Summary of the DJ Formal System

Alexander Ahern and Nobuko Yoshida

April 7, 2005

1 Syntax

Syntax occurring only at runtime appears in shaded regions.

(Types)	$T ::=$	$\texttt{bool} \mid C \mid \vec{U} \to U \mid \texttt{ser}\langle \vec{U} \rangle$
(Returnable)	$U ::=$	$\texttt{void} \mid T$
(Classes)	$L ::=$	$\texttt{class}\, C\, \texttt{extends}\, D\{\vec{T}\,\vec{f}; K\,\vec{M}\}$
(Constructors)	$K ::=$	$C\,(\vec{T}\,\vec{f})\{\texttt{super}(\vec{f}); \texttt{this}.\vec{\texttt{f}} := \vec{f}\}$
(Methods)	$M ::=$	$U\,\texttt{m}\,(\vec{T}\vec{x})\{e\}$
(Expressions)	$e ::=$	$v \mid x \mid \texttt{this} \mid \texttt{if}\ (e)\ \{e\}\ \texttt{else}\ \{e\} \mid \texttt{while}\ (e)\ \{e\} \mid e.\texttt{f} \mid e;e$
	$\mid$	$x := e \mid e.\texttt{f} := e \mid \texttt{new}\, C(\vec{e}) \mid e.\texttt{m}(\vec{e}) \mid T\ x = e \mid \texttt{return}\ e$
	$\mid$	$\texttt{return} \mid \texttt{freeze}[t](\vec{T}\ \vec{x};\ e) \mid \texttt{defrost}(\vec{e};\ e) \mid \texttt{fork}(e)$
	$\mid$	$\texttt{sync}\ (e)\ \{e\} \mid e.\texttt{wait} \mid e.\texttt{notify} \mid e.\texttt{notifyAll}$
	$\mid$	$\texttt{new}\, C^l(\vec{v}) \mid \texttt{download}\ \vec{C}\ \texttt{from}\ l\ \texttt{in}\ e \mid \texttt{resolve}\ \vec{C}\ \texttt{from}\ l\ \texttt{in}\ e$
	$\mid$	$\texttt{await}\ c \mid \texttt{insync}\ o\ \{e\} \mid \texttt{ready}\ o\ n \mid \texttt{waiting}(c)\ n \mid \texttt{Error}$
(Tags)	$t ::=$	$\texttt{eager} \mid \texttt{lazy} \mid \vec{C}$
(Values)	$v ::=$	$\texttt{true} \mid \texttt{false} \mid \texttt{null} \mid \vec{v} \mid o \mid \lambda(\vec{T}\ \vec{x}).(\nu\,\vec{u})(l,e,\sigma,\texttt{CT})$
(Class Sig.)	$\mathsf{CSig} ::=$	$\emptyset \mid \mathsf{CSig}\cdot C : \texttt{extends}\ D\ [\texttt{remote}]\ \vec{T}\vec{\texttt{f}}\ \ \{\texttt{m}_\texttt{i} : \vec{T_i} \to U_i\}$
(Identifiers)	$u ::=$	$x \mid o \mid c$
(Threads)	$P ::=$	$\mathbf{0} \mid P_1 \,\vert\, P_2 \mid (\nu\,u)P \mid \texttt{forked}\ e \mid \texttt{go}\ e\ \texttt{with}\ c \mid e\ \texttt{with}\ c$
	$\mid$	$\texttt{go}\ e\ \texttt{to}\ c \mid \texttt{return}(c)\ e \mid \texttt{Error}$
(Configurations)	$F ::=$	$(\nu\,\vec{u})(P,\sigma,\texttt{CT})$
(Networks)	$N ::=$	$\mathbf{0} \mid l[F] \mid N_1 \,\vert\, N_2 \mid (\nu\,u)N$
(Stores)	$\sigma ::=$	$\emptyset \mid \sigma\cdot[x \mapsto v] \mid \sigma\cdot[o \mapsto (C, \vec{f} : \vec{v}, n, \{\vec{c}\})]$
(Class tables)	$\texttt{CT} ::=$	$\emptyset \mid \texttt{CT}\cdot[C \mapsto L]$

For clarity, we introduce two *derived* constructs that are syntactic sugar for serialisation and deserialisation.

$$\begin{aligned} \texttt{serialize}(\vec{u}) &\stackrel{\text{def}}{=} \texttt{freeze}[\texttt{lazy}](\varepsilon;\ \vec{u}) \\ \texttt{deserialize}(e) &\stackrel{\text{def}}{=} \texttt{defrost}(\varepsilon;\ e) \end{aligned}$$

Evaluation of an expression results in a value, however when this value can be safely ignored we say that the expression is *promotable*. Promotable expressions are defined as:

$$pe ::= x := e \mid e.\texttt{f} := e \mid \texttt{new}\, C(\vec{e}) \mid e.\texttt{m}(\vec{e}) \mid \texttt{while}\ (e)\ \{e\}$$

1.1 Evaluation contexts

$$\begin{aligned} E ::= & [\,] \mid \texttt{if}\ (E)\ \{e\}\ \texttt{else}\ \{e\} \mid E.\texttt{f} \mid E;e \mid x := E \\ & \mid E.\texttt{f} := e \mid o.\texttt{f} := E \mid \texttt{new}\, C^{-}(\vec{v},E,\vec{e}) \mid E.\texttt{m}(\vec{e}) \mid o.\texttt{m}(\vec{v},E,\vec{e}) \\ & \mid T\ x = E \mid \texttt{defrost}(\vec{v},E,\vec{e};\ e) \mid \texttt{defrost}(\vec{v};\ E) \\ & \mid \texttt{sync}\ (E)\ \{e\} \mid E.\texttt{wait} \mid E.\texttt{notify} \mid E.\texttt{notifyAll} \\ & \mid \texttt{insync}\ o\ \{E\} \mid \texttt{forked}\ E \mid \texttt{go}\ E\ \texttt{with}\ c \mid E\ \texttt{with}\ c \\ & \mid \texttt{go}\ E\ \texttt{to}\ c \mid \texttt{return}(c)\ E \end{aligned}$$

2 Auxiliary Definitions

2.1 Structural equivalence

Configurations

$$\begin{aligned} (\nu\, u)P,\sigma,\texttt{CT} &\equiv (\nu\, u)(P,\sigma,\texttt{CT}) && u \notin \mathsf{fn}(\sigma) \cup \mathsf{fn}(\texttt{CT}) \\ (\nu\, u)(\nu\, u')F &\equiv (\nu\, u')(\nu\, u)F && \\ (\nu\, x)(P,\sigma \cdot [x \mapsto v],\texttt{CT}) &\equiv P,\sigma,\texttt{CT} && x \notin \mathsf{fv}(P) \\ (\nu\, o)(P,\sigma \cdot [o \mapsto (C,\vec{f} : \vec{v},n,\{\vec{c}\})],\texttt{CT}) &\equiv P,\sigma,\texttt{CT} && o \notin \mathsf{fn}(P) \cup \mathsf{fn}(\sigma) \end{aligned}$$

Threads

$$\begin{aligned} P \mid \mathbf{0} &\equiv P \\ P \mid P_0 &\equiv P_0 \mid P \\ P \mid (P_0 \mid P_1) &\equiv (P \mid P_0) \mid P_1 \\ (\nu\, u)(P \mid P_0) &\equiv (\nu\, u)P \mid P_0 \quad u \notin \mathsf{fn}(P_0) \\ (\nu\, c)\mathbf{0} &\equiv \mathbf{0} \\ (\nu\, u)(\nu\, u')P &\equiv (\nu\, u')(\nu\, u)P \\ \texttt{return}(d)\ \varepsilon &\equiv \texttt{return}(d) \\ \varepsilon;e &\equiv e \\ \texttt{return}\ \varepsilon &\equiv \texttt{return} \end{aligned}$$

Networks

$$\begin{aligned} N \mid \mathbf{0} &\equiv N \\ N \mid N_0 &\equiv N_0 \mid N \\ N \mid (N_0 \mid N_1) &\equiv (N \mid N_0) \mid N_1 \\ (\nu\, u)(N \mid N_0) &\equiv (\nu\, u)N \mid N_0 \quad u \notin \mathsf{fnv}(N_0) \\ (\nu\, c)\mathbf{0} &\equiv \mathbf{0} \\ (\nu\, u)(\nu\, u')N &\equiv (\nu\, u')(\nu\, u)N \\ l[(\nu\, u)(F)] &\equiv (\nu\, u)l[F] \end{aligned}$$

2.2 Lookup functions

Field lookup

$$\mathsf{fields}(Object) = \bullet \qquad \frac{\mathsf{CSig}(C) = \texttt{extends}\ D\ \ \vec{T}\,\vec{\texttt{f}}\ \ \{\texttt{m}_\texttt{i} : \vec{T_i} \rightarrow U_i\} \qquad \mathsf{fields}(D) = \vec{T}'\,\vec{f}'}{\mathsf{fields}(C) = \vec{T}'\,\vec{f}', \vec{T}\,\vec{f}}$$

Method type lookup

$$\frac{\mathsf{CSig}(C){=}\texttt{extends}\ D\ [\texttt{remote}]\ \vec{T}\,\vec{\texttt{f}}\ \ \{\texttt{m}_\texttt{i}{:}\vec{T_i}{\rightarrow}U_i\}}{\mathsf{mtype}(\texttt{m}_\texttt{i}, C) = \vec{T_i}' \rightarrow U_i'} \qquad \frac{\mathsf{CSig}(C){=}\texttt{extends}\ D\ [\texttt{remote}]\ \vec{T}\,\vec{\texttt{f}}\ \ \{\texttt{m}_\texttt{i}{:}\vec{T_i}{\rightarrow}U_i\} \quad m{\notin}\{\vec{m}\}}{\mathsf{mtype}(\texttt{m}, C) = \mathsf{mtype}(\texttt{m}, D)}$$

Method body lookup

$$\frac{\texttt{CT}(C){=}\texttt{class}\ C\ \texttt{extends}\ D\{\vec{T}\,\vec{f};K\,\vec{M}\} \qquad U\,\texttt{m}\,(\vec{T}\vec{x})\{e\} \in \vec{M}}{\mathsf{mbody}(\texttt{m}, C, \texttt{CT}) = (\vec{x}, e)} \qquad \frac{\texttt{CT}(C){=}\texttt{class}\ C\ \texttt{extends}\ D\{\vec{T}\,\vec{f};K\,\vec{M}\} \qquad U\,\texttt{m}\,(\vec{T}\vec{x})\{e\} \notin \vec{M}}{\mathsf{mbody}(\texttt{m}, C, \texttt{CT}) = \mathsf{mbody}(\texttt{m}, D, \texttt{CT})}$$

Valid method overriding

$$\frac{\mathsf{mtype}(\texttt{m}, D) = \vec{T} \rightarrow U \text{ implies } \vec{T} = \vec{T}' \text{ and } U = U'}{\mathsf{override}(\texttt{m}, D, \vec{T}' \rightarrow U')}$$

2.3 Lock and queue manipulation

The predicate $\mathsf{insync}(o, E)$ is true if there exist E_1 and E_2 such that $E = E_1[\texttt{insync}\ o\ \{E_2[\,]\}]$. We also use the following functions. Let $\sigma(o) = (C, \vec{f} : \vec{v}, n, \{\vec{c}\})$.

- read/update the counter:

$$\mathsf{setLock}(\sigma, o, n') = \sigma[o \mapsto (C, \vec{f} : \vec{v}, n', \{\vec{c}\})]$$
$$\mathsf{getLock}(\sigma, o) = n$$

- read/update the queue:

$$\mathsf{blocked}(\sigma, o) = \{\vec{c}\}$$
$$\mathsf{block}(\sigma, o, c) = \sigma[o \mapsto (C, \vec{f} : \vec{v}, n, \{\vec{c}\} \cup \{c\})]$$
$$\mathsf{unblock}(\sigma, o, \vec{c}') = \sigma[o \mapsto (C, \vec{f} : \vec{v}, n, \{\vec{c}\} \setminus \{\vec{c}'\})]$$

Operational Semantics

[Expression]

RC-Var

$x, \sigma, \texttt{CT} \longrightarrow_l \sigma(x), \sigma, \texttt{CT}$

RC-Cond

$\texttt{if}\ (\texttt{true})\ \{e_1\}\ \texttt{else}\ \{e_2\}, \sigma, \texttt{CT} \longrightarrow_l e_1, \sigma, \texttt{CT}$

$\texttt{if}\ (\texttt{false})\ \{e_1\}\ \texttt{else}\ \{e_2\}, \sigma, \texttt{CT} \longrightarrow_l e_2, \sigma, \texttt{CT}$

RC-While

$\texttt{while}\ (e_1)\ \{e_2\}, \sigma, \texttt{CT} \longrightarrow_l \texttt{if}\ (e_1)\ \{e_2; \texttt{while}\ (e_1)\ \{e_2\}\}\ \texttt{else}\ \{\varepsilon\}, \sigma, \texttt{CT}$

RC-Fld

$$\frac{\sigma(o) = (C, \vec{f} : \vec{v}, n, \vec{c})}{o.\texttt{f}_\texttt{i}, \sigma, \texttt{CT} \longrightarrow_l v_i, \sigma, \texttt{CT}}$$

RC-Seq

$$\frac{e_1, \sigma, \texttt{CT} \longrightarrow_l (\nu\, \vec{u})(v, \sigma', \texttt{CT}')}{e_1; e_2, \sigma, \texttt{CT} \longrightarrow_l (\nu\, \vec{u})(e_2, \sigma', \texttt{CT}')}\ \vec{u} \notin \mathsf{fnv}(e_2)$$

RC-Ass

$x := v, \sigma, \texttt{CT} \longrightarrow_l v, \sigma[x \mapsto v], \texttt{CT}$

RC-FldAss

$$\frac{\sigma' = \sigma[o \mapsto \sigma(o)[f \mapsto v]]}{o.\texttt{f} := v, \sigma, \texttt{CT} \longrightarrow_l v, \sigma', \texttt{CT}}\ o \in \mathsf{dom}_\mathsf{o}(\sigma)$$

RC-New

$$\frac{\mathsf{fields}(C) = \vec{T}\,\vec{f}}{\texttt{new}\ C^{-}(\vec{v}), \sigma, \texttt{CT} \longrightarrow_l (\nu\, o)(o, \sigma \cdot [o \mapsto (C, \vec{f} : \vec{v},], ,)\texttt{CT})}\ C \in \mathsf{dom}(\texttt{CT})$$

RC-NewR

$\texttt{new}\ C^m(\vec{v}), \sigma, \texttt{CT} \longrightarrow_l \texttt{download}\ C\ \texttt{from}\ m\ \texttt{in}\ \texttt{new}\ C(\vec{v}), \sigma, \texttt{CT} \quad C \notin \mathsf{dom}(\texttt{CT})$

RC-Dec

$T\ x = v; e, \sigma, \texttt{CT} \longrightarrow_l (\nu\, x)(e, \sigma \cdot [x \mapsto v], \texttt{CT})\ \ x \notin \mathsf{dom}_\mathsf{v}(\sigma)$

RC-Cong

$$\frac{e, \sigma, \texttt{CT} \longrightarrow_l (\nu\, \vec{u})(e', \sigma', \texttt{CT}')}{E[e], \sigma, \texttt{CT} \longrightarrow_l (\nu\, \vec{u})(E[e'], \sigma', \texttt{CT}')}\ \vec{u} \notin \mathsf{fnv}(E)$$

[Synchronisation]

Fork

$$E[\texttt{fork}(e)],\sigma,\texttt{CT} \longrightarrow_l E[\varepsilon] \,|\, \texttt{forked}\ e,\sigma,\texttt{CT}$$

ThreadDeath

$$\texttt{forked}\ v,\sigma,\texttt{CT} \longrightarrow_l \mathbf{0},\sigma,\texttt{CT}$$

Sync

$$\frac{\mathsf{getLock}(\sigma,o) = \begin{cases} 0 & \mathsf{setLock}(\sigma,o,1) = \sigma' \\ n > 0 & \mathsf{insync}(o,E) \implies \mathsf{setLock}(\sigma,o,n+1) = \sigma' \end{cases}}{E[\texttt{sync}\ (o)\ \{e\}],\sigma,\texttt{CT} \longrightarrow_l E[\texttt{insync}\ o\ \{e\}],\sigma',\texttt{CT}}$$

Wait

$$\frac{\mathsf{insync}(o,E) \qquad \mathsf{getLock}(\sigma,o) = n \qquad \mathsf{setLock}(\sigma,o,0) = \sigma'' \qquad \mathsf{block}(\sigma'',o,c) = \sigma'}{E[o.\texttt{wait}] \,|\, Q,\sigma,\texttt{CT} \longrightarrow_l (\nu\, c)(E[\texttt{waiting}(c)\ on] \,|\, Q,\sigma',\texttt{CT})}$$

Notify

$$\frac{\mathsf{insync}(o,E) \qquad c \in \mathsf{blocked}(\sigma,o) \qquad \mathsf{unblock}(\sigma,o,c) = \sigma'}{E[o.\texttt{notify}] \,|\, E_1[\texttt{waiting}(c)\ on],\sigma,\texttt{CT} \longrightarrow_l E[\varepsilon] \,|\, E_1[\texttt{ready}\ o\ n],\sigma',\texttt{CT}}$$

NotifyAll

$$\frac{\mathsf{insync}(o,E) \qquad \mathsf{blocked}(\sigma,o) = \{\vec{c}\} \qquad m \geq 0 \qquad \mathsf{unblock}(\sigma,o,\vec{c}) = \sigma'}{\begin{array}{c} E[o.\texttt{notifyAll}] \,|\, E_1[\texttt{waiting}(c_1)\ on_1] \,|\, \cdots \,|\, E_m[\texttt{waiting}(c_m)\ on_m],\sigma,\texttt{CT} \\ \longrightarrow_l E[\varepsilon] \,|\, E_1[\texttt{ready}\ o\ n_1] \,|\, \cdots \,|\, E_m[\texttt{ready}\ o\ n_m],\sigma',\texttt{CT} \end{array}}$$

NotifyNone

$$\frac{\mathsf{insync}(o,E) \qquad \mathsf{blocked}(\sigma,o) = \emptyset}{E[o.\texttt{notify}],\sigma,\texttt{CT} \longrightarrow_l E[\varepsilon],\sigma,\texttt{CT}}$$

Ready

$$\frac{\mathsf{getLock}(\sigma,o) = 0 \qquad \mathsf{setLock}(\sigma,o,n) = \sigma'}{\texttt{ready}\ o\ n,\sigma,\texttt{CT} \longrightarrow_l \varepsilon,\sigma',\texttt{CT}}$$

LeaveCritical

$$\frac{\mathsf{getLock}(\sigma,o) = n \qquad \mathsf{setLock}(\sigma,o,n-1) = \sigma'}{\begin{array}{c} \texttt{insync}\ o\ \{v\},\sigma,\texttt{CT} \longrightarrow_l v,\sigma',\texttt{CT} \\ \texttt{insync}\ o\ \{\texttt{return}(c)\ v\},\sigma,\texttt{CT}, \longrightarrow_l \texttt{return}(c)\ v,\sigma',\texttt{CT} \end{array}}$$

[Method Invocation]

RC-MethLocal

$$E[o.\mathtt{m}(\vec{v})] \,|\, P, \sigma, \mathtt{CT} \longrightarrow_l (\nu\, c)(E[\mathtt{await}\ c] \,|\, o.\mathtt{m}(\vec{v})\ \mathtt{with}\ c \,|\, P, \sigma, \mathtt{CT}) \quad c \text{ fresh}, o \in \mathsf{dom_o}(\sigma)$$

RC-MethRemote

$$E[o.\mathtt{m}(\vec{v})] \,|\, P, \sigma, \mathtt{CT} \longrightarrow_l (\nu\, c)(E[\mathtt{await}\ c] \,|\, \mathsf{go}\ o.\mathtt{m}(\mathtt{serialize}(\vec{v}))\ \mathtt{with}\ c \,|\, P, \sigma, \mathtt{CT}) \quad c \text{ fresh}, o \notin \mathsf{dom_o}(\sigma)$$

RC-MethInvoke

$$\frac{\sigma(o) = (C, \dots) \qquad \mathsf{mbody}(\mathtt{m}, C, \mathtt{CT}) = (\vec{x}, e)}{o.\mathtt{m}(\vec{v})\ \mathtt{with}\ c, \sigma, \mathtt{CT} \longrightarrow_l (\nu\, \vec{x})(e[o/\mathtt{this}][\mathtt{return}(c)/\mathtt{return}], \sigma \cdot [\vec{x} \mapsto \vec{v}], \mathtt{CT})}$$

RC-Await

$$E[\mathtt{await}\ c] \,|\, \mathtt{return}(c)\ v, \sigma, \mathtt{CT} \longrightarrow_l E[v], \sigma, \mathtt{CT}$$

RN-SerReturn

$$l[\mathtt{return}(c)\ v \,|\, P, \sigma, \mathtt{CT}] \longrightarrow l[\mathsf{go}\ \mathtt{serialize}(v)\ \mathtt{to}\ c \,|\, P, \sigma, \mathtt{CT}] \quad c \notin \mathsf{fn}(P)$$

RN-Leave

$$\begin{array}{l} l_1[\mathsf{go}\ o.\mathtt{m}(\vec{v})\ \mathtt{with}\ c \,|\, P_1, \sigma_1, \mathtt{CT}_1] \,|\, l_2[P_2, \sigma_2, \mathtt{CT}_2] \\ \quad \longrightarrow l_1[P_1, \sigma_1, \mathtt{CT}_1] \,|\, l_2[o.\mathtt{m}(\mathtt{deserialize}(\vec{v}))\ \mathtt{with}\ c \,|\, P_2, \sigma_2, \mathtt{CT}_2] \end{array} \quad o \in \mathsf{dom_o}(\sigma_2)$$

RN-Return

$$\begin{array}{l} l_1[\mathsf{go}\ v\ \mathtt{to}\ c \,|\, P_1, \sigma_1, \mathtt{CT}_1] \,|\, l_2[P_2, \sigma_2, \mathtt{CT}_2] \\ \quad \longrightarrow l_1[P_1, \sigma_1, \mathtt{CT}_1] \,|\, l_2[\mathtt{return}(c)\ \mathtt{deserialize}(v) \,|\, P_2, \sigma_2, \mathtt{CT}_2] \end{array} \quad c \in \mathsf{fn}(P_2)$$

[Code mobility]

Freeze

$$\frac{\begin{array}{c} \{\vec{y}\} = \mathsf{fv}(e) \setminus \{\vec{x}\} \qquad \sigma_y = \bigcup \sigma(y_i) \\ \sigma' = \mathsf{og}(\sigma, \mathsf{fn}(e) \cup \mathsf{fn}(\sigma_y)) \cup \sigma_y \qquad \{\vec{u}\} = \mathsf{dom}(\sigma') \\ \mathtt{CT}' = \begin{cases} \mathsf{cg}(\mathtt{CT}, \mathsf{fcl}(e) \cup \mathsf{fcl}(\sigma')) & t = \mathtt{eager} \\ \mathsf{cg}(\mathtt{CT}, \vec{C}) & t = \vec{C} \\ \emptyset & t = \mathtt{lazy} \end{cases} \end{array}}{\mathtt{freeze}[t](\vec{T}\ \vec{x};\ e), \sigma, \mathtt{CT} \longrightarrow_l \lambda(\vec{T}\ \vec{x}).(\nu\, \vec{u})(l, e, \sigma', \mathtt{CT}'), \sigma, \mathtt{CT}}$$

Defrost

$$\frac{\{\vec{C}\} = \mathsf{fcl}(e) \setminus \mathsf{dom}(\mathtt{CT}') \qquad \{\vec{F}\} = \mathsf{fcl}(\sigma') \setminus \mathsf{dom}(\mathtt{CT}')}{\begin{array}{c} \mathtt{defrost}(\vec{v};\ \lambda(\vec{T}\ \vec{x}).(\nu\, \vec{u})(m, e, \sigma', \mathtt{CT}')), \sigma, \mathtt{CT} \\ \longrightarrow_l (\nu\, \vec{x}\vec{u})(\mathtt{download}\ \vec{F}\ \mathtt{from}\ m\ \mathtt{in}\ e[\vec{C^m}/\vec{C}], \sigma \cup \sigma' \cdot [\vec{x} \mapsto \vec{v}], \mathtt{CT} \cup \mathtt{CT}') \end{array}}$$

[Class Downloading]

Resolve

$$\frac{\mathtt{CT}(C_i) = \mathtt{class}\ C_i\ \mathtt{extends}\ D_i\,\{\vec{T}\,\vec{f};\,K\,\vec{M}\}}{\mathtt{resolve}\ \vec{C}\ \mathtt{from}\ l'\ \mathtt{in}\ e, \sigma, \mathtt{CT} \longrightarrow_l \mathtt{download}\ \vec{D}\ \mathtt{from}\ l'\ \mathtt{in}\ e, \sigma, \mathtt{CT}}$$

Download

$$\frac{\{\vec{D}\} = \{\vec{C}\} \setminus \mathsf{dom}(\mathtt{CT}_1) \qquad \{\vec{F}\} = \mathsf{fcl}(\mathtt{CT}_2(\vec{D})) \qquad \mathtt{CT}' = \mathtt{CT}_2(\vec{D})[\vec{F}^{l_2}/\vec{F}]}{\begin{array}{c} l_1[E[\mathtt{download}\ \vec{C}\ \mathtt{from}\ l_2\ \mathtt{in}\ e]\,|\,P, \sigma_1, \mathtt{CT}_1]\,|\,l_2[P_2, \sigma_2, \mathtt{CT}_2] \\ \longrightarrow l_1[E[\mathtt{resolve}\ \vec{D}\ \mathtt{from}\ l_2\ \mathtt{in}\ e]\,|\,P, \sigma_1, \mathtt{CT}_1 \cup \mathtt{CT}']\,|\,l_2[P_2, \sigma_2, \mathtt{CT}_2] \end{array}}$$

DNothing

$$\mathtt{download}\ \vec{C}\ \mathtt{from}\ l'\ \mathtt{in}\ e, \sigma, \mathtt{CT} \longrightarrow_l e, \sigma, \mathtt{CT} \qquad C_i \in \mathsf{dom}(\mathtt{CT})$$

[Threads]

RC-Par

$$\frac{P_1, \sigma, \mathtt{CT} \longrightarrow_l (\nu\,\vec{u})(P_1', \sigma', \mathtt{CT}')}{P_1\,|\,P_2, \sigma, \mathtt{CT} \longrightarrow_l (\nu\,\vec{u})(P_1'\,|\,P_2, \sigma', \mathtt{CT}')}\ \vec{u} \notin \mathsf{fnv}(P_2)$$

RC-Str

$$\frac{F \equiv F_0 \longrightarrow_l F_0' \equiv F'}{F \longrightarrow_l F'}$$

RC-Res

$$\frac{(\nu\,\vec{u})(P, \sigma, \mathtt{CT}) \longrightarrow_l (\nu\,\vec{u}')(P', \sigma', \mathtt{CT}')}{(\nu\,u\vec{u})(P, \sigma, \mathtt{CT}) \longrightarrow_l (\nu\,u\vec{u}')(P', \sigma', \mathtt{CT}')}$$

[Network]

RN-Conf

$$\frac{F \longrightarrow_l F'}{l[F] \longrightarrow l[F']}$$

RN-Par

$$\frac{N \longrightarrow N'}{N\,|\,N_0 \longrightarrow N'\,|\,N_0}$$

RN-Res

$$\frac{N \longrightarrow N'}{(\nu\,u)N \longrightarrow (\nu\,u)N'}$$

RN-Str

$$\frac{N \equiv N_0 \longrightarrow N_0' \equiv N'}{N \longrightarrow N'}$$

[Errors]

Err-NullFld
$\texttt{null.f}, \sigma, \texttt{CT} \longrightarrow_l \texttt{Error}, \sigma, \texttt{CT}$

Err-NullFldAss
$\texttt{null.f} := v, \sigma, \texttt{CT} \longrightarrow_l \texttt{Error}, \sigma, \texttt{CT}$

Err-NullMeth
$\texttt{null.m}(\vec{v}), \sigma, \texttt{CT} \longrightarrow_l \texttt{Error}, \sigma, \texttt{CT}$

Err-LostCall
$\texttt{go}\ o.\texttt{m}(\vec{v})\ \texttt{with}\ c, \sigma, \texttt{CT} \longrightarrow_l \texttt{Error}, \sigma, \texttt{CT}$

Err-LostReturn
$\texttt{go}\ v\ \texttt{to}\ c, \sigma, \texttt{CT} \longrightarrow_l \texttt{Error}, \sigma, \texttt{CT}$

Err-ClassNotFound

$$\frac{\exists C_i \in \vec{C}. C_i \notin \mathsf{dom}(\texttt{CT}_1) \cup \mathsf{dom}(\texttt{CT}_2)}{\begin{array}{c} l_1[E[\texttt{download}\ \vec{C}\ \texttt{from}\ l_2\ \texttt{in}\ e] \mid P, \sigma_1, \texttt{CT}_1] \mid l_2[P_2, \sigma_2, \texttt{CT}_2] \\ \longrightarrow l_1[E[\texttt{Error}] \mid P, \sigma_1, \texttt{CT}_1] \mid l_2[P_2, \sigma_2, \texttt{CT}_2] \end{array}}$$

Err-Monitor

$$\frac{\neg \mathsf{insync}(o, E)}{\begin{array}{c} E[o.\texttt{notify}], \sigma, \texttt{CT} \longrightarrow_l E[\texttt{Error}], \sigma, \texttt{CT} \\ E[o.\texttt{notifyAll}], \sigma, \texttt{CT} \longrightarrow_l E[\texttt{Error}], \sigma, \texttt{CT} \\ E[o.\texttt{wait}], \sigma, \texttt{CT} \longrightarrow_l E[\texttt{Error}], \sigma, \texttt{CT} \end{array}}$$

3 Typing System

[Types] $\boxed{\vdash S : \texttt{tp}}$

Wf-Base

$$\frac{}{\begin{array}{c} \vdash \texttt{void} : \texttt{tp} \\ \vdash \texttt{bool} : \texttt{tp} \\ \vdash \texttt{chan} : \texttt{tp} \end{array}}$$

Wf-SC

$$\frac{\vdash U : \texttt{tp} \vee U \in \mathsf{CSig}}{\begin{array}{c} \vdash \texttt{chanI}(U) : \texttt{tp} \\ \vdash \texttt{chanO}(U) : \texttt{tp} \\ \vdash \texttt{ret}(U) : \texttt{tp} \\ \vdash \texttt{thunk}\langle U \rangle : \texttt{tp} \end{array}}$$

Wf-Vec

$$\frac{\vdash U_i : \texttt{tp}}{\vdash \vec{U} : \texttt{tp}}$$

Wf-Ser

$$\frac{\vdash \vec{U} : \texttt{tp}}{\vdash \texttt{ser}\langle \vec{U} \rangle : \texttt{tp}}$$

Wf-Sig

$$\frac{\begin{array}{c} \mathsf{override}(\texttt{m}_\texttt{i}, D_i, \vec{T}_i \to U_i) \quad \vdash D : \texttt{tp} \\ \forall S \in \{\vec{T}, \vec{U}, \vec{T}_i\}.\ \vdash S : \texttt{tp} \vee S \in \mathsf{dom}(\mathsf{CSig}) \end{array}}{\vdash \texttt{extends}\ D\ [\texttt{remote}]\ \vec{T}\vec{\texttt{f}}\ \ \{\texttt{m}_\texttt{i} : \vec{T}_i \to U_i\} : \texttt{tp}}$$

Wf-Ctp

$$\frac{\vdash \mathsf{CSig}(C) : \texttt{tp}}{\vdash C : \texttt{tp}}$$

Wf-CSig

$$\frac{\begin{array}{c} \forall C \in \mathsf{dom}(\mathsf{CSig}) \\ \vdash C : \texttt{tp} \end{array}}{\vdash \mathsf{CSig} : \texttt{ok}}$$

[Subtyping] $\boxed{C <: D}$

ST-Refl
$$\frac{}{T <: T}$$

ST-Trans
$$\frac{C <: D \qquad D <: E}{C <: E}$$

ST-Vec
$$\frac{U'_i <: U_i \qquad 0 \leq i \lneq n}{\vec{U}' <: \vec{U}}$$

ST-Ser
$$\frac{\vec{U}' <: \vec{U}}{\mathtt{ser}\langle \vec{U}' \rangle <: \mathtt{ser}\langle \vec{U} \rangle}$$

ST-Expr
$$\frac{U' <: U}{\mathtt{thunk}\langle U' \rangle <: \mathtt{thunk}\langle U \rangle \qquad \mathtt{ret}(U') <: \mathtt{ret}(U)}$$

ST-Class
$$\frac{\mathsf{CSig}(C) = \mathtt{extends}\ D\ [\mathtt{remote}]\ \vec{T}\vec{\mathtt{f}}\ \ \{\mathtt{m_i} : \vec{T_i} \to U_i\}}{C <: D}$$

[Environments] $\boxed{\Gamma; \Delta \vdash \mathtt{Env}}$

E-Nil
$$\frac{}{\emptyset \vdash \mathtt{Env}}$$

E-Var
$$\frac{\vdash T : \mathtt{tp} \qquad x \notin \mathsf{dom}(\Gamma)}{\Gamma, x : T \vdash \mathtt{Env}}$$

E-Oid
$$\frac{\vdash C : \mathtt{tp} \qquad o \notin \mathsf{dom}(\Gamma)}{\Gamma, o : C \vdash \mathtt{Env}}$$

E-This
$$\frac{\vdash C : \mathtt{tp} \qquad \mathtt{this} \notin \mathsf{dom}(\Gamma)}{\Gamma, \mathtt{this} : C \vdash \mathtt{Env}}$$

E-CNil
$$\frac{}{\Gamma; \emptyset \vdash \mathtt{Env}}$$

E-Chan
$$\frac{\vdash \tau : \mathtt{tp} \qquad \Gamma; \Delta \vdash \mathtt{Env} \qquad c \notin \mathsf{dom}(\Delta)}{\Gamma; \Delta, c : \tau \vdash \mathtt{Env}}$$

[Stores] $\boxed{\Gamma; \Delta \vdash \mathtt{Env}}$

S-CNil
$$\frac{}{\Gamma \vdash \emptyset : \mathtt{ok}}$$

S-Var
$$\frac{\Gamma \vdash \sigma : \mathtt{ok} \qquad \Gamma \vdash x : T \qquad x \notin \mathsf{dom_v}(\sigma) \qquad \Gamma \vdash v : T' \qquad T' <: T}{\Gamma \vdash \sigma \cdot [x \mapsto v] : \mathtt{ok}}$$

S-Oid
$$\frac{\Gamma; \Delta \vdash \sigma : \mathtt{ok} \qquad \Gamma \vdash o : C \qquad \Gamma \vdash \vec{v} : \vec{T}' \qquad \vec{T}' <: \vec{T} \qquad o \notin \mathsf{dom_o}(\sigma) \qquad \mathsf{fields}(C) = \vec{T}\,\vec{f} \qquad n \geq 0 \qquad \Gamma; \Delta \vdash c_i : \mathtt{chan0}(\mathtt{void})}{\Gamma; \Delta \vdash \sigma \cdot [o \mapsto (C, \vec{f} : \vec{v}, n, \{\vec{c}\})] : \mathtt{ok}}$$

[Values] $\boxed{\Gamma \vdash v : U}$

TV-Bool
$$\frac{\Gamma \vdash \mathtt{Env}}{\Gamma \vdash \mathtt{true} : \mathtt{bool} \qquad \Gamma \vdash \mathtt{false} : \mathtt{bool}}$$

TV-Null
$$\frac{\vdash C : \mathtt{tp}}{\Gamma \vdash \mathtt{null} : C}$$

TV-Oid
$$\frac{\Gamma, o : C, \Gamma' \vdash \mathtt{Env}}{\Gamma, o : C, \Gamma' \vdash o : C}$$

TV-Empty
$$\frac{\Gamma \vdash \mathtt{Env}}{\Gamma \vdash \varepsilon : \mathtt{void}}$$

TV-FrozenArrow
$$\frac{\vec{x} : \vec{T}, \vec{u} : \vec{T}' \vdash e : U \qquad \vec{u} : \vec{T}' \vdash \sigma : \mathtt{ok} \qquad \vdash \mathtt{CT} : \mathtt{ok}}{\vdash \lambda(\vec{T}\ \vec{x}).(\nu\,\vec{u})(l, e, \sigma, \mathtt{CT}) : \vec{T} \to U}$$

TV-FrozenVec
$$\frac{\vec{u} : \vec{T}' \vdash \vec{v} : \vec{U} \qquad \vec{u} : \vec{T}' \vdash \sigma : \mathtt{ok} \qquad \vdash \mathtt{CT} : \mathtt{ok}}{\vdash (\nu\,\vec{u})(l, \vec{v}, \sigma, \mathtt{CT}) : \mathtt{ser}\langle \vec{U} \rangle}$$

[Expressions] $\boxed{\Gamma \vdash e : S}$

TE-Var
$$\frac{\Gamma, x : T, \Gamma' \vdash \texttt{Env}}{\Gamma, x : T, \Gamma' \vdash x : T}$$

TE-This
$$\frac{\Gamma, \texttt{this} : C, \Gamma' \vdash \texttt{Env}}{\Gamma, \texttt{this} : C, \Gamma' \vdash \texttt{this} : C}$$

TE-Cond
$$\frac{\exists S : S_1 <: S \wedge S_2 <: S \quad \Gamma \vdash e : \texttt{bool} \quad \Gamma \vdash e_1 : S_1 \quad \Gamma \vdash e_2 : S_2}{\Gamma \vdash \texttt{if}\ (e)\ \{e_1\}\ \texttt{else}\ \{e_2\} : S}$$

TE-While
$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{void}}{\Gamma \vdash \texttt{while}\ (e_1)\ \{e_2\} : \texttt{void}}$$

TE-Fld
$$\frac{\Gamma \vdash e : C \quad \vdash C : \texttt{tp} \quad e \neq \texttt{this}, o \implies \mathsf{local}(C) \quad \mathsf{fields}(C) = \vec{T}\,\vec{f}}{\Gamma \vdash e.\texttt{f}_\texttt{i} : T_i}$$

TE-Seq
$$\frac{\Gamma \vdash e_1 : U \quad \Gamma \vdash e_2 : S}{\Gamma \vdash e_1 ; e_2 : S}$$

TE-Ass
$$\frac{\Gamma \vdash e : T' \quad T' <: T \quad \Gamma \vdash x : T}{\Gamma \vdash x := e : T'}$$

TE-FldAss
$$\frac{\Gamma \vdash e.\texttt{f} : T \quad T' <: T \quad \Gamma \vdash e' : T'}{\Gamma \vdash e.\texttt{f} := e' : T'}$$

TE-New
$$\frac{\mathsf{fields}(C) = \vec{T}\,\vec{f} \quad T_i' <: T_i \quad \Gamma \vdash e_i : T_i' \quad \vdash C : \texttt{tp}}{\Gamma \vdash \texttt{new}\ C^{\text{-}}(\vec{e}) : C}$$

TE-Meth
$$\frac{\mathsf{mtype}(\texttt{m}, C) = \vec{T} \to U \quad \Gamma \vdash e_0 : C \quad \Gamma \vdash \vec{e} : \vec{T}' \quad T_i' <: T_i}{\Gamma \vdash e_0.\texttt{m}(\vec{e}) : U}$$

TE-Dec
$$\frac{\Gamma \vdash e : T \quad T <: T' \quad \Gamma, x : T \vdash e_0 : S}{\Gamma \vdash T'\ x = e ; e_0 : S}$$

TE-Return
$$\frac{\Gamma \vdash e : U}{\Gamma \vdash \texttt{return}\ e : \texttt{ret}(U)}$$

TE-ReturnVoid
$$\frac{\Gamma \vdash \texttt{Env}}{\Gamma \vdash \texttt{return} : \texttt{ret}(\texttt{void})}$$

TE-FreezeArrow
$$\frac{\Gamma, \vec{x} : \vec{T} \vdash e : U}{\Gamma \vdash \texttt{freeze}[t](\vec{T}\ \vec{x};\ e) : \vec{T} \to U}$$

TE-FreezeVec
$$\frac{\Gamma \vdash \vec{v} : \vec{U}}{\Gamma \vdash \texttt{freeze}[t](\vec{v}) : \texttt{ser}\langle \vec{U} \rangle}$$

TE-DefrostArrow
$$\frac{\Gamma \vdash \vec{e} : \vec{T}' \quad \vec{T}' <: \vec{T} \quad \Gamma \vdash e : \vec{T} \to U}{\Gamma \vdash \texttt{defrost}(\vec{e};\ e) : U}$$

TE-DefrostVec
$$\frac{\Gamma \vdash e : \texttt{ser}\langle \vec{U} \rangle}{\Gamma \vdash \texttt{defrost}(e) : \vec{U}}$$

TE-Fork
$$\frac{\Gamma \vdash e : S}{\Gamma \vdash \texttt{fork}(e) : \texttt{void}}$$

TE-Sync
$$\frac{e \neq \texttt{this}, o \implies \mathsf{local}(C) \quad \Gamma \vdash e_1 : C \quad \Gamma \vdash e_2 : S}{\Gamma \vdash \texttt{sync}\ (e_1)\ \{e_2\} : S}$$

TE-Monitor
$$\frac{e \neq \texttt{this}, o \implies \mathsf{local}(C) \quad \Gamma \vdash e : C}{\Gamma \vdash e.\texttt{wait} : \texttt{void} \quad e.\texttt{notify} : \texttt{void} \quad e.\texttt{notifyAll} : \texttt{void}}$$

TE-ClassLoad
$$\frac{\Gamma \vdash e : \vec{U} \quad \vdash \vec{C} : \texttt{tp}}{\Gamma \vdash \texttt{download}\ \vec{C}\ \texttt{from}\ l\ \texttt{in}\ e : \vec{U} \quad \Gamma \vdash \texttt{resolve}\ \vec{C}\ \texttt{from}\ l\ \texttt{in}\ e : \vec{U}}$$

TE-InSync
$$\frac{\Gamma \vdash o : C \quad \Gamma \vdash e : S}{\Gamma \vdash \texttt{insync}\ o\ \{e\} : S}$$

TE-Ready
$$\frac{\Gamma \vdash o : C \quad n > 0}{\Gamma \vdash \texttt{ready}\ o\ n : \texttt{void}}$$

TE-Pe
$$\frac{\Gamma \vdash pe : T}{\Gamma \vdash pe : \texttt{void}}$$

TE-Hole
$$\frac{\vdash U : \texttt{tp}}{\Gamma \vdash [\,]^U : U}$$

[Threads] $\boxed{\Gamma;\Delta,c:\mathtt{chan}\vdash P:\mathtt{thread}}$

TT-Nil
$$\frac{\Gamma;\emptyset\vdash\mathtt{Env}}{\Gamma;\emptyset\vdash\mathbf{0}:\mathtt{thread}}$$

TT-Par
$$\frac{\Gamma;\Delta_i\vdash P_i:\mathtt{thread}\qquad\Delta_1\asymp\Delta_2}{\Gamma;\Delta_1\odot\Delta_2\vdash P_1\,|\,P_2:\mathtt{thread}}$$

TT-Weak
$$\frac{\Gamma;\Delta\vdash P:\mathtt{thread}\qquad c\notin\mathsf{dom}(\Delta)}{\Gamma;\Delta,c:\mathtt{chan}\vdash P:\mathtt{thread}}$$

TT-Await
$$\frac{\Gamma;\Delta\vdash E[\,]^{U}:\mathtt{thread}\qquad c\notin\mathsf{dom}(\Delta)}{\Gamma;\Delta,c:\mathtt{chanI}(U)\vdash E[\mathtt{await}\ c]^{U}:\mathtt{thread}}$$

TT-Res
$$\frac{\Gamma;\Delta,c:\mathtt{chan}\vdash P:\mathtt{thread}}{\Gamma;\Delta\vdash(\nu\,c)P:\mathtt{thread}}$$

TT-Return
$$\frac{\Gamma\vdash e:\mathtt{ret}(U')\qquad U'<:U}{\Gamma;c:\mathtt{chanO}(U)\vdash e[\mathtt{return}(c)/\mathtt{return}]:\mathtt{thread}}$$

TT-Waiting
$$\frac{\Gamma;\Delta\vdash E[\,]^{\mathtt{void}}:\mathtt{thread}\qquad c\notin\mathsf{dom}(\Delta)\qquad n>0}{\Gamma;\Delta,c:\mathtt{chanI}(\mathtt{void})\vdash E[\mathtt{waiting}(c)\ on]^{\mathtt{void}}:\mathtt{thread}}$$

TT-Forked
$$\frac{\Gamma\vdash e:S}{\Gamma;\emptyset\vdash\mathtt{forked}\ e:\mathtt{thread}}$$

TT-GoSer
$$\frac{\Gamma\vdash o:C\qquad\Gamma\vdash\vec{v}:\vec{T}'\qquad\vec{T}'<:\vec{T}\qquad\mathsf{remote}(C)\qquad\mathsf{mtype}(\mathtt{m},C)=\vec{T}\rightarrow U}{\Gamma;c:\mathtt{chanO}(U)\vdash\mathsf{go}\ o.\mathtt{m}(\mathtt{serialize}(\vec{v}))\ \mathtt{with}\ c:\mathtt{thread}}$$

TT-MethWith
$$\frac{\Gamma\vdash o:C\qquad\Gamma\vdash v_i:T_i'\qquad T_i'<:T_i\qquad\mathsf{mtype}(\mathtt{m},C)=\vec{T}\rightarrow U}{\Gamma;c:\mathtt{chanO}(U)\vdash o.\mathtt{m}(\vec{v})\ \mathtt{with}\ c:\mathtt{thread}}$$

TT-DeserWith
$$\frac{\begin{array}{c}\Gamma\vdash o:C\\ \Gamma\vdash\lambda\vec{o}.(\vec{v},\sigma,l):\mathsf{ser}\langle\vec{T}'\rangle\qquad\vec{T}'<:\vec{T}\qquad\mathsf{remote}(C)\qquad\mathsf{mtype}(\mathtt{m},C)=\vec{T}\rightarrow U\end{array}}{\begin{array}{c}\Gamma;c:\mathtt{chanO}(U)\vdash o.\mathtt{m}(\mathtt{deserialize}(\lambda\vec{o}.(\vec{v},\sigma,l)))\ \mathtt{with}\ c:\mathtt{thread}\\ \mathsf{go}\ o.\mathtt{m}(\lambda\vec{o}.(\vec{v},\sigma,l))\ \mathtt{with}\ c:\mathtt{thread}\end{array}}$$

TT-ValTo
$$\frac{\Gamma\vdash v:U'\qquad U'<:U\qquad\neg\mathsf{local}(U')}{\begin{array}{c}\Gamma;c:\mathtt{chanO}(U)\vdash\mathsf{go}\ \mathtt{serialize}(v)\ \mathtt{to}\ c:\mathtt{thread}\\ \mathsf{go}\ v\ \mathtt{to}\ c:\mathtt{thread}\end{array}}$$

TT-GoTo
$$\frac{\Gamma\vdash e:\mathsf{ser}\langle C'\rangle\qquad C'<:C}{\Gamma;c:\mathtt{chanO}(C)\vdash\mathsf{go}\ e\ \mathtt{to}\ c:\mathtt{thread}}$$

[Configuration] $\boxed{\Gamma;\Delta,c:\mathtt{chan}\vdash F:\mathtt{conf}}$

TC-ResC
$$\frac{\Gamma;\Delta,c:\mathtt{chan}\vdash F:\mathtt{conf}}{\Gamma;\Delta\vdash(\nu\,c)F:\mathtt{conf}}$$

TC-ResId
$$\frac{\Gamma,u:T;\Delta\vdash F:\mathtt{conf}\qquad u\in\mathsf{dom}(F)}{\Gamma;\Delta\vdash(\nu\,u)F:\mathtt{conf}}$$

TC-Conf
$$\frac{\begin{array}{c}\Gamma;\Delta_1\vdash P:\mathtt{thread}\qquad\Gamma;\Delta_2\vdash\sigma:\mathtt{ok}\\ \vdash\mathtt{CT}:\mathtt{ok}\qquad\mathtt{FCT}\subseteq\mathtt{CT}\qquad\Delta_1\asymp\Delta_2\end{array}}{\Gamma;\Delta_1\odot\Delta_2\vdash P,\sigma,\mathtt{CT}:\mathtt{conf}}$$

[Network] $\boxed{\Gamma;\Delta, c : \mathtt{chan} \vdash N : \mathtt{net}}$

$$\textbf{TN-Nil}\quad \frac{\Gamma;\emptyset \vdash \mathtt{Env}}{\Gamma;\emptyset \vdash \mathbf{0} : \mathtt{net}} \qquad \textbf{TN-Conf}\quad \frac{\Gamma;\Delta \vdash F : \mathtt{conf}}{\Gamma;\Delta \vdash l[F] : \mathtt{net}} \qquad \textbf{TN-Par}\quad \frac{\Gamma;\Delta_i \vdash N_i : \mathtt{net} \quad \Delta_1 \asymp \Delta_2 \quad \mathsf{dom}(N_1) \cap \mathsf{dom}(N_2) = \emptyset \quad \mathsf{loc}(N_1) \cap \mathsf{loc}(N_2) = \emptyset}{\Gamma;\Delta_1 \odot \Delta_2 \vdash N_1 \,|\, N_2 : \mathtt{net}}$$

$$\textbf{TN-Weak}\quad \frac{\Gamma;\Delta \vdash N : \mathtt{net} \quad c \notin \mathsf{dom}(\Delta)}{\Gamma;\Delta, c : \mathtt{chan} \vdash N : \mathtt{net}} \qquad \textbf{TN-ResId}\quad \frac{\Gamma, u : T;\Delta \vdash N : \mathtt{net} \quad u \in \mathsf{dom}(N)}{\Gamma;\Delta \vdash (\nu u)N : \mathtt{net}} \qquad \textbf{TN-ResC}\quad \frac{\Gamma;\Delta, c : \mathtt{chan} \vdash N : \mathtt{net}}{\Gamma;\Delta \vdash (\nu c)N : \mathtt{net}}$$

[Method] $\boxed{\Gamma \vdash M : \mathtt{ok\ in}\ C}$ **[Class]** $\boxed{\vdash L : \mathtt{ok}}$

$$\textbf{M-ok}\quad \frac{\mathsf{mtype}(\mathtt{m}, C) = \vec{T} \to U \quad U' <: U \quad \mathtt{this} : C, \vec{x} : \vec{T} \vdash e : \mathtt{ret}(U')}{\mathtt{this} : C \vdash U\,\mathtt{m}\,(\vec{T}\vec{x})\{e\} : \mathtt{ok\ in}\ C}$$

$$\textbf{C-ok}\quad \frac{\mathsf{fields}(D) = \vec{T}'\,\vec{f}' \quad \mathsf{fields}(C) = \vec{T}\,\vec{f} \quad K = C\,(\vec{T}'\,\vec{f}', \vec{T}\,\vec{f})\{\mathtt{super}(\vec{f}'); \mathtt{this}.\vec{\mathtt{f}} := \vec{f}\} \quad \mathtt{this} : C \vdash \vec{M} : \mathtt{ok\ in}\ C}{\vdash \mathtt{class}\ C\ \mathtt{extends}\ D\{\vec{T}\,\vec{f}; K\,\vec{M}\} : \mathtt{ok}}$$

[Class Table] $\boxed{\vdash \mathtt{CT} : \mathtt{ok}}$

$$\textbf{CT-Nil}\quad \frac{}{\vdash \emptyset : \mathtt{ok}} \qquad \textbf{CT}\quad \frac{\vdash \mathtt{class}\ C\ \mathtt{extends}\ D\{\vec{T}\,\vec{f}; K\,\vec{M}\} : \mathtt{ok} \quad \vdash \mathtt{CT} : \mathtt{ok}}{\vdash \mathtt{CT} \cdot [C \mapsto \mathtt{class}\ C\ \mathtt{extends}\ D\{\vec{T}\,\vec{f}; K\,\vec{M}\}] : \mathtt{ok}}$$