dcl	name of derived class
$Method$, $meth$	method name
$Field$, f	field name
Var, var	term variable
$Pointer$, oid	object identifier
j, k, l	index

P	::=			program (cld list)
	|	$\overline{cld}$	M	def.
$\overline{cld}$	::=			class def.'s (cld list)
	|	$[\,]$	M	empty
	|	$cld_1 \,..\, cld_k$	M	def.
cld	::=			class definition
	|	**class** dcl **extends** $cl\{\overline{fd}\ \overline{meth\_def}\}$		def.
C, cl	::=			class name
	|	`Object`		top class
	|	fqn		fully qualified name
fqn	::=			fully-qualified name
	|	dcl		def.
$\overline{fd}$	::=			field declarations (fd list)
	|	$[\,]$	M	empty
	|	$fd_1 \,..\, fd_k$	M	def.
fd	::=			field declaration
	|	$cl\, f;$		def.
$\overline{meth\_def}$	::=			method def.'s ($meth\_def$ list)
	|	$[\,]$	M	empty
	|	$meth\_def_1 \,..\, meth\_def_k$	M	def.
$meth\_def$	::=			method definition
	|	$meth\_sig\{meth\_body\}$		def.
$meth\_sig$	::=			method signature
	|	$cl\ meth(\overline{vd})$		def.
$\overline{vd}$	::=			variable declarations (vd list)
	|	$vd_1 \,..\, vd_k$	M	def.
vd	::=			variable declaration
	|	$cl\ var$		def.
$meth\_body$	::=			method body
	|	$s_1 \,..\, s_k$ **return** y;		def.

s	::=			statement
	|	$\{\,\overline{s_k}^{\,k}\,\}$		block
	|	$var = x;$		variable assignment
	|	$var = x.f;$		field read
	|	$x.f = y;$		field write
	|	$\textbf{if}\ (x == y)\, s\ \textbf{else}\ s'$		conditional branch
	|	$var = \textbf{new}_{ctx}\, cl();$		object creation
	|	$var = x.meth(\overline{y});$		method call
$TVar,\ x,\ y$	::=			term variable
	|	var		normal variable
	|	**this**		ref. to current object
$\overline{x},\ \overline{y}$	::=			term variables (x list)
	|	$x_1 .. x_k$	M	def
ctx	::=			context
	|			no context
$\overline{f}$	::=			fields (f list)
	|	$[\,]$	M	empty
	|	$f_1 .. f_k$	M	def.
	|	$\overline{f}; \overline{f}'$	M	append
$\overline{f}_{opt}$	::=			fields option ($\overline{f}$ option)
	|	**null**	M	none
	|	$\overline{f}$	M	some
$meth\_def_{opt}$	::=			method def. option ($meth\_def$ option)
	|	**null**	M	none
	|	$meth\_def$	M	some
$ctxmeth\_def_{opt}$	::=			method def. in context option (($ctx \times meth\_def$) option)
	|	**null**	M	none
	|	$(ctx, meth\_def)$	M	some
$\overline{meth}$	::=			method names ($meth$ list)
	|	$[\,]$	M	empty
	|	$meth_1 .. meth_k$	M	def.
	|	$\overline{meth}; \overline{meth}'$	M	append
$ctxcld$	::=			class def. in context ($ctx \times cld$)
	|	(ctx, cld)	M	def.
$\overline{ctxcld}$	::=			class def.'s in context ($ctxcld$ list)
	|	$[\,]$	M	empty
	|	$ctxcld_1 .. ctxcld_k$	M	def.
	|	$\overline{ctxcld}@[ctxcld]$	M	rev cons
$ctxcld_{opt}$	::=			class def. lookup result ($ctxcld$ option)
	|	**null**	M	none

	| $ctxcld$	M	some
$\overline{ctxcld}_{opt}$	::=		class def.'s lookup result ($\overline{ctxcld}$ option)
	| **null**	M	none
	| $\overline{ctxcld}$	M	some
Type, τ	::=		type
	| `Object`		supertype of all types
	| $ctx.dcl$		class identifier
τ_{opt}	::=		result of type lookup (τ option)
	| **null**	M	none
	| τ	M	some
	| $\Gamma(x)$	M	static type lookup
	| $H(oid)$	M	dynamic type lookup
$\tau_{opt}^{\perp}$	::=		result of type lookup that can abort
	| τ_{opt}		result of type lookup
	| $\perp$		failed to find a type
$\overline{\tau}$	::=		types (τ list)
	| $\tau_1 .. \tau_k$	M	def.
π	::=		method type
	| $\overline{\tau} \rightarrow \tau$		def.
Γ	::=		type environment ($x \rightharpoonup \tau$)
	| $[x_1 \mapsto \tau_1 .. x_k \mapsto \tau_k]$	M	type mappings
	| $\Gamma[x \mapsto \tau]$	M	Γ with $x \mapsto \tau$
θ	::=		variable mapping ($x \rightharpoonup x$)
	| $[x_1 \mapsto y_1 .. x_k \mapsto y_k]$	M	variable mappings
	| $\theta[x \mapsto y]$	M	θ with $x \mapsto y$
Val, v, w	::=		value
	| **null**		null value
	| oid		object identifier
v_{opt}	::=		result of value lookup (v option)
	| v	M	some
	| $L(x)$	M	lookup value of local variable
	| $H(oid, f)$	M	lookup value of field
L	::=		variable state ($x \rightharpoonup v$)
	| $[\,]$	M	empty variable state
	| $L[x \mapsto v]$	M	L with $x \mapsto v$
	| $L[x_1 \mapsto v_1 .. x_k \mapsto v_k]$	M	L with many mappings
H	::=		heap ($oid \rightharpoonup (\tau \times (f \rightharpoonup v))$)
	| $[\,]$	M	empty heap
	| $H[oid \mapsto (\tau, f_1 \mapsto v_1 .. f_k \mapsto v_k)]$	M	H with new oid of type τ

	$H[[oid, f) \mapsto v]$	M	H with $(oid,f)\mapsto v$

config	::=			configuration
	|	$(P, L, H, \overline{s_k}^{\,k})$		normal configuration
	|	$(P, L, H, Exception)$		exception occurred
Exception	::=			exception
	|	**NPE**		null-pointer exception
nn	::=			natural number (nat)
	|	0	M	zero
	|	1	M	one
	|	$nn + nn'$	M	plus

$\boxed{class\_name(cld) = dcl}$ – extract the class name from a class

CLASS_NAME

$$\overline{class\_name(\textbf{class}\ dcl\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}) = dcl}$$

$\boxed{superclass\_name(cld) = cl}$ – extract the name of the superclass from a class

SUPERCLASS_NAME

$$\overline{superclass\_name(\textbf{class}\ dcl\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}) = cl}$$

$\boxed{class\_fields(cld) = \overline{fd}}$ – extract field declarations from a class

CLASS_FIELDS

$$\overline{class\_fields(\textbf{class}\ dcl\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}) = \overline{fd}}$$

$\boxed{class\_methods(cld) = \overline{meth\_def}}$ – extract method definitions from a class

CLASS_METHODS

$$\overline{class\_methods(\textbf{class}\ dcl\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}) = \overline{meth\_def}}$$

$\boxed{method\_name(meth\_def) = meth}$ – extract the method name from a method definition

METHOD_NAME

$$\overline{method\_name(cl\ meth(\overline{vd})\{meth\_body\}) = meth}$$

$\boxed{distinct\_names(P)}$ – names of type definitions are distinct

DN_DEF

$$\frac{1.\ P = \overline{cld_k}^{\,k} \quad 2.\ \overline{class\_name(cld_k) = dcl_k}^{\,k} \quad 3.\ \textbf{distinct}\ (\overline{dcl_k}^{\,k})}{distinct\_names(P)}$$

$\boxed{find\_cld(P, ctx, fqn) = ctxcld_{opt}}$ – class lookup

FC_EMPTY

$$\frac{}{find\_cld([\,], ctx, fqn) = \textbf{null}}$$

FC_CONS_TRUE

$$\frac{\begin{array}{l}1.\ P = cld\ cld_2\ ..\ cld_k \\ 2.\ cld = \textbf{class}\ dcl\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}\end{array}}{find\_cld(P, ctx, dcl) = (ctx, cld)}$$

FC_CONS_FALSE

$$\frac{\begin{array}{l}1.\ cld = \textbf{class}\ dcl'\ \textbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\} \\ 2.\ dcl \neq dcl' \\ 3.\ find\_cld(cld_2\ ..\ cld_k, ctx, dcl) = ctxcld_{opt}\end{array}}{find\_cld(cld\ cld_2\ ..\ cld_k, ctx, dcl) = ctxcld_{opt}}$$

$\boxed{find\_type(P, ctx, cl) = \tau_{opt}}$ – type lookup

FT_OBJ

$$\frac{}{find\_type(P, ctx, \texttt{Object}) = \texttt{Object}}$$

FT_NULL

$$\frac{1.\ find\_cld(P, ctx, fqn) = \textbf{null}}{find\_type(P, ctx, fqn) = \textbf{null}}$$

FT_DCL

$$\frac{1.\ find\_cld(P, ctx, dcl) = (ctx', cld)}{find\_type(P, ctx, dcl) = ctx'.dcl}$$

$\boxed{(P, ctx, cl, nn) \in path\_length}$ – get the length of the inheritance path for cl

PL_OBJ

$$\frac{}{(P, ctx, \texttt{Object}, 0) \in path\_length}$$

PL_FQN

$$\frac{\begin{array}{l}1.\ find\_cld(P, ctx, fqn) = (ctx', cld) \\ 2.\ superclass\_name(cld) = cl \\ 3.\ (P, ctx', cl, nn) \in path\_length\end{array}}{(P, ctx, fqn, nn + 1) \in path\_length}$$

$\boxed{acyclic\_cldsP}$ – the class inheritance hierarchy in P is acyclic

AC_DEF

$$\frac{1.\ \forall ctx\ fqn.\left(\begin{array}{c}(\exists ctx'\ cld.find\_cld(P, ctx, fqn) = (ctx', cld)) \longrightarrow \\ \exists nn.(P, ctx, fqn, nn) \in path\_length\end{array}\right)}{acyclic\_cldsP}$$

$\boxed{find\_path\_rec(P, ctx, cl, \overline{ctxcld}) = \overline{ctxcld}_{opt}}$ – class path lookup (recursive part)

FPR_OBJ

$$\frac{}{find\_path\_rec(P, ctx, \texttt{Object}, \overline{ctxcld}) = \overline{ctxcld}}$$

FPR_NULL

$$\frac{1.\ (\neg acyclic\_cldsP)\ \vee\ find\_cld(P, ctx, fqn) = \textbf{null}}{find\_path\_rec(P, ctx, fqn, \overline{ctxcld}) = \textbf{null}}$$

FPR_FQN

$$\frac{\begin{array}{l}1.\ acyclic\_cldsP\ \wedge\ find\_cld(P, ctx, fqn) = (ctx', cld) \\ 2.\ superclass\_name(cld) = cl \\ 3.\ find\_path\_rec(P, ctx', cl, \overline{ctxcld}@[(ctx', cld)]) = \overline{ctxcld}_{opt}\end{array}}{find\_path\_rec(P, ctx, fqn, \overline{ctxcld}) = \overline{ctxcld}_{opt}}$$

$\boxed{find\_path(P, ctx, cl) = \overline{ctxcld}_{opt}}$ – class path lookup with a class name

FP_DEF

$$\frac{1.\ find\_path\_rec(P, ctx, cl, [\,]) = \overline{ctxcld}_{opt}}{find\_path(P, ctx, cl) = \overline{ctxcld}_{opt}}$$

$\boxed{find\_path(P, \tau) = \overline{ctxcld}_{opt}}$ – class path lookup with a type

FPTY_OBJ

$$\frac{}{find\_path(P, \texttt{Object}) = [\,]}$$

FPTY_DCL

$$\frac{1.\ find\_path(P, ctx, dcl) = \overline{ctxcld}_{opt}}{find\_path(P, ctx.dcl) = \overline{ctxcld}_{opt}}$$

$\boxed{fields\_in\_path(\overline{ctxcld}) = \overline{f}}$ – fields lookup in a class path

FIP_EMPTY

$$\frac{}{fields\_in\_path([\,]) = [\,]}$$

FIP_CONS

$$\frac{\begin{array}{l}1.\ class\_fields(cld) = \overline{cl_j\ f_j}^{\,j} \\ 2.\ fields\_in\_path(ctxcld_2\ ..\ ctxcld_k) = \overline{f} \\ 3.\ \overline{f}' = \overline{f_j}^{\,j}; \overline{f}\end{array}}{fields\_in\_path((ctx, cld)\ ctxcld_2\ ..\ ctxcld_k) = \overline{f}'}$$

$\boxed{\mathbf{fields}\ (P, \tau) = \overline{f}_{\,opt}}$ – fields lookup in type τ

FIELDS_NONE

$$\frac{1.\ find\_path(P, \tau) = \mathbf{null}}{\mathbf{fields}\ (P, \tau) = \mathbf{null}}$$

FIELDS_SOME

$$\frac{\begin{array}{l}1.\ find\_path(P, \tau) = \overline{ctxcld} \\ 2.\ fields\_in\_path(\overline{ctxcld}) = \overline{f}\end{array}}{\mathbf{fields}\ (P, \tau) = \overline{f}}$$

$\boxed{methods\_in\_path(\overline{cld}) = \overline{meth}}$ – method names lookup in a path

MIP_EMPTY

$$\frac{}{methods\_in\_path([\,]) = [\,]}$$

MIP_CONS

$$\frac{\begin{array}{l}1.\ class\_methods(cld) = \overline{meth\_def_l}^{\,l} \\ 2.\ \dfrac{}{meth\_def_l = cl_l\ meth_l\ (\overline{vd}_l)\{meth\_body_l\}}^{\,l} \\ 3.\ methods\_in\_path(cld_2\ ..\ cld_k) = \overline{meth}' \\ 4.\ \overline{meth} = \overline{meth_l}^{\,l}; \overline{meth}'\end{array}}{methods\_in\_path(cld\ cld_2\ ..\ cld_k) = \overline{meth}}$$

$\boxed{\mathbf{methods}\ (P, \tau) = \overline{meth}}$ – method names lookup in a type

METHODS_METHODS

$$\frac{\begin{array}{l}1.\ find\_path(P, \tau) = \overline{(ctx_k, cld_k)}^{\,k} \\ 2.\ methods\_in\_path(\overline{cld_k}^{\,k}) = \overline{meth}\end{array}}{\mathbf{methods}\ (P, \tau) = \overline{meth}}$$

$\boxed{ftype\_in\_fds(P, ctx, \overline{fd}, f) = \tau^{\perp}_{opt}}$ – field type lookup in a list

FTIF_EMPTY

$$\frac{}{ftype\_in\_fds(P, ctx, [\,], f) = \mathbf{null}}$$

FTIF_CONS_BOT

$$\frac{1.\ find\_type(P, ctx, cl) = \mathbf{null}}{ftype\_in\_fds(P, ctx, cl\,f;\ fd_2\ ..\ fd_k, f) = \perp}$$

FTIF_CONS_TRUE

$$\frac{1.\ find\_type(P, ctx, cl) = \tau}{ftype\_in\_fds(P, ctx, cl\,f;\ fd_2\ ..\ fd_k, f) = \tau}$$

FTIF_CONS_FALSE

$$\frac{\begin{array}{l}1.\ f \neq f' \\ 2.\ ftype\_in\_fds(P, ctx, fd_2\ ..\ fd_k, f') = \tau^{\perp}_{opt}\end{array}}{ftype\_in\_fds(P, ctx, cl\,f;\ fd_2\ ..\ fd_k, f') = \tau^{\perp}_{opt}}$$

$\boxed{ftype\_in\_path(P, \overline{ctxcld}, f) = \tau_{opt}}$ – field type lookup in a path

FTIP_EMPTY

$$\frac{}{ftype\_in\_path(P, [\,], f) = \mathbf{null}}$$

FTIP_CONS_BOT

$$\frac{\begin{array}{l}1.\ class\_fields(cld) = \overline{fd} \\ 2.\ ftype\_in\_fds(P, ctx, \overline{fd}, f) = \perp\end{array}}{ftype\_in\_path(P, (ctx, cld)\ ctxcld_2\ ..\ ctxcld_k, f) = \mathbf{null}}$$

FTIP_CONS_TRUE

$$\frac{\begin{array}{l}1.\ class\_fields(cld) = \overline{fd} \\ 2.\ ftype\_in\_fds(P, ctx, \overline{fd}, f) = \tau\end{array}}{ftype\_in\_path(P, (ctx, cld)\ ctxcld_2\ ..\ ctxcld_k, f) = \tau}$$

FTIP_CONS_FALSE

$$\frac{\begin{array}{l}1.\ class\_fields(cld) = \overline{fd} \\ 2.\ ftype\_in\_fds(P, ctx, \overline{fd}, f) = \mathbf{null} \\ 3.\ ftype\_in\_path(P, ctxcld_2\ ..\ ctxcld_k, f) = \tau_{opt}\end{array}}{ftype\_in\_path(P, (ctx, cld)\ ctxcld_2\ ..\ ctxcld_k, f) = \tau_{opt}}$$

$\boxed{\textbf{ftype}\,(P,\tau,f)=\tau'}$ – field type lookup

FTYPE

$$\frac{1.\ find\_path(P,\tau)=\overline{ctxcld} \quad 2.\ ftype\_in\_path(P,\overline{ctxcld},f)=\tau'}{\textbf{ftype}\,(P,\tau,f)=\tau'}$$

$\boxed{find\_meth\_def\_in\_list(\overline{meth\_def},meth)=meth\_def_{opt}}$ – meth. def. lookup (list)

FMDIL_EMPTY

$$\frac{}{find\_meth\_def\_in\_list([\,],meth)=\textbf{null}}$$

FMDIL_CONS_TRUE

$$\frac{1.\ meth\_def=cl\ meth(\overline{vd})\{meth\_body\}}{find\_meth\_def\_in\_list(meth\_def\ meth\_def_2\ ..\ meth\_def_k,meth)=meth\_def}$$

FMDIL_CONS_FALSE

$$\frac{1.\ meth\_def=cl\ meth'(\overline{vd})\{meth\_body\} \quad 2.\ meth\neq meth' \quad 3.\ find\_meth\_def\_in\_list(meth\_def_2\ ..\ meth\_def_k,meth)=meth\_def_{opt}}{find\_meth\_def\_in\_list(meth\_def\ meth\_def_2\ ..\ meth\_def_k,meth)=meth\_def_{opt}}$$

$\boxed{find\_meth\_def\_in\_path(\overline{ctxcld},meth)=ctxmeth\_def_{opt}}$ – meth. def. lookup (path)

FMDIP_EMPTY

$$\frac{}{find\_meth\_def\_in\_path([\,],meth)=\textbf{null}}$$

FMDIP_CONS_TRUE

$$\frac{1.\ class\_methods(cld)=\overline{meth\_def} \quad 2.\ find\_meth\_def\_in\_list(\overline{meth\_def},meth)=meth\_def}{find\_meth\_def\_in\_path((ctx,cld)\ ctxcld_2\ ..\ ctxcld_k,meth)=(ctx,meth\_def)}$$

FMDIP_CONS_FALSE

$$\frac{1.\ class\_methods(cld)=\overline{meth\_def} \quad 2.\ find\_meth\_def\_in\_list(\overline{meth\_def},meth)=\textbf{null} \quad 3.\ find\_meth\_def\_in\_path(ctxcld_2\ ..\ ctxcld_k,meth)=ctxmeth\_def_{opt}}{find\_meth\_def\_in\_path((ctx,cld)\ ctxcld_2\ ..\ ctxcld_k,meth)=ctxmeth\_def_{opt}}$$

$\boxed{find\_meth\_def(P,\tau,meth)=ctxmeth\_def_{opt}}$ – method def. lookup in a type

FMD_NULL

$$\frac{1.\ find\_path(P,\tau)=\textbf{null}}{find\_meth\_def(P,\tau,meth)=\textbf{null}}$$

FMD_OPT

$$\frac{1.\ find\_path(P,\tau)=\overline{ctxcld} \quad 2.\ find\_meth\_def\_in\_path(\overline{ctxcld},meth)=ctxmeth\_def_{opt}}{find\_meth\_def(P,\tau,meth)=ctxmeth\_def_{opt}}$$

$\boxed{\textbf{mtype}\,(P,\tau,meth)=\pi}$ – method type lookup

MTYPE

$$\frac{\begin{array}{l}1.\ find\_meth\_def(P,\tau,meth)=(ctx,meth\_def)\\ 2.\ meth\_def=cl\ meth(\overline{cl_k\ var_k}^{\,k})\{meth\_body\}\\ 3.\ find\_type(P,ctx,cl)=\tau'\\ 4.\ \overline{find\_type(P,ctx,cl_k)=\tau_k}^{\,k}\\ 5.\ \pi=\overline{\tau_k}^{\,k}\rightarrow\tau'\end{array}}{\textbf{mtype}\,(P,\tau,meth)=\pi}$$

$\boxed{P \vdash \tau \prec \tau'}$ – subtyping

STY_OBJ

$$\frac{1.\ find\_path(P,\tau) = \overline{ctx\,cld}}{P \vdash \tau \prec \texttt{Object}}$$

STY_DCL

$$\frac{\begin{array}{l}1.\ find\_path(P,\tau) = \overline{(ctx_k,\ cld_k)}^{\,k} \\ 2.\ \overline{class\_name(cld_k) = dcl_k}^{\,k} \\ 3.\ (ctx',\ dcl') \in \overline{(ctx_k,\ dcl_k)}^{\,k}\end{array}}{P \vdash \tau \prec ctx'.dcl'}$$

$\boxed{P \vdash \overline{\tau} \prec \overline{\tau}'}$ – normal, multiple subtyping

STY_MANY

$$\frac{\begin{array}{l}1.\ \overline{\tau} = \overline{\tau_k}^{\,k} \\ 2.\ \overline{\tau}' = \overline{\tau'_k}^{\,k} \\ 3.\ \overline{P \vdash \tau_k \prec \tau'_k}^{\,k}\end{array}}{P \vdash \overline{\tau} \prec \overline{\tau}'}$$

$\boxed{P \vdash \tau_{opt} \prec \tau_{opt}{}'}$ – option subtyping

STY_OPTION

$$\frac{\begin{array}{l}1.\ \tau_{opt} = \tau \\ 2.\ \tau_{opt}{}' = \tau' \\ 3.\ P \vdash \tau \prec \tau'\end{array}}{P \vdash \tau_{opt} \prec \tau_{opt}{}'}$$

$\boxed{P, H \vdash v_{opt} \prec \tau_{opt}}$ – well-formed value

WF_NULL

$$\frac{1.\ \tau_{opt} = \tau}{P, H \vdash \mathbf{null} \prec \tau_{opt}}$$

WF_OBJECT

$$\frac{1.\ P \vdash H(oid) \prec \tau_{opt}}{P, H \vdash oid \prec \tau_{opt}}$$

$\boxed{P, \Gamma, H \vdash L}$ – well-formed variable state

WF_VARSTATE

$$\frac{\begin{array}{l}1.\ \mathbf{finite}\,(\mathbf{dom}\,(L)) \\ 2.\ \forall x \in \mathbf{dom}\,(\Gamma).P, H \vdash L(x) \prec \Gamma(x)\end{array}}{P, \Gamma, H \vdash L}$$

$\boxed{P \vdash H}$ – well-formed heap

WF_HEAP

$$\frac{\begin{array}{l}1.\ \mathbf{finite}\,(\mathbf{dom}\,(H)) \\ 2.\ \forall oid \in \mathbf{dom}\,(H).\left(\begin{array}{r}\exists \tau.H(oid) = \tau\ \wedge\ \exists \overline{f}.\mathbf{fields}\,(P,\tau) = \overline{f}\ \wedge \\ \forall f \in \overline{f}.\exists \tau'.\left(\begin{array}{l}\mathbf{ftype}\,(P,\tau,f) = \tau'\ \wedge \\ P, H \vdash H(oid, f) \prec \tau'\end{array}\right)\end{array}\right)\end{array}}{P \vdash H}$$

$\boxed{\Gamma \vdash \mathit{config}}$ – well-formed configuration

WF_ALL_EX

$$\frac{1.\ \vdash P \quad 2.\ P \vdash H \quad 3.\ P, \Gamma, H \vdash L}{\Gamma \vdash (P, L, H, Exception)}$$

WF_ALL

$$\frac{\begin{array}{l}1.\ \vdash P \\ 2.\ P \vdash H \\ 3.\ P, \Gamma, H \vdash L \\ 4.\ \overline{P, \Gamma \vdash s_k}^{\,k}\end{array}}{\Gamma \vdash (P, L, H, \overline{s_k}^{\,k})}$$

$\boxed{P, \Gamma \vdash s}$ – well-formed statement

WF_BLOCK

$$\frac{1.\ \overline{P, \Gamma \vdash s_k}^{\,k}}{P, \Gamma \vdash \{\ \overline{s_k}^{\,k}\ \}}$$

WF_VAR_ASSIGN

$$\frac{1.\ P \vdash \Gamma(x) \prec \Gamma(var)}{P, \Gamma \vdash var = x;}$$

WF_FIELD_READ

$$\frac{1.\ \Gamma(x) = \tau \quad 2.\ \mathbf{ftype}\ (P, \tau, f) = \tau' \quad 3.\ P \vdash \tau' \prec \Gamma(var)}{P, \Gamma \vdash var = x.f;}$$

WF_FIELD_WRITE

$$\frac{1.\ \Gamma(x) = \tau \quad 2.\ \mathbf{ftype}\ (P, \tau, f) = \tau' \quad 3.\ P \vdash \Gamma(y) \prec \tau'}{P, \Gamma \vdash x.f = y;}$$

WF_IF

$$\frac{1.\ P \vdash \Gamma(x) \prec \Gamma(y)\ \vee\ P \vdash \Gamma(y) \prec \Gamma(x) \quad 2.\ P, \Gamma \vdash s_1 \quad 3.\ P, \Gamma \vdash s_2}{P, \Gamma \vdash \mathbf{if}\ (x == y) s_1\ \mathbf{else}\ s_2}$$

WF_NEW

$$\frac{1.\ find\_type(P, ctx, cl) = \tau \quad 2.\ P \vdash \tau \prec \Gamma(var)}{P, \Gamma \vdash var = \mathbf{new}_{ctx}\, cl();}$$

WF_MCALL

$$\frac{1.\ \overline{y} = \overline{y_k}^{\,k} \quad 2.\ \Gamma(x) = \tau \quad 3.\ \mathbf{mtype}\ (P, \tau, meth) = \overline{\tau_k}^{\,k} \to \tau' \quad 4.\ \overline{P \vdash \Gamma(y_k) \prec \tau_k}^{\,k} \quad 5.\ P \vdash \tau' \prec \Gamma(var)}{P, \Gamma \vdash var = x.meth(\overline{y});}$$

$\boxed{P \vdash_\tau meth\_def}$ – well-formed method in τ

WF_METHOD

$$\frac{1.\ \mathbf{distinct}\ (\overline{var_k}^{\,k}) \quad 2.\ \overline{find\_type(P, ctx, cl_k) = \tau_k}^{\,k} \quad 3.\ \Gamma = [\overline{var_k \mapsto \tau_k}^{\,k}][\mathbf{this} \mapsto ctx.dcl] \quad 4.\ \overline{P, \Gamma \vdash s_l}^{\,l} \quad 5.\ find\_type(P, ctx, cl) = \tau \quad 6.\ P \vdash \Gamma(y) \prec \tau}{P \vdash_{ctx.dcl} cl\ meth(\overline{cl_k\ var_k}^{\,k})\{\ \overline{s_l}^{\,l}\ \mathbf{return}\ y;\ \}}$$

$\boxed{P \vdash_{ctx} (dcl, cl, \overline{fd}, \overline{meth\_def})}$ – well-formed class in ctx (generic rule)

WF_CLASS_COMMON

$$\frac{\begin{array}{l}1.\ find\_type(P, ctx, cl) = \tau \quad 2.\ ctx.dcl \neq \tau \quad 3.\ \mathbf{distinct}\ (\overline{f_j}^{\,j}) \quad 4.\ \mathbf{fields}\ (P, \tau) = \overline{f} \quad 5.\ \overline{f_j}^{\,j} \perp \overline{f} \\ 6.\ \overline{find\_type(P, ctx, cl_j) = \tau_j}^{\,j} \quad 7.\ \overline{P \vdash_{ctx.dcl} meth\_def_k}^{\,k} \\ 8.\ \overline{method\_name(meth\_def_k) = meth_k}^{\,k} \quad 9.\ \mathbf{distinct}\ (\overline{meth_k}^{\,k}) \\ 10.\ \mathbf{methods}\ (P, \tau) = \overline{meth'_l}^{\,l} \quad 11.\ \overline{\mathbf{mtype}\ (P, ctx.dcl, meth'_l) = \pi_l}^{\,l} \\ 12.\ \overline{\mathbf{mtype}\ (P, \tau, meth'_l) = \pi'_l}^{\,l} \quad 13.\ \overline{meth'_l \in \overline{meth_k}^{\,k} \longrightarrow \pi_l = \pi'_l}^{\,l}\end{array}}{P \vdash_{ctx} (dcl, cl, \overline{cl_j\ f_j;}^{\,j}, \overline{meth\_def_k}^{\,k})}$$

$\boxed{P \vdash cld}$ – well-formed class

WF_CLASS

$$\frac{1.\ \mathbf{class}\ dcl\ \mathbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\} \in P \quad 2.\ P \vdash (dcl, cl, \overline{fd}, \overline{meth\_def})}{P \vdash \mathbf{class}\ dcl\ \mathbf{extends}\ cl\{\overline{fd}\ \overline{meth\_def}\}}$$

$\boxed{\vdash P}$ – well-formed program

WF_PROGRAM

$$\frac{1.\ P = \overline{cld_k}^{\,k} \quad 2.\ distinct\_names(P) \quad 3.\ \overline{P \vdash cld_k}^{\,k} \quad 4.\ acyclic\_cldsP}{\vdash P}$$

$\boxed{\theta \vdash s \rightsquigarrow s'}$ – variable translation for a statement

$$\text{TR\_S\_BLOCK}\ \frac{1.\ \overline{\theta \vdash s_k \rightsquigarrow s'_k}^{\,k}}{\theta \vdash \{\ \overline{s_k}^{\,k}\ \} \rightsquigarrow \{\ \overline{s'_k}^{\,k}\ \}} \qquad \text{TR\_S\_VAR\_ASSIGN}\ \frac{1.\ \theta(var) = var' \quad 2.\ \theta(x) = x'}{\theta \vdash var = x; \rightsquigarrow var' = x';}$$

$$\text{TR\_S\_FIELD\_READ}\ \frac{1.\ \theta(var) = var' \quad 2.\ \theta(x) = x'}{\theta \vdash var = x.f; \rightsquigarrow var' = x'.f;}$$

$$\text{TR\_S\_FIELD\_WRITE}\ \frac{1.\ \theta(x) = x' \quad 2.\ \theta(y) = y'}{\theta \vdash x.f = y; \rightsquigarrow x'.f = y';}$$

$$\text{TR\_S\_IF}\ \frac{1.\ \theta(x) = x' \quad 2.\ \theta(y) = y' \quad 3.\ \theta \vdash s_1 \rightsquigarrow s'_1 \quad 4.\ \theta \vdash s_2 \rightsquigarrow s'_2}{\theta \vdash \textbf{if}\ (x == y) s_1\ \textbf{else}\ s_2 \rightsquigarrow \textbf{if}\ (x' == y') s'_1\ \textbf{else}\ s'_2}$$

$$\text{TR\_S\_NEW}\ \frac{1.\ \theta(var) = var'}{\theta \vdash var = \textbf{new}_{ctx}\, cl(); \rightsquigarrow var' = \textbf{new}_{ctx}\, cl();} \qquad \text{TR\_S\_MCALL}\ \frac{1.\ \theta(var) = var' \quad 2.\ \theta(x) = x' \quad 3.\ \overline{\theta(y_k) = y'_k}^{\,k}}{\theta \vdash var = x.meth(\overline{y_k}^{\,k}); \rightsquigarrow var' = x'.meth(\overline{y'_k}^{\,k});}$$

$\boxed{config \longrightarrow config'}$ – reduction of a statement

$$\text{R\_BLOCK}\ \frac{}{(P, L, H, \{\ \overline{s_k}^{\,k}\ \}\ \overline{s'_l}^{\,l}) \longrightarrow (P, L, H,\ \overline{s_k}^{\,k}\ \overline{s'_l}^{\,l})} \qquad \text{R\_VAR\_ASSIGN}\ \frac{1.\ L(x) = v}{(P, L, H, var = x;\ \overline{s_l}^{\,l}) \longrightarrow (P, L[var \mapsto v], H,\ \overline{s_l}^{\,l})}$$

$$\text{R\_FIELD\_READ\_NPE}\ \frac{1.\ L(x) = \textbf{null}}{(P, L, H, var = x.f;\ \overline{s_l}^{\,l}) \longrightarrow (P, L, H, \textbf{NPE})} \qquad \text{R\_FIELD\_READ}\ \frac{1.\ L(x) = oid \quad 2.\ H(oid, f) = v}{(P, L, H, var = x.f;\ \overline{s_l}^{\,l}) \longrightarrow (P, L[var \mapsto v], H,\ \overline{s_l}^{\,l})}$$

$$\text{R\_FIELD\_WRITE\_NPE}\ \frac{1.\ L(x) = \textbf{null}}{(P, L, H, x.f = y;\ \overline{s_l}^{\,l}) \longrightarrow (P, L, H, \textbf{NPE})} \qquad \text{R\_FIELD\_WRITE}\ \frac{1.\ L(x) = oid \quad 2.\ L(y) = v}{(P, L, H, x.f = y;\ \overline{s_l}^{\,l}) \longrightarrow (P, L, H[(oid, f) \mapsto v],\ \overline{s_l}^{\,l})}$$

$$\text{R\_IF\_TRUE}\ \frac{1.\ L(x) = v \quad 2.\ L(y) = w \quad 3.\ v = w}{(P, L, H, \textbf{if}\ (x == y) s_1\ \textbf{else}\ s_2\ \overline{s'_l}^{\,l}) \longrightarrow (P, L, H,\ s_1\ \overline{s'_l}^{\,l})} \qquad \text{R\_IF\_FALSE}\ \frac{1.\ L(x) = v \quad 2.\ L(y) = w \quad 3.\ v \neq w}{(P, L, H, \textbf{if}\ (x == y) s_1\ \textbf{else}\ s_2\ \overline{s'_l}^{\,l}) \longrightarrow (P, L, H,\ s_2\ \overline{s'_l}^{\,l})}$$

$$\text{R\_NEW}\ \frac{1.\ find\_type(P, ctx, cl) = \tau \quad 2.\ \textbf{fields}\,(P, \tau) = \overline{f_k}^{\,k} \quad 3.\ oid \notin \textbf{dom}\,(H) \quad 4.\ H' = H[oid \mapsto (\tau,\ \overline{f_k \mapsto \textbf{null}}^{\,k})]}{(P, L, H, var = \textbf{new}_{ctx}\, cl();\ \overline{s_l}^{\,l}) \longrightarrow (P, L[var \mapsto oid], H',\ \overline{s_l}^{\,l})}$$

$$\text{R\_MCALL\_NPE}\ \frac{1.\ L(x) = \textbf{null}}{(P, L, H, var = x.meth(\overline{y_k}^{\,k});\ \overline{s_l}^{\,l}) \longrightarrow (P, L, H, \textbf{NPE})}$$

$$\text{R\_MCALL}\ \frac{\begin{array}{l}1.\ L(x) = oid \qquad 2.\ H(oid) = \tau \\ 3.\ find\_meth\_def(P, \tau, meth) = (ctx, cl\ meth(\overline{cl_k\ var_k}^{\,k})\{\overline{s'_j}^{\,j}\ \textbf{return}\ y;\}) \\ 4.\ \overline{var'_k}^{\,k} \perp \textbf{dom}\,(L) \quad 5.\ \textbf{distinct}\,(\overline{var'_k}^{\,k}) \quad 6.\ x' \notin \textbf{dom}\,(L) \\ 7.\ x' \notin \overline{var'_k}^{\,k} \qquad 8.\ \overline{L(y_k) = v_k}^{\,k} \\ 9.\ L' = L[\overline{var'_k \mapsto v_k}^{\,k}][x' \mapsto oid] \\ 10.\ \theta = [\overline{var_k \mapsto var'_k}^{\,k}][\textbf{this} \mapsto x'] \qquad 11.\ \overline{\theta \vdash s'_j \rightsquigarrow s''_j}^{\,j} \\ 12.\ \theta(y) = y'\end{array}}{(P, L, H, var = x.meth(\overline{y_k}^{\,k});\ \overline{s_l}^{\,l}) \longrightarrow (P, L', H,\ \overline{s''_j}^{\,j}\ var = y';\ \overline{s_l}^{\,l})}$$