP_{single_{length}} & \text{otherwise} \end{cases}$$

$$n_{addCount} = n_{arg} - 2$$

$$LP_{single_{move1}} = \bigcup_{i\in\{n_{moveStart},\ldots,n_{len}-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{s}_{from_1}) \cup LP_{move1_1} \cup LP_{move1_2}$$

$$LP_{move1_1} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to_1}) & \text{if } \widehat{\mathsf{true}} = \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from_1}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{move1_2} = \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to_1} & \text{if } \widehat{\mathsf{false}} = \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from_1}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{single_{move2}} = \bigcup_{i\in\{0,\ldots,n_{len}-n_{moveStart}-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{s}_{from_2}) \cup LP_{move2_1} \cup LP_{move2_2}$$

$$LP_{move2_1} = \begin{cases} \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to_1}) & \text{if } \widehat{\mathsf{true}} = \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from_1}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{move2_2} = \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to_1} & \text{if } \widehat{\mathsf{false}} = \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from_1}) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{s}_{to_1} = i - n_{count} \mathbin{\hat{+}} n_{addCount}$$

$$\hat{s}_{to_2} = n_{len} - 1 - i - n_{\hat{c}ount} + n_{addCount}$$

$$\hat{s}_{from_1} = \hat{\hat{i}}$$

$$\hat{s}_{from_2} = n_{len} \mathbin{\hat{-}} 1 - i$$

$$LP_{single_{add}} = \bigcup_{i\in\{0,\ldots,n_{addCount}-1\}} getArgValue(\hat{H},\hat{C},i \mathbin{\hat{+}} 2) \cup \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},n_{st\hat{a}rt}+i)$$

$$LP_{single_{delete}} = \bigcup_{i\in\{n_{newLen},\ldots,n_{len}-1\}} \underline{\widehat{\mathsf{Delete}}}_{use}(\hat{H},\hat{l},\hat{i})$$

$$LP_{single_{length}} = \underline{\widehat{\mathsf{PropStore}}}_{use}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”})$$

$$n_{newLen} = n_{len} + n_{addCount} - n_{count}$$

$$LP_{top} = \bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{top_{proto}} \cup LP_{top_{store}} \cup LP_{top_{delete}} \cup LP_{top_{get}} & \text{otherwise} \end{cases}$$

$$\hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”}))$$

$$LP_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”})$$

$$LP_{top_{proto}} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$LP_{top_{store}} = \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$LP_{top_{delete}} = \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$LP_{top_{get}} = getArgValue_{use}(\hat{H},\hat{C},\mathsf{NumStr})$$

$$LP_4 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.unshift}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

where $\hat{n}_{arg} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}len\hat{g}th\text{”}))$

$$LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}len\hat{g}th\text{”})LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{arg} = \bot_{Number} \\ LP_{single} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ LP_{top} & \text{otherwise} \end{cases}$$

$$\hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”}))$$

$$LP_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”})$$

$$LP_{single} = \bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ LP_{unshift} \cup LP_{add} \cup LP_{single_{length}} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) \cup LP_{store} \cup LP_{delete} & \text{otherwise} \end{cases}$$

$$LP_{unshift} = \bigcup_{i\in\{0,\ldots n_{len}-1\}} LP_{unshift1} \cup LP_{unshift2}$$

$$LP_{unshift1} = \begin{cases} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{s}_{from}) \cup \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from}) \\ \{\} & \text{otherwise} \end{cases}$$

$$LP_{unshift2} = \begin{cases} \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\hat{s}_{to}) & \text{if } \widehat{\mathsf{true}} \sqsubseteq \underline{\widehat{\mathsf{HasProperty}}}(\hat{H},\hat{l},\hat{s}_{from}) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{s}_{to} = n_{len} - 1 \hat{-} i + n_{arg}$$

$$LP_{add} = \bigcup_{i\in\{0,\ldots n_{arg}-1\}} getArgValue_{use}(\hat{H},\hat{C},i) \cup \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},i)$$

$$LP_{single_{length}} = \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”}))$$

$$LP_{delete} = \underline{\widehat{\mathsf{Delete}}}_{use/def}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$LP_{store} = \underline{\widehat{\mathsf{PropStore}}}_{use/def}(\hat{H},\hat{l},\mathsf{NumStr})$$

$$LP_{top} = \bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup \begin{cases} \{\} & \text{if } \hat{n}_{len} = \bot_{Number} \\ \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\mathsf{NumStr}) \cup LP_{store} \cup LP_{delete} & \text{otherwise} \end{cases}$$

$$LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.indexOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

where $\hat{n}_{arg} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}len\hat{g}th\text{”}))$

$$LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}len\hat{g}th\text{”})$$

$$LP_2 = \begin{cases} LP_{search} \cup LP_{single} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{v}_{search} = getArgValue(\hat{H},\hat{C},\text{“}\hat{0}\text{”})$$

$$LP_{search} = getArgValue_{use}(\hat{H},\hat{C},\text{“}\hat{0}\text{”})$$

$$LP_{single} = \bigcup_{\hat{l}\in\hat{C}.2} LP_{len} \cup \begin{cases} LP_{start} \cup LP_{find} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} \neq 0 \\ \{\} & \text{otherwise} \end{cases}$$

$$\hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”}))$$

$$LP_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\text{“}len\hat{g}th\text{”})$$

$$\hat{n}_{start} = \begin{cases} \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H},\hat{C},\text{“}1\text{”})) & \text{if } n_{arg} > 1 \\ \hat{0} & \text{otherwise} \end{cases}$$

$$LP_{start} = \begin{cases} getArgValue_{use}(\hat{H},\hat{C},\text{“}1\text{”} & \text{if } n_{arg} > 1 \\ \hat{0} & \text{otherwise} \end{cases}$$

$$LP_{find} = \begin{cases} \bigcup_{i\in\{0,\ldots,n_k-1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H},\hat{l},\hat{i}) & \text{if } (\hat{n}_{start} = \mathsf{UIntSingle}(n_{start}) \vee \hat{n}_{start} = \mathsf{NUIntSingle}(n_{start})) \wedge n_{start} \\ \{\} & \text{otherwise} \end{cases}$$

$$n_k = \begin{cases} n_{len} - abs(n_{start}) & \text{if } n_{start} < 0 \\ n_{start} & \text{otherwise} \end{cases}$$

$$LP_3 = \{\ \langle \#Pur\hat{e}Local_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Array.prototype.lastIndexOf}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$

$$\begin{array}{ll}
\text{where} & \hat{n}_{arg} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}\hat{length}\text{”})) \\
& LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}\hat{length}\text{”}) \\
& LP_2 = \begin{cases} LP_{search} \cup LP_{single} & \text{if } \hat{n}_{arg} = \mathsf{UIntSingle}(n_{arg}) \\ \{\} & \text{otherwise} \end{cases} \\
& \hat{v}_{search} = getArgValue(\hat{H}, \hat{C}, \text{“}\hat{0}\text{”}) \\
& LP_{search} = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}\hat{0}\text{”}) \\
& LP_{single} = \bigcup_{\hat{l} \in \hat{C}.2} LP_{len} \cup \begin{cases} LP_{start} \cup LP_{find} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n_{len}) \wedge n_{len} \neq 0 \\ \{\} & \text{otherwise} \end{cases} \\
& \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(\underline{\widehat{\mathsf{Proto}}}(\hat{H}, \hat{l}, \text{“}\hat{length}\text{”})) \\
& LP_{len} = \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, \text{“}\hat{length}\text{”}) \\
& \hat{n}_{start} = \begin{cases} \underline{\widehat{\mathsf{toInteger}}}(getArgValue(\hat{H}, \hat{C}, \text{“}1\text{”})) & \text{if } n_{arg} > 1 \\ \hat{0} & \text{otherwise} \end{cases} \\
& LP_{start} = \begin{cases} getArgValue_{use}(\hat{H}, \hat{C}, \text{“}1\text{”} & \text{if } n_{arg} > 1 \\ \hat{0} & \text{otherwise} \end{cases} \\
& LP_{find} = \begin{cases} \bigcup_{i \in \{0, \ldots, n_k - 1\}} \underline{\widehat{\mathsf{Proto}}}_{use}(\hat{H}, \hat{l}, k \mathbin{\hat{-}} i) & \text{if } (\hat{n}_{start} = \mathsf{UIntSingle}(n_{start}) \vee \hat{n}_{start} = \mathsf{NUIntSingle}(n_{start})) \wedge n_s \\ \{\} & \text{otherwise} \end{cases} \\
& n_k = \begin{cases} min(n_{start}, n_{len} - 1) & \text{if } n_{start} \geq 0 \\ n_{len} - abs(n_{start}) & \text{otherwise} \end{cases} \\
& LP_3 = \{ \ \langle \#\hat{PureLocal}_R, @return \rangle \ \}
\end{array}$$

11.3.8 String

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
&\qquad\quad \hat{s} = \begin{cases} \text{“}\hat{\epsilon}\text{”} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 0\\ \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H},\hat{C},\text{“0”}))) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n > 0\\ \bot_{String} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \top_{String} & \text{otherwise}\end{cases}\\
&\qquad\quad LP_1 = \bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{s\in\underline{\widehat{\mathsf{NewString}}}_{def}(\hat{s})} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\quad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.fromCharCode}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
&\qquad\quad LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
&\qquad\quad LP_2 = \begin{cases} \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n == 0\\ getArgValue_{use}(\hat{H},\hat{C},\text{“0”}) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n > 0\\ getArgValue_{use}(\hat{H},\hat{C},\text{“0”}) & \text{if } \mathsf{UInt} \sqsubseteq \hat{n}_{len}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
&\qquad\quad LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
&\qquad\quad LP_2 = \begin{cases} getArgValue_{use}(\hat{H},\hat{C},\text{“0”}) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n > 0\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad \hat{s} = \begin{cases} \text{“}\hat{\epsilon}\text{”} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 0\\ \underline{\widehat{\mathsf{toString}}}(\underline{\widehat{\mathsf{toPrimitive}}}(getArgValue(\hat{H},\hat{C},\text{“0”}))) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n > 0\\ \bot_{String} & \text{if } \hat{n}_{len} = \bot_{Number}\\ \top_{String} & \text{otherwise}\end{cases}\\
&\qquad\quad LP_3 = \bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{s\in\underline{\widehat{\mathsf{NewString}}}_{def}(\hat{s})} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\quad LP_4 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{String.fromCharCode}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
&\qquad\quad LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
&\qquad\quad LP_2 = \begin{cases} \bigcup_{i\in\{0,\ldots,n-1\}} getArgValue_{use}(\hat{H},\hat{C},i) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n)\\ getArgValue_{use}(\hat{H},\hat{C},\mathsf{NumStr}) & \text{if } \mathsf{UInt} \sqsubseteq \hat{n}_{len}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\quad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.9 String.prototype

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toString”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.valueOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$$
$$\text{where } \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\hat{class}).1.2.1.5 \neq \text{“}\hat{String}\text{”} \\ \{\} & \text{otherwise} \end{cases}$$
$$LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})$$
$$LP_2 = \{\ \langle \#\hat{PureLocal}_R, @return\rangle\ \}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charAt”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.charCodeAt”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.concat”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.indexOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.lastIndexOf”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.localeCompare”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.slice”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.substring”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toLowerCase”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toLocaleLowerCase”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toUpperCase”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.toLocaleUpperCase”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“String.prototype.trim”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\text{where } LP_1 = \{\ \langle \#\hat{PureLocal}_R, @return\rangle\ \}$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toString}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.valueOf}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \widehat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{String}\text{”} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
&\qquad\quad \hat{L}_{string} = \{\ \hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{String}\text{”}\ \}\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{L}_{string}} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_3 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\widehat{es})\\
&\qquad\quad LP_4 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.charAt}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.charCodeAt}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.concat}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } \hat{n}_{len} = \underline{\widehat{\mathsf{toUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
&\qquad\quad LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_3 = \begin{cases} \{\} & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n = 0\\ \bigcup_{i\in\{0,\ldots,n-1\}} getArgValue_{use}(\hat{H},\hat{C},i) & \text{if } \hat{n}_{len} = \mathsf{UIntSingle}(n) \wedge n > 0\\ getArgValue_{use}(\hat{H},\hat{C},\mathsf{NumStr}) & \text{if } \mathsf{UInt} \sqsubseteq \hat{n}_{len}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_4 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.localeCompare}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.indexOf}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.lastIndexOf}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.slice}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.substring}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”}) \cup getArgValue_{use}(\hat{H},\hat{C},\text{“}1\text{”})\\
&\qquad\quad LP_2 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toLowerCase}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toLocaleLowerCase}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toUpperCase}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.toLocaleUpperCase}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}\llbracket \mathsf{BuiltintCall}(\text{“}\mathsf{String.prototype.trim}\text{”}, args)\rrbracket(\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\quad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

11.3.10 Boolean

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Boolean”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Boolean.constructor”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{s\in\underline{\widehat{\mathsf{NewBoolean}}}_{def}} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Boolean”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}0\text{”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}length\text{”})\\
&\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Boolean.constructor”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}0\text{”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}length\text{”})\\
&\qquad\qquad LP_3 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \bigcup_{s\in\underline{\widehat{\mathsf{NewBoolean}}}_{def}} \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.11 Boolean.prototype

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Boolean.prototype.toString”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Boolean.prototype.valueOf”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists\hat{l}\in\hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Boolean}\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Boolean.prototype.toString”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Boolean.prototype.valueOf”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
&\qquad\qquad \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists\hat{l}\in\hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Boolean}\text{”}\\ \{\} & \text{otherwise}\end{cases}\\
&\qquad\qquad LP_2 = \underline{\widehat{\mathsf{RaiseException}}}_{use}(\hat{es})\\
&\qquad\qquad \hat{L}_{bool} = \{\ \hat{l} \mid \hat{l}\in\hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{Boolean}\text{”}\ \}\\
&\qquad\qquad LP_3 = \textstyle\bigcup_{\hat{l}\in\hat{L}_{bool}} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.12 Number

$$\hat{\mathcal{I}}_{def}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Number}\text{”}, args)\rrbracket(\hat{H}, \hat{C}) = LP_1$$
$$\text{where } LP_1 = \{ \ \langle \#\hat{PureLocal}_R, @return\rangle \ \}$$

$$\hat{\mathcal{I}}_{def}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Number.constructor}\text{”}, args)\rrbracket(\hat{H}, \hat{C}) = LP_1 \cup LP_2$$
$$\text{where } LP_1 = \bigcup_{\hat{l} \in \hat{C}.2} \bigcup_{s \in \underline{\widehat{\textsf{NewNumber}}}_{def}} \{ \ \langle \hat{l}, s\rangle \ \}$$
$$LP_2 = \{ \ \langle \#\hat{PureLocal}_R, @return\rangle \ \}$$

$$\hat{\mathcal{I}}_{use}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Number}\text{”}, args)\rrbracket(\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3$$
$$\text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})$$
$$LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}length\text{”})$$
$$LP_3 = \{ \ \langle \#\hat{PureLocal}_R, @return\rangle \ \}$$

$$\hat{\mathcal{I}}_{use}\llbracket \textsf{BuiltintCall}(\text{“}\textsf{Number.constructor}\text{”}, args)\rrbracket(\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$
$$\text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})$$
$$LP_2 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}length\text{”})$$
$$LP_3 = \bigcup_{\hat{l} \in \hat{C}.2} \bigcup_{s \in \underline{\widehat{\textsf{NewNumber}}}_{def}} \{ \ \langle \hat{l}, s\rangle \ \}$$
$$LP_4 = \{ \ \langle \#\hat{PureLocal}_R, @return\rangle \ \}$$

11.3.13 Number.prorotype

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.toString}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{n}_{arglen} = \underline{\widehat{\mathsf{ToUInt32}}}(getArgValue(\hat{H}, \hat{C}, \text{“}length\text{”}))\\
&\qquad\quad \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\quad \hat{es}_1 = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(\widehat{@class}).1.2.1.5 \neq \text{“}\widehat{Number}\text{”} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad \hat{es}_2 = \begin{cases} \{\} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n) \wedge n = 0 \\ \{\mathsf{RangeError}\} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n) \wedge n > 0 \wedge (\widehat{\mathsf{true}} \sqsubseteq \hat{v}_1 \hat{<} \hat{2} \vee \widehat{\mathsf{true}} \sqsubseteq \hat{v}_1 \hat{>} \widehat{36}) \\ \{\} & \text{if } \hat{n}_{arglen} = \mathsf{UIntSingle}(n) \wedge n > 0 \wedge \widehat{\mathsf{false}} \sqsubseteq \hat{v}_1 \hat{<} \hat{2} \wedge \widehat{\mathsf{false}} \sqsubseteq \hat{v}_1 \hat{>} \widehat{36}) \\ \{\} & \text{if } \hat{n}_{arglen} = \perp_{number} \\ \{\mathsf{RangeError}\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es}_1 \sqcup \hat{es}_2)\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.toLocaleString}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.valueOf}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{es} = \begin{cases} \{\mathsf{TypeError}\} & \text{if } \exists \hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(\widehat{@class}).1.2.1.5 \neq \text{“}\widehat{Number}\text{”} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.toFixed}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\quad \hat{v}_2 = \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{0} \sqcup \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1 \\ \hat{v}_1 & \text{otherwise} \end{cases}\\
&\qquad\quad \hat{es} = \begin{cases} \{\mathsf{RangeError}\} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{<} \hat{0} \vee \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{>} \widehat{20} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.toExponential}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\quad \hat{v}_2 = \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1 \\ \hat{v}_1 & \text{otherwise} \end{cases}\\
&\qquad\quad \hat{es} = \begin{cases} \{\mathsf{RangeError}\} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{<} \hat{0} \vee \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{>} \widehat{20} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Number.prototype.toPrecision}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v}_1 = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad\quad \hat{v}_2 = \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1 \\ \hat{v}_1 & \text{otherwise} \end{cases}\\
&\qquad\quad \hat{es} = \begin{cases} \{\mathsf{RangeError}\} & \text{if } \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{<} \hat{1} \vee \widehat{\mathsf{true}} \sqsubseteq \hat{v}_2 \hat{>} \widehat{21} \\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\quad LP_1 = \underline{\widehat{\mathsf{RaiseException}}}_{def}(\hat{es})\\
&\qquad\quad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}
$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4 \cup LP_5 \cup LP_6$$

$$\begin{aligned}
\text{where } \hat{n}_{arglen} &= \underline{\widehat{\textsf{ToUInt32}}}(getArgValue(\hat{H},\hat{C},\text{“}length\text{”}))\\
LP_1 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
\hat{v}_1 &= getArgValue(\hat{H},\hat{C},\text{“}0\text{”})\\
\hat{es}_1 &= \begin{cases} \{\textsf{TypeError}\} & \text{if } \exists\hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Number}\text{”}\\ \{\} & \text{otherwise} \end{cases}\\
LP_2 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
\hat{L}_{num} &= \{\ \hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{Number}\text{”}\ \}\\
LP_3 &= \textstyle\bigcup_{\hat{l}\in\hat{L}_{Number}} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
\hat{(}es_2, LP_4) &= \begin{cases} (\{\},\{\}) & \text{if } \hat{n}_{arglen} = \textsf{UIntSingle}(n) \wedge n = 0\\ (\{\textsf{RangeError}\}, getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})) & \text{if } \hat{n}_{arglen} = \textsf{UIntSingle}(n) \wedge n > 0 \wedge (\widehat{\textsf{true}} \sqsubseteq \hat{v}_1 \mathrel{\hat{<}} \hat{2} \vee \textsf{tr}\\ (\{\}, getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})) & \text{if } \hat{n}_{arglen} = \textsf{UIntSingle}(n) \wedge n > 0 \wedge \widehat{\textsf{false}} \sqsubseteq \hat{v}_1 \mathrel{\hat{<}} \hat{2} \wedge \textsf{fa}\\ (\{\},\{\}) & \text{if } \hat{n}_{arglen} = \bot_{number}\\ (\{\textsf{RangeError}\}, getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})) & \text{otherwise} \end{cases}\\
LP_5 &= \underline{\widehat{\textsf{RaiseException}}}_{use}(\hat{es}_1 \sqcup \hat{es}_2)\\
LP_6 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toLocaleString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$\begin{aligned}
\text{where } LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
LP_2 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}length\text{”})\\
LP_3 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.valueOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4$$

$$\begin{aligned}
\text{where } \hat{es} &= \begin{cases} \{\textsf{TypeError}\} & \text{if } \exists\hat{l} \in \hat{C}.2 : \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 \neq \text{“}\widehat{Number}\text{”}\\ \{\} & \text{otherwise} \end{cases}\\
\hat{L}_{num} &= \{\ \hat{l} \mid \hat{l} \in \hat{C}.2 \wedge \hat{H}(\hat{l})(@\widehat{class}).1.2.1.5 = \text{“}\widehat{Number}\text{”}\ \}\\
LP_1 &= \textstyle\bigcup_{\hat{l}\in\hat{C}.2} \{\ \langle \hat{l}, @class\rangle\ \}\\
LP_2 &= \textstyle\bigcup_{\hat{l}\in\hat{L}_{Number}} \{\ \langle \hat{l}, @primitive\rangle\ \}\\
LP_3 &= \underline{\widehat{\textsf{RaiseException}}}_{use}(\hat{es})\\
LP_4 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toFixed}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$\begin{aligned}
\text{where } \hat{v}_1 &= getArgValue(\hat{H},\hat{C},\text{“}0\text{”})\\
LP_1 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
\hat{v}_2 &= \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{0} \sqcup \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1\\ \hat{v}_1 & \text{otherwise} \end{cases}\\
\hat{es} &= \begin{cases} \{\textsf{RangeError}\} & \text{if } \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{<}} \hat{0} \vee \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{>}} \widehat{20}\\ \{\} & \text{otherwise} \end{cases}\\
LP_2 &= \underline{\widehat{\textsf{RaiseException}}}_{use}(\hat{es})\\
LP_3 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toExponential}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$\begin{aligned}
\text{where } \hat{v}_1 &= getArgValue(\hat{H},\hat{C},\text{“}0\text{”})\\
LP_1 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
\hat{v}_2 &= \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1\\ \hat{v}_1 & \text{otherwise} \end{cases}\\
\hat{es} &= \begin{cases} \{\textsf{RangeError}\} & \text{if } \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{<}} \hat{0} \vee \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{>}} \widehat{20}\\ \{\} & \text{otherwise} \end{cases}\\
LP_2 &= \underline{\widehat{\textsf{RaiseException}}}_{use}(\hat{es})\\
LP_3 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Number.prototype.toPrecision}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3$$

$$\begin{aligned}
\text{where } \hat{v}_1 &= getArgValue(\hat{H},\hat{C},\text{“}0\text{”})\\
LP_1 &= getArgValue_{use}(\hat{H},\hat{C},\text{“}0\text{”})\\
\hat{v}_2 &= \begin{cases} Value(PValue(\hat{v}_1.1.1, \hat{v}_1.1.2, \hat{v}_1.1.3, \hat{v}_1.1.4, \hat{v}_1.1.5), \hat{v}_1.2) & \text{if } \top_{Undef} \sqsubseteq \hat{v}.1.1\\ \hat{v}_1 & \text{otherwise} \end{cases}\\
\hat{es} &= \begin{cases} \{\textsf{RangeError}\} & \text{if } \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{<}} \hat{1} \vee \widehat{\textsf{true}} \sqsubseteq \hat{v}_2 \mathrel{\hat{>}} \widehat{21}\\ \{\} & \text{otherwise} \end{cases}\\
LP_2 &= \underline{\widehat{\textsf{RaiseException}}}_{use}(\hat{es})\\
LP_3 &= \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.14 Math

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.abs”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.acos”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.asin”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.atan”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.atan2”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.ceil”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.cos”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.exp”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.floor”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.max”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.min”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.pow”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.log”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.random”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.round”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.sin”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.sqrt”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“Math.tan”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.random”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.abs”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.acos”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.asin”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.atan”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.ceil”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.cos”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.exp”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.floor”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.log”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.round”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.sin”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.sqrt”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.tan”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“0”}) \wedge\ LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.atan2”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.pow”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“0”}) \wedge\ LP_2 = getArgValue_{use}(\hat{H},\hat{C}, \text{“1”})\\
\qquad\qquad \wedge\ LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.max”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“Math.min”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
\quad \text{where } \hat{n} = getArgValue(\hat{H},\hat{C}, \text{“}length\text{”})\\
\qquad\qquad LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}length\text{”})\\
\qquad\qquad LP_2 = \begin{cases} \bigsqcup_{i\in\{0,\ldots,n-1\}} getArgValue_{use}(\hat{H},\hat{C}, i) & \text{if } \hat{n} = \mathsf{UIntSingle}(n)\\ \{\} & \text{otherwise}\end{cases}\\
\qquad\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{array}$$

11.3.15 Date

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\widehat{\underline{\mathsf{NewDate}}}_{def}}\ \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.now}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.constructor}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1 \cup LP_2 \cup LP_3 \cup LP_4\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H},\hat{C}, \text{“}length\text{”})\\
&\qquad\qquad LP_2 = getArgValue_{use}(\hat{H},\hat{C}, \text{“0”})\\
&\qquad\qquad LP_3 = \textstyle\bigcup_{\hat{l}\in\hat{C}.2}\bigcup_{s\in\widehat{\underline{\mathsf{NewDate}}}_{def}}\ \{\ \langle \hat{l}, s\rangle\ \}\\
&\qquad\qquad LP_4 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.now}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.16 Date.prototype

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toDateString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toTimeString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleDateString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toLocaleTimeString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toUTCString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.toISOString}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.valueOf}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getTime}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getFullYear}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getMonth}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getDate}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getDay}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getHours}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getMinutes}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getSeconds}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getMilliseconds}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getTimezoneOffset}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCFullYear}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCMonth}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCDate}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCDay}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCHours}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCMinutes}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCSeconds}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{Date.prototype.getUTCMilliseconds}\text{”}, args)]\!](\hat{H},\hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setTime”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setMilliseconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setSeconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setMinutes”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setHours”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setDate”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setMonth”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setFullYear”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCMilliseconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCSeconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCMinutes”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCHours”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCDate”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCMonth”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

$\hat{\mathcal{I}}_{def}[\![\text{BuiltintCall}(\text{“Date.prototype.setUTCFullYear”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2$

where $LP_1 = \bigcup_{\hat{l}\in\hat{C}.2} \{ \ \langle \hat{l}, @primitive \rangle \ \}$

$LP_2 = \{ \ \langle \#\widehat{PureLocal}_R, @return \rangle \ \}$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toDateString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toTimeString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toLocaleString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toLocaleDateString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toLocaleTimeString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toUTCString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.toISOString”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getFullYear”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getMonth”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getDate”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getDay”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getHours”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getMinutes”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getSeconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getMilliseconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getTimezoneOffset”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCFullYear”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCMonth”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCDate”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCDay”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCHours”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCMinutes”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCSeconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

$\hat{\mathcal{I}}_{use}[\![\text{BuiltintCall}(\text{“Date.prototype.getUTCMilliseconds”}, args)]\!](\hat{H}, \hat{C}) = LP_1$

where $LP_1 = \{ \ \langle \#\widehat{PureLocal}_R, @return \rangle \ \}$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.valueOf}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.getTime}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setTime}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMilliseconds}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setSeconds}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMinutes}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setHours}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setDate}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setMonth}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setFullYear}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMilliseconds}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCSeconds}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMinutes}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCHours}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCDate}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCMonth}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Date.prototype.setUTCFullYear}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = \bigcup_{\hat{l} \in \hat{C}.2} \{\ \langle \hat{l}, @primitive \rangle\ \}\\
&\qquad\qquad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return \rangle\ \}
\end{aligned}
$$

11.3.17 RegExp

11.3.18 RegExp.prototype

11.3.19 Error

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Error.constructor}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“}0\text{”})\\
&\qquad\qquad LP_1 = \begin{cases} \{\ \langle \widehat{\#ErrorLoc}_O, \text{“}message\text{”} \rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return \rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Error.constructor}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“}0\text{”})\\
&\qquad\qquad LP_2 = \begin{cases} \{\ \langle \widehat{\#ErrorLoc}_O, \text{“}message\text{”} \rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad\qquad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return \rangle\ \}
\end{aligned}
$$

11.3.20 Error.prototype

$$
\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“}\textsf{Error.prototype.toString}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1\\
&\quad \text{where } LP_1 = \{\ \langle \widehat{\#PureLocal}_R, @return \rangle\ \}
\end{aligned}
$$

$$
\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“}\textsf{Error.prototype.toString}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = \bigsqcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\textsf{Proto}}}_{use}(\hat{H}, \hat{l}, \widehat{\text{“}name\text{”}})\\
&\qquad\qquad LP_2 = \bigsqcup_{\hat{l} \in \hat{C}.2} \underline{\widehat{\textsf{Proto}}}_{use}(\hat{H}, \hat{l}, \widehat{\text{“}message\text{”}})\\
&\qquad\qquad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return \rangle\ \}
\end{aligned}
$$

11.3.21 EvalError

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“EvalError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_1 = \begin{cases} \{\ \langle \widehat{\#EvalErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“EvalError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_2 = \begin{cases} \{\ \langle \widehat{\#EvalErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

11.3.22 RangeError

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“RangeError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_1 = \begin{cases} \{\ \langle \widehat{\#RangeErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“RangeError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_2 = \begin{cases} \{\ \langle \widehat{\#RangeErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

11.3.23 ReferenceError

$$\begin{array}{l}
\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“ReferenceError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \\
\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_1 = \begin{cases} \{\ \langle \widehat{\#RefErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_2 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

$$\begin{array}{l}
\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“ReferenceError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3 \\
\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”}) \\
\qquad LP_2 = \begin{cases} \{\ \langle \widehat{\#RefErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef} \\ \{\} & \text{otherwise} \end{cases} \\
\qquad LP_3 = \{\ \langle \widehat{\#PureLocal}_R, @return\rangle\ \}
\end{array}$$

11.3.24 SyntaxError

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“SyntaxError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_1 = \begin{cases} \{\ \langle \#\widehat{SyntaxErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“SyntaxError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_2 = \begin{cases} \{\ \langle \#\widehat{SyntaxErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.25 TypeError

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“TypeError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_1 = \begin{cases} \{\ \langle \#\widehat{TypeErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“TypeError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_2 = \begin{cases} \{\ \langle \#\widehat{TypeErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.26 URIError

$$\begin{aligned}
&\hat{\mathcal{I}}_{def}[\![\textsf{BuiltintCall}(\text{“URIError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2 \cup LP_3\\
&\quad \text{where } \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_1 = \begin{cases} \{\ \langle \#\widehat{URIErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_2 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

$$\begin{aligned}
&\hat{\mathcal{I}}_{use}[\![\textsf{BuiltintCall}(\text{“URIError.constructor”}, args)]\!](\hat{H}, \hat{C}) = LP_1 \cup LP_2\\
&\quad \text{where } LP_1 = getArgValue_{use}(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad \hat{v} = getArgValue(\hat{H}, \hat{C}, \text{“0”})\\
&\qquad LP_2 = \begin{cases} \{\ \langle \#\widehat{URIErrorLoc}_O, \text{“}message\text{”}\rangle\ \} & \text{if } \hat{v}.1.1 \not\sqsubseteq \bot_{Undef}\\ \{\} & \text{otherwise} \end{cases}\\
&\qquad LP_3 = \{\ \langle \#\widehat{PureLocal}_R, @return\rangle\ \}
\end{aligned}$$

11.3.27 JSON

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{JSON.parse}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\quad \text{where } \; LP_1 = \{ \;\; \langle \#\widehat{PureLocal}_R, @return \rangle \;\; \}$$

$$\hat{\mathcal{I}}_{def}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{JSON.stringify}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\quad \text{where } \; LP_1 = \{ \;\; \langle \#\widehat{PureLocal}_R, @return \rangle \;\; \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{JSON.parse}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\quad \text{where } \; LP_1 = \{ \;\; \langle \#\widehat{PureLocal}_R, @return \rangle \;\; \}$$

$$\hat{\mathcal{I}}_{use}[\![\mathsf{BuiltintCall}(\text{“}\mathsf{JSON.stringify}\text{”}, args)]\!](\hat{H}, \hat{C}) = LP_1$$
$$\quad \text{where } \; LP_1 = \{ \;\; \langle \#\widehat{PureLocal}_R, @return \rangle \;\; \}$$