1 Syntax

$\mathcal{P}$	$::=$	$\overline{\mathcal{S}}\ \overline{\mathcal{F}}$	*Program*
$\mathcal{F}$	$::=$	$\texttt{fn}\ f\langle\overline{\mathrm{k}:\kappa}\rangle\langle y:\varepsilon\rangle(\overline{e\ x:\delta}) \xrightarrow{e} \delta,\ A\ \{\ t\ \}\ \text{where}\ \gamma$	*Function Definition*
$\mathcal{S}$	$::=$	$\texttt{struct}\ s\langle\overline{\mathrm{k}:\kappa}\rangle\{\ \overline{x:\delta}\ \}$	*Struct Declaration*

t	$::=$		*Term:*
		p	*place expression*
		$\ominus t$	*unary operation*
		$t\ \oplus\ t$	*binary operation*
		$\texttt{let}\ x{:}\delta = t$	*definition*
		$\texttt{let}(e)\ x:\delta$	*declaration*
		$p = t$	*assigment*
		$\&r\ \omega\ p$	*(unique) borrow*
		$[e]\{\ t\ \}^{\overline{r}}$	*block*
		$t\ ;\ t$	*sequence*
		$(\overline{t})$	*tuple*
		$s\langle\overline{\tau}\rangle\{\ \overline{x:t}\ \}$	*struct*
		$f{::}\langle\overline{\tau}\rangle{::}\langle e\rangle(\overline{p})$	*function call*
		$f{::}\langle\overline{\tau}\rangle{::}\lll d, d\ ;\ \overline{r},\ \overline{\delta}\ggg(\overline{t})$	*kernel call*
		$\texttt{sched}(e_L.d)\ y\ \texttt{in}\ e\ \{\ t\ \}$	*schedule*
		$\texttt{split}(e_L.d)\ e\ \texttt{at}\ \eta\ \{\ y \Rightarrow t,\ y \Rightarrow t\ \}$	*split exec*
		$\texttt{sync}(e)$	*barrier*
t	$::=$		*Term:*
		$\texttt{for}\ x\ \texttt{in}\ \eta..\eta\ \{\ t\ \}$	*for-loop*
		$\texttt{while}\ t\ \{\ t\ \}$	*while-loop*
		$\texttt{if}\ t\ \{\ t\ \}\ \texttt{else}\ \{\ t\ \}$	*if-else*
		$\texttt{unsafe}\ t$	*unsafe*
p	$::=$		*Place Expressions:*
		x	*variable*
		$p.\mathrm{j} \mid p.x$	*projections*
		$*p$	*dereference*
		$p[t]$	*index*
		$p[\![e]\!]$	*select*
		$p.v$	*view*
v	$::=$	$f_v{::}\langle\overline{\tau}\rangle(\overline{v})$	*view expression*
f_v	$::=$		
		$\texttt{to\_view} \mid \texttt{grp}$	
		$[\eta..\eta] \mid \texttt{reverse}$	
		$\texttt{transpose} \mid \texttt{map}$	

Figure 1: *Descend* programs, terms and place expressions

κ	$::=$	$\texttt{dty} \mid \texttt{rgn} \mid \texttt{mem} \mid \texttt{nat}$	*Kinds*
τ	$::=$	$\delta \mid \varrho \mid \mu \mid \eta$	*Type-level terms*
k	$::=$	$\alpha \mid \imath \mid m \mid n$	*Type-level identifier*
		$\langle\overline{x:k}\rangle(\delta_1,\ldots,\delta_n) \xrightarrow{y:\varepsilon} \delta\ \texttt{where}\ \gamma$	*Function type*
δ	$::=$		*Data Types:*
		α	*type variable*
		$\texttt{bool} \mid \texttt{int} \mid \texttt{float} \mid \texttt{AtomicU32}$	*base types*
		$(\overline{\delta})$	*tuple type*
		$s\{\ \overline{x:\delta}\ \}$	*struct type*
		$[\delta;\eta] \mid [\![\delta;\eta]\!]$	*array (view) type*
		$\&\rho\ \omega\ \mu\ \delta$	*reference type*
		$\delta\ @\ \mu$	*boxed type*
ω	$::=$	$\texttt{shrd} \mid \texttt{uniq}$	*Borrowing Mode*
μ	$::=$		*Memory:*
		$\texttt{cpu.mem}$	
		$\texttt{gpu.global}$	
		$\texttt{gpu.shared}$	
		m	
ρ	$::=$		*Lifetimes:*
		$\imath$	*abstract lifetime*
		r	*concrete lifetime*
η	$::=$		*Natural Numbers:*
		$\eta \oplus \eta \mid n \mid 0 \mid 1 \mid\ \ldots$	
γ	$::=$	$\texttt{true} \mid \eta = \eta \mid \eta < \eta$	*Nat Constraints:*
		$\gamma\ \texttt{and}\ \gamma \mid \gamma\ \texttt{or}\ \gamma$	

Figure 2: Formal syntax of kinds and types in *Descend*.

Δ	$::=$	$\bullet \mid \Delta, \alpha:\texttt{dty} \mid \Delta, \imath:\texttt{rgn} \mid \Delta, m:\texttt{mem} \mid \Delta, n:\texttt{nat}$	*Kinding Environemnt*
Γ	$::=$	$\bullet \mid \Gamma, (\mathcal{F})$	*Stack Environment*
$\mathcal{F}$	$::=$	$\bullet \mid \mathcal{F}, x:e\ \tilde{\delta} \mid \mathcal{F}, \_ : e\ \delta \mid \mathcal{F}, r \mapsto \{\ \overline{\ell}\ \}$	*Stack Frame*
$\tilde{\delta}$	$::=$	$\delta \mid \lfloor\delta\rfloor \mid (\tilde{\delta},\ldots,\tilde{\delta})$	*Partial Types*
ℓ	$::=$	${}^{\omega}p$	*Loans*
A	$::=$	$\bullet \mid \mathrm{A}, \ell$	*Access Environment*

Figure 3: Environments and other syntax relevant for typing

$$
\begin{array}{lll}
e & ::= & \texttt{grid}\langle \mathscr{d}, \mathscr{d}\rangle.\texttt{blocks}[e_{\mathsf{R}}]\ldots[e_{\mathsf{R}}].\texttt{threads}[e_{\mathsf{R}}]\ldots[e_{\mathsf{R}}] \\
 & & \texttt{grid}\langle \mathscr{d}, \mathscr{d}\rangle.\texttt{blocks}[e_{\mathsf{R}}]\ldots[e_{\mathsf{R}}].\texttt{warps}[e_{\mathsf{R}}].\texttt{lanes}[e_{\mathsf{R}}] \\
 & & y \mid e[..\eta]_{e_L.d} \mid e[\eta..]_{e_L.d} \\
e_L & ::= & \texttt{blocks} \mid \texttt{warps} \mid \texttt{threads} \mid \texttt{lanes} \\
e_{\mathsf{R}} & ::= & n \mid \eta..\eta
\end{array}
$$

$$
\begin{array}{llll}
c & ::= & \texttt{cond} \mid \texttt{all} & \textit{Conditional Select} \\
\mathscr{d} & ::= & & \textit{Dimensions:} \\
 & & \texttt{xyz}\langle\eta,\eta,\eta\rangle & \textit{3-dim} \\
 & & \texttt{xy}\langle\eta,\eta\rangle \mid \texttt{xz}\langle\eta,\eta\rangle \mid \texttt{yz}\langle\eta,\eta\rangle & \textit{2-dim} \\
 & & \texttt{x}\langle\eta\rangle \mid \texttt{y}\langle\eta\rangle \mid \texttt{z}\langle\eta\rangle & \textit{1-dim} \\
d & ::= & \texttt{x} \mid \texttt{y} \mid \texttt{z} & \textit{Dim-selector} \\
\varepsilon & ::= & & \textit{Exec Types:} \\
 & & \texttt{gpu.grid}\langle \mathscr{d}, \mathscr{d}\rangle.\texttt{blocks}[n]_0..[n]_{d-1}.\texttt{threads}[n]_0..[n]_{d-1} & \\
 & & \texttt{gpu.grid}\langle \mathscr{d}, \mathscr{d}\rangle.\texttt{blocks}[n]_0..[n]_{d-1}.\texttt{warps}[n].\texttt{lanes}[n] & \\
 & & \texttt{cpu.Thread} & \\
 & & \texttt{gpu.Grid}\ \mathscr{d}\ \mathscr{d} & \\
 & & \texttt{gpu.Block}\ \mathscr{d} & \\
 & & \texttt{gpu.Thread} & \\
 & & \texttt{gpu.Warp} & \\
 & & \texttt{gpu.BlockGrp}\ \mathscr{d}\ \mathscr{d} & \\
 & & \texttt{gpu.ThreadGrp}\ \mathscr{d} & \\
 & & \texttt{gpu.WarpGrp}\ \eta & \\
 & & \texttt{gpu.GlobalThreads}\ \mathscr{d}\ \varepsilon & \\
 & & \texttt{Any} &
\end{array}
$$

Figure 4: Execution Resources

2 Well-formedness Judgements

$\boxed{\vdash \mathscr{P}}$: Program Typing

$$
\frac{\forall \mathcal{F} \in \mathscr{P}.\ \mathscr{P} \vdash \mathcal{F}}{\vdash \mathscr{P}}
$$

$\boxed{\mathscr{P} \vdash \mathcal{F}}$: Check types and well-formedness of function definitions.

$$
\frac{\begin{array}{c}
\Delta \vdash \varepsilon \qquad \dfrac{\dfrac{\Delta = \overline{k : \kappa}}{\overline{y : \varepsilon \vdash e' : \varepsilon'}}\quad y : \varepsilon \vdash e : \varepsilon''}{\overline{\Delta;\ \bullet \vdash \delta : \texttt{dty}}} \\
\Delta\,;\, \bullet,\ (\overline{e'\ x : \delta})\ \mid\ y : \varepsilon\,;\, e\ \mid\ \bullet\ \vdash \boxed{t\ : \delta'}\ \dashv\ \Gamma'\ \mid\ \mathrm{A} \\
\Delta\,;\, \bullet\ \vdash^{\bullet}\ \delta' \rightsquigarrow \delta \dashv \bullet \\
\mathscr{P};\ \Delta;\ \Gamma \mid y : \varepsilon\ ; e \vdash \mathrm{A}
\end{array}}{\mathscr{P} \vdash \texttt{fn}\ f\langle\overline{k : \kappa}\rangle\langle y : \varepsilon\rangle(\overline{e'\ x : \delta}) \xrightarrow{e} \delta,\ \mathrm{A}\ \{\ t\ \}\ \text{where}\ \gamma}
$$

$\boxed{\Delta;\ y : \varepsilon;\ e \vdash \Gamma}$: Check that the Stack Environment is well-formed.

$$
\frac{}{\Delta;\ y : \varepsilon;\ e \vdash \bullet}
\qquad
\frac{\begin{array}{c}
\Delta;\ y : \varepsilon;\ e \vdash \Gamma \\
\forall x : e'\ \delta \in \mathcal{F}.\ \Delta;\ \Gamma, (\mathcal{F}) \vdash \delta : \texttt{dty} \\
\forall x : e'\ \delta \in \mathcal{F}.\ y : \varepsilon \vdash e' : \varepsilon' \\
\forall r \mapsto \{\ \overline{\ell}\ \} \in \mathcal{F}.\ \forall^{\omega} p.\ \Delta\,;\, \Gamma, (\mathcal{F})\ \mid\ y : \varepsilon\,;\, e\ \vdash^{\omega}_{\texttt{pl}} p\ : \delta'
\end{array}}{\Delta;\ y : \varepsilon;\ e \vdash \Gamma, (\mathcal{F})}
$$

$\boxed{\Gamma_g;\ \Delta;\ \Gamma \mid y : \varepsilon\ ; e \vdash A}$: Check that the Access Environment is well-formed.

$$
\frac{\forall^{\omega} p \in \mathrm{A}.\ \Delta\,;\, \Gamma\ \mid\ y : \varepsilon\,;\, e\ \vdash^{\omega}_{\texttt{pl}} p\ : \delta'}{\Gamma_g;\ \Delta;\ \Gamma \mid y : \varepsilon\ ; e \vdash \mathrm{A}}
$$

$\boxed{\Delta \vdash^{n} e_{\mathsf{R}}}$

$$\textsc{WF-Range}\quad \frac{n \le s \qquad 0 < n - m}{\Delta \vdash^{s} m..n} \qquad\qquad \textsc{WF-RangeVar}\quad \frac{\Delta(x) = m..n \qquad n \le s \qquad 0 < n - m}{\Delta \vdash^{s} x}$$

$\boxed{\Delta \vdash e : \varepsilon}$: Well-formedness of execution resource types.

$$\textsc{T-CpuThread}\quad \frac{}{\Delta \vdash \texttt{cpu.thread} : \texttt{cpu.Thread}}$$

$$\textsc{T-Base}\quad \frac{\forall 0 \le i < d.\ \Delta \vdash^{\mathscr{d}_{g\,i}} e_{\mathsf{R}i} \land \Delta \vdash^{\mathscr{d}_{b\,i}} e_{\mathsf{R}i}}{\begin{array}{c}\Delta \vdash \texttt{gpu.grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[e_{\mathsf{R}\texttt{blocks}0}]_0 \ldots [e_{\mathsf{R}\texttt{blocks}d-1}]_{d-1}.\texttt{threads}[e_{\mathsf{R}\texttt{threads}0}] \ldots [e_{\mathsf{R}\texttt{threads}d-1}] : \\ \texttt{gpu.grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[m_0]_0..[m_{d-1}]_{d-1}.\texttt{threads}[n_0]_0..[n_{d-1}]_{d-1}\end{array}}$$

$$\textsc{T-BaseWarps}\quad \frac{\begin{array}{c}\forall 0 \le i < d.\ m_{2i} \le \mathscr{d}_{g_i} \land m_i = m_{2i} - m_{1i} \land m_i > 0 \qquad o_2 \le \mathsf{WarpSize} \qquad o = o_2 - o_1 \qquad o > 0 \\ n_b = \sum_{i=0}^{d-1} \mathscr{d}_{bi} \qquad n_w = \dfrac{n_b}{\mathsf{WarpSize}} \qquad n_2 \le n_w \qquad n_b \bmod \mathsf{WarpSize} = 0 \qquad n = n_2 - n_1 \qquad n > 0\end{array}}{\begin{array}{c}\Delta \vdash \texttt{gpu.grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[m_{1\,0}..m_{2\,0}] \ldots [m_{1\,d-1}..m_{2\,d-1}].\texttt{warps}[n_1..n_2].\texttt{lanes}[o_1..o_2] : \\ \texttt{gpu.Grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[m_0]_0..[m_{d-1}]_{d-1}.\texttt{warps}[n].\texttt{lanes}[o]\end{array}}$$

$$\textsc{T-SelectRange}\quad \frac{\begin{array}{c}\Delta \vdash^{n_k} m_1..m_2 \\ \Delta \vdash e : \texttt{gpu.Grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.e_{L0}[n_{0,0}]..[n_{0,d-1}] \ldots e_L[n_0]..[n_k]..[n_{d-1}] \ldots e_{Lo}[n_{n,0}]..[n_{n,d-1}]\end{array}}{\Delta \vdash e[m_1..m_2]_{e_L.k} : \texttt{gpu.grid}\langle \mathscr{d}, \mathscr{d} \rangle.e_{L0}[n_{0,0}]..[n_{0,d-1}] \ldots e_L[n_0]..[m_2 - m_1]..[n_{d-1}] \ldots e_{Lo}[n_{n,0}]..[n_{n,d-1}]}$$

$$\textsc{T-ToWarps}\quad \frac{\begin{array}{c}n_b = \sum_{i=0}^{d-1} \mathscr{d}_{bi} \qquad n_w = \dfrac{n_b}{\mathsf{WarpSize}} \qquad n_b \bmod \mathsf{WarpSize} = 0 \\ \Delta \vdash e : \texttt{gpu.Grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[m_0]..[m_{d-1}]\end{array}}{\Delta \vdash e.\texttt{to\_warps} : \texttt{gpu.Grid}\langle \mathscr{d}_g, \mathscr{d}_b \rangle.\texttt{blocks}[m_0]..[m_{d-1}].\texttt{warps}[n_w]}$$

$\boxed{\Delta;\ \Gamma \vdash \tau : \kappa}$

$\boxed{\Delta;\ \Gamma \vdash \rho : \texttt{rgn}}$

$$\textsc{WF-ConcreteRegion}\quad \frac{r \in \mathrm{dom}(\Gamma)}{\Delta;\ \Gamma \vdash r : \texttt{rgn}} \qquad\qquad \textsc{WF-AbstractRegion}\quad \frac{\Delta(\imath) = \texttt{rgn}}{\Delta;\ \Gamma \vdash \imath : \texttt{rgn}}$$

$\boxed{\Delta;\ \Gamma \vdash \mu : \texttt{mem}}$

$$\textsc{WF-ConcreteMem}\quad \frac{\mu \in \{\ \texttt{cpu.mem}, \texttt{gpu.global}, \texttt{gpu.shared}\ \}}{\Delta;\ \Gamma \vdash \mu : \texttt{mem}} \qquad\qquad \textsc{WF-AbstractMem}\quad \frac{\Delta(m) = \texttt{mem}}{\Delta;\ \Gamma \vdash m : \texttt{mem}}$$

$\boxed{\Delta;\ \Gamma \vdash \delta : \texttt{dty}}$

$$\textsc{WF-BaseType}\quad \frac{}{\Delta;\ \Gamma \vdash \texttt{int} : \texttt{dty}} \qquad \textsc{WF-TVar}\quad \frac{\Delta(\alpha) = \texttt{dty}}{\Delta;\ \Gamma \vdash \alpha : \texttt{dty}}$$

$$\textsc{WF-Ref}\quad \frac{\Delta;\ \Gamma \vdash \rho : \texttt{rgn} \qquad \Delta;\ \Gamma \vdash \mu : \texttt{mem} \qquad \Delta;\ \Gamma \vdash \delta : \texttt{dty}}{\Delta;\ \Gamma \vdash \&\rho\ \omega\ \mu\ \delta : \texttt{dty}} \qquad \textsc{WF-Array}\quad \frac{\Delta;\ \Gamma \vdash \delta : \texttt{dty}}{\Delta;\ \Gamma \vdash [\delta;\ \eta] : \texttt{dty}}$$

$$\textsc{WF-View}\quad \frac{\Delta;\ \Gamma \vdash \delta : \texttt{dty}}{\Delta;\ \Gamma \vdash [\![\delta;\ \eta]\!] : \texttt{dty}} \qquad \textsc{WF-At}\quad \frac{\Delta;\ \Gamma \vdash \delta : \texttt{dty} \qquad \Delta;\ \Gamma \vdash \mu : \texttt{mem}}{\Delta;\ \Gamma \vdash \delta @ \mu : \texttt{dty}} \qquad \textsc{WF-Tuple}\quad \frac{\forall \delta \in \overline{\delta}.\ \Delta;\ \Gamma \vdash \delta : \texttt{dty}}{\Delta;\ \Gamma \vdash (\overline{\delta}) : \texttt{dty}}$$

3 Typing Rules

$$\boxed{\Delta ; \Gamma \mid e ; \mathrm{A} \mid t \vdash \boxed{\delta : \Gamma'} \dashv \mathrm{A}' \mid}$$

T-Write-Local

$$\frac{\begin{array}{c}\mathrm{isPlace}(p)\\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{t : \delta_t} \dashv \Gamma' \mid \mathrm{A}' \\ \Gamma'(p) = (\delta_p, e_p) \qquad e = e_p \\ \Delta ; \Gamma' \vdash^{=} \delta_t \rightsquigarrow \delta_p \dashv \Gamma' \\ \Gamma' \vdash^{\texttt{uniq}} \mathbf{borrow}\ p \Rightarrow \{ {}^{\texttt{uniq}}p \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{p = t : \texttt{unit}} \dashv \Gamma'[p \mapsto \delta_t] \mid \mathrm{A}'}$$

T-Write-SharedMem

$$\frac{\begin{array}{c}\neg\mathrm{isPlace}(p) \qquad y : \varepsilon \vdash e : \texttt{GpuThread}\\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{t : \delta_t} \dashv \Gamma' \mid \mathrm{A}' \\ \Delta ; \Gamma' \mid y : \varepsilon ; e \vdash^{\texttt{uniq}}_{\texttt{pl}} p : \delta_p \\ \Delta ; \Gamma' \vdash^{+} \delta_t \rightsquigarrow \delta_p \dashv \Gamma'' \\ \Delta ; \Gamma' \mid y\varepsilon ; e \mid \mathrm{A}' \vdash^{\texttt{uniq}} \mathbf{borrow}^{+}\ p \Rightarrow \{ \overline{{}^{\texttt{uniq}}p'} \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{p = t : \texttt{unit}} \dashv \Gamma'' \mid \mathrm{A}', \overline{\ell}}$$

T-Read-By-Copy

$$\frac{\begin{array}{c}\mathrm{isPlace}(p) \qquad \mathrm{isCopyable}(\delta)\\ \Delta ; \Gamma \mid y : \varepsilon ; e \vdash^{\texttt{shrd}}_{\texttt{pl}} p : \delta \\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash^{\texttt{shrd}} \mathbf{borrow}^{+}\ p \Rightarrow \{ \overline{{}^{\omega}p'} \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{p : \delta} \dashv \Gamma \mid \mathrm{A}}$$

T-Read-By-Move

$$\frac{\begin{array}{c}\mathrm{isPlace}(p) \qquad \neg\mathrm{isCopyable}(\delta)\\ \Delta ; \Gamma \mid y : \varepsilon ; e \vdash^{\texttt{uniq}}_{\texttt{pl}} p : \delta \\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash^{\texttt{uniq}} \mathbf{borrow}^{+}\ p \Rightarrow \{ {}^{\texttt{uniq}}p \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{p : \delta} \dashv \Gamma[p \mapsto \delta^{\dagger}] \mid \mathrm{A}}$$

T-Read-SharedMem

$$\frac{\begin{array}{c}\neg\mathrm{isPlace}(p) \qquad \mathrm{isCopyable}(\delta)\\ \Delta ; \Gamma \mid y : \varepsilon ; e \vdash^{\texttt{shrd}}_{\texttt{pl}} p : \delta \\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash^{\texttt{shrd}} \mathbf{borrow}^{+}\ p \Rightarrow \{ \overline{{}^{\omega}p'} \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{p : \delta} \dashv \Gamma \mid \mathrm{A}, \overline{\ell}}$$

T-Borrow

$$\frac{\begin{array}{c}\neg\mathrm{isPlace}(p) \qquad \Gamma(r) = \emptyset\\ \Delta ; \Gamma \mid y : \varepsilon ; e \vdash^{\omega}_{\texttt{pl}} p : \delta,\ \mu \\ \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash^{\omega} \mathbf{borrow}^{+}\ p \Rightarrow \{ \overline{{}^{\omega}p'} \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\&r\ \omega\ p : \&r\ \omega\ \mu\ \delta} \dashv \Gamma \mid \mathrm{A}}$$

T-Let

$$\frac{\begin{array}{c}\Delta ; \Gamma' \vdash^{+} \delta' \rightsquigarrow \delta \dashv \Gamma'', (\mathcal{F}) \qquad \forall r \in \text{free-regions}(\delta).\ \Gamma'', (\mathcal{F}) \vdash r\ \texttt{rnrb}\\ \Delta ;\ \Gamma \vdash \delta : \texttt{dty} \qquad \Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{t : \delta'} \dashv \Gamma' \mid \mathrm{A}'\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\texttt{let}\ x : \delta = t : \texttt{unit}} \dashv \Gamma'', (\mathcal{F},\ x : e\ \delta) \mid \mathrm{A}'}$$

T-Sync

$$\frac{\begin{array}{c}y : \varepsilon \vdash e' : \varepsilon'\\ \varepsilon' \in \{ \texttt{gpu.Block}\ \mathscr{d},\ \texttt{gpu.Warp} \} \\ \mathrm{A}' = \{ {}^{\omega}\text{release-ownership}(e', p) \mid {}^{\omega}p \in \mathrm{A} \}\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\texttt{sync}(e') : \texttt{unit}} \dashv \Gamma \mid \mathrm{A}'}$$

T-Sched

$$\frac{y : \varepsilon \vdash e[n]_{e_L.d} : \varepsilon' \qquad \Delta, n : [\eta_1..\eta_2] ; \Gamma \mid y : \varepsilon ; e[n]_{e_L.d} \mid \mathrm{A} \vdash \boxed{\{ t[y := e[n]_{e_L.d}] \} : \texttt{unit}} \dashv \Gamma' \mid \mathrm{A}'}{\Delta ; \Gamma \mid y :\ \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\texttt{sched}(e_L.d)\ y\ \texttt{in}\ e\ \{ t \} : \texttt{unit}} \dashv \Gamma' \mid \mathrm{A}'}$$

T-Split-Exec

$$\frac{\begin{array}{c}y : \varepsilon \vdash e[..\eta]_{e_L.d} : \varepsilon_1 \qquad y : \varepsilon \vdash e[\eta..]_{e_L.d} : \varepsilon_2\\ \Delta ; \Gamma \mid y : \varepsilon ; e[..\eta]_{e_L.d} \mid \mathrm{A} \vdash \boxed{\{ t_1[y_1 := e[..\eta]_{e_L.d}] \} : \texttt{unit}} \dashv \Gamma_1 \mid \mathrm{A}_1 \\ \Delta ; \Gamma_1 \mid y : \varepsilon ; e[\eta..]_{e_L.d} \mid \mathrm{A}_1 \vdash \boxed{\{ t_2[y_2 := e[\eta..]_{e_L.d}] \} : \texttt{unit}} \dashv \Gamma_2 \mid \mathrm{A}_2\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\texttt{split}(e_L.d)\ e\ \texttt{at}\ \eta\ \{ y_1 \Rightarrow t_1,\ y_2 \Rightarrow t_2 \} : \texttt{unit}} \dashv \Gamma_2 \mid \mathrm{A}_2}$$

T-While

$$\frac{\begin{array}{c}\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{t_c : \texttt{bool}} \dashv \Gamma' \mid \mathrm{A}' \qquad \Delta ; \Gamma' \mid y : \varepsilon ; e.\texttt{cond} \mid \mathrm{A}' \vdash \boxed{t : \texttt{unit}} \dashv \Gamma'' \mid \mathrm{A}''\\ \Delta ; \Gamma'' \mid y : \varepsilon ; e \mid \mathrm{A}'' \vdash \boxed{t_c : \texttt{bool}} \dashv \Gamma'' \mid \mathrm{A}'' \qquad \Delta ; \Gamma'' \mid y : \varepsilon ; e.\texttt{cond} \mid \mathrm{A}'' \vdash \boxed{t : \texttt{unit}} \dashv \Gamma'' \mid \mathrm{A}''\end{array}}{\Delta ; \Gamma \mid y : \varepsilon ; e \mid \mathrm{A} \vdash \boxed{\texttt{while}\ t_c\ \{ t \} : \texttt{unit}} \dashv \Gamma'' \mid \mathrm{A}''}$$

Figure 5: Memory operations (Read, Write, Borrow)

T-FunctionCall

$$\frac{\begin{array}{c}\mathscr{P}(f) = \texttt{fn}\ f\langle\overline{\mathrm{k}:\kappa}\rangle\langle y' : \varepsilon'\rangle(\overline{e_2\ x : \delta'}) \xrightarrow{e_1} \delta_r',\ \mathrm{A}\ \{\ t\ \}\ \text{where}\ \gamma \\ \overline{\Delta;\ \Gamma \vdash \tau : \kappa} \qquad \overline{\delta = \delta'\overline{[\mathrm{k} := \tau]}} \qquad \delta_r = \delta_r'\overline{[\mathrm{k} := \tau]} \\ y : \varepsilon \vdash e_f : \varepsilon' \qquad e = e_1[y' := e_f] \\ \forall i \in \{\ 1 \ldots n\ \}.\ \Delta\,;\Gamma_{i-1} \ \mid\ y : \varepsilon\,;\, e_{2i}[y' := e_f] \ \mid\ \mathrm{A}_{i-1} \ \vdash \boxed{p_i\ \ : \delta_i} \ \dashv\ \Gamma_i \ \mid\ \mathrm{A}_i\end{array}}{\Delta\,;\Gamma_0 \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_0 \ \vdash \boxed{f{::}\langle\overline{\tau}\rangle{::}\langle e_f\rangle(\overline{p})\ \ : \delta_r} \ \dashv\ \Gamma_n \ \mid\ \mathrm{A}_n \Cup \mathrm{A}\overline{[\mathrm{k} := \tau]}[y' := e_f]\overline{[x := p]}}$$

T-KernelCall

$$\frac{\begin{array}{c} y : \varepsilon \vdash e : \texttt{cpu.Thread} \\ e_g = \texttt{gpu.grid}\langle \mathscr{d}_g, \mathscr{d}_b\rangle \qquad e_s = e_g.\texttt{blocks}[n_1]\ldots[n_k] \\ \mathscr{P}(f) = \texttt{fn}\ f\langle\overline{\mathrm{k}:\kappa}\rangle\langle y' : \varepsilon'\rangle(\overline{e_g\ x : \delta'}, \overline{e_s\ x_s : \&\mathscr{r}\ \texttt{uniq}\ \texttt{gpu.shared}\ \delta_s}) \xrightarrow{e_g} \texttt{unit},\ \mathrm{A}\ \{\ t\ \}\ \text{where}\ \gamma \\ \overline{\Delta;\ \Gamma_0 \vdash \delta_s : \texttt{dty}} \\ \overline{\Delta;\ \Gamma_0 \vdash \tau : \kappa} \qquad \overline{\delta = \delta'\overline{[\mathrm{k} := \tau]}} \qquad \delta_r = \delta_r'\overline{[\mathrm{k} := \tau]} \\ \forall i \in \{\ 1 \ldots n\ \}.\ \Delta\,;\Gamma_{i-1} \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_{i-1} \ \vdash \boxed{p_i\ \ : \delta_i} \ \dashv\ \Gamma_i \ \mid\ \mathrm{A}_i\end{array}}{\Delta\,;\Gamma_0 \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{f{::}\langle\overline{\tau}\rangle{::}\lll \mathscr{d}_g, \mathscr{d}_b\ ;\ \overline{r, \delta_s}\ggg(\overline{p})\ \ : \texttt{unit}} \ \dashv\ \Gamma_n \ \mid\ \mathrm{A}_n}$$

T-ForNat

$$\frac{\forall i \in \eta_1..\eta_2.\ \Delta\,;\Gamma_{i-1} \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_{i-1} \ \vdash \boxed{\{\ t[n := i]\ \}\ \ : \texttt{unit}} \ \dashv\ \Gamma_i \ \mid\ \mathrm{A}_i}{\Delta\,;\Gamma_0 \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_0 \ \vdash \boxed{\texttt{for}\ n\ \texttt{in}\ \eta_1..\eta_2\ \{\ t\ \}\ \ : \texttt{unit}} \ \dashv\ \Gamma_n \ \mid\ \mathrm{A}_n}$$

T-IfElse

$$\frac{\begin{array}{c}\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_c\ \ : \texttt{bool}} \ \dashv\ \Gamma'' \ \mid\ \mathrm{A}'' \qquad \Delta\,;\Gamma'' \ \mid\ y : \varepsilon\,;\, e.\texttt{cond} \ \mid\ \mathrm{A}'' \vdash \boxed{\{\ t_1\ \}\ \ : \delta_1} \ \dashv\ \Gamma_1 \ \mid\ \mathrm{A}_1 \\ \Delta\,;\Gamma'' \ \mid\ y : \varepsilon\,;\, e.\texttt{cond} \ \mid\ \mathrm{A}'' \ \vdash \boxed{\{\ t_2\ \}\ \ : \delta_2} \ \dashv\ \Gamma_2 \ \mid\ \mathrm{A}_2 \qquad \delta = \delta_1 \vee \delta = \delta_2 \qquad \Delta\,;\Gamma_1 \ \vdash^{+}\ \delta_1 \leadsto \delta \dashv \Gamma_1' \\ \Delta\,;\Gamma_2 \ \vdash^{+}\ \delta_2 \leadsto \delta \dashv \Gamma_2' \qquad \Gamma_1' \Cup \Gamma_2' = \Gamma' \qquad \mathrm{A}_1 \Cup \mathrm{A}_2 = \mathrm{A}'\end{array}}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{\texttt{if}\ t_c\ \{\ t_1\ \}\ \texttt{else}\ \{\ t_2\ \}\ \ : \delta} \ \dashv\ \Gamma' \ \mid\ \mathrm{A}'}$$

T-Let-Uninit

$$\frac{y : \varepsilon \vdash e' : \varepsilon' \qquad \forall r \in \text{free-regions}(\delta).\ \Gamma \vdash r\ \texttt{rnrb}}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{\texttt{let}(e')\ x : \delta\ \ : \texttt{unit}} \ \dashv\ \Gamma,\ x : e'\delta^{\dagger} \ \mid\ \mathrm{A}}$$

T-Seq

$$\frac{\begin{array}{c}\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_1\ \ : \delta_1} \ \dashv\ \Gamma' \ \mid\ \mathrm{A}' \\ \Delta\,;\text{gc-loans}(\Gamma') \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_2\ \ : \delta_2} \ \dashv\ \Gamma'' \ \mid\ \mathrm{A}''\end{array}}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_1;\ t_2\ \ : \delta_2} \ \dashv\ \Gamma'' \ \mid\ \mathrm{A}''}$$

T-Block

$$\frac{\Delta\,;\Gamma, (r \mapsto \{\ \ \}) \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t\ \ : \delta} \ \dashv\ \Gamma', (\mathscr{F}) \ \mid\ \mathrm{A}'}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{[e]\{\ t\ \}^{\overline{r}}\ \ : \delta} \ \dashv\ \Gamma' \ \mid\ \text{remove-plexpr}_{\Gamma'}(\mathrm{A}')}$$

T-Tuple

$$\frac{\forall i \in \{\ 1 \ldots n\ \}.\ \Delta\,;\Gamma_{i-1}, \_ : \delta_{i-1} \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_{i-1} \ \vdash \boxed{t_i\ \ : \delta_i} \ \dashv\ \Gamma_i \ \mid\ \mathrm{A}_i}{\Delta\,;\Gamma_0 \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_0 \ \vdash \boxed{(\overline{t})\ \ : (\overline{\delta})} \ \dashv\ \text{remove-anon}(\Gamma_n) \ \mid\ \mathrm{A}_n}$$

T-Struct

$$\frac{\begin{array}{c}\mathscr{P}(s) = \texttt{struct}\ s\langle\overline{\mathrm{k}:\kappa}\rangle\{\ \overline{x : \delta}\ \} \qquad \overline{\Delta;\ \Gamma_0 \vdash \tau : \kappa} \qquad \overline{\delta' = \delta[\mathrm{k} := \tau]} \\ \forall i \in \{\ 1 \ldots n\ \}.\ \Delta\,;\Gamma_{i-1}, \_ : \delta_{i-1}' \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_{i-1} \ \vdash \boxed{t_i\ \ : \delta_i'} \ \dashv\ \Gamma_i \ \mid\ \mathrm{A}_i\end{array}}{\Delta\,;\Gamma_0 \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A}_0 \ \vdash \boxed{s\langle\overline{\tau}\rangle\{\ \overline{x : t}\ \}\ \ : s\{\ \overline{x : \delta'}\ \}} \ \dashv\ \text{remove-anon}(\Gamma_n) \ \mid\ \mathrm{A}_n}$$

T-BinOp

$$\frac{\begin{array}{c}\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_1\ \ : b} \ \dashv\ \Gamma' \ \mid\ \mathrm{A}' \\ \Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_2\ \ : b} \ \dashv\ \Gamma'' \ \mid\ \mathrm{A}''\end{array}}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t_1 \oplus t_2\ \ : b} \ \dashv\ \Gamma'' \ \mid\ \mathrm{A}''}$$

T-UnOp

$$\frac{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{t\ \ : b} \ \dashv\ \Gamma' \ \mid\ \mathrm{A}'}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{\oplus t\ \ : b} \ \dashv\ \Gamma' \ \mid\ \mathrm{A}'}$$

T-Literal

$$\frac{}{\Delta\,;\Gamma \ \mid\ y : \varepsilon\,;\, e \ \mid\ \mathrm{A} \ \vdash \boxed{\underline{l}\ \ : b} \ \dashv\ \Gamma \ \mid\ \mathrm{A}}$$

Figure 6: Other term typings rules

$\boxed{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p \;:\delta}$

$$\textsc{T-Pl-Var}\quad \frac{\Gamma(x)=\delta}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} x \;:\delta}$$

$$\textsc{T-Pl-Tuple}\quad \frac{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p \;:(\delta_1,\ldots,\delta_i,\ldots,\delta_{n-1})}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p.i \;:\delta_i}$$

$$\textsc{T-Pl-Deref}\quad \frac{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p \;:\&\rho\;\omega'\;\mu\;\delta \qquad \omega'\leq\omega}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} *p \;:\delta}$$

$$\textsc{T-Pl-View}\quad \frac{\Gamma_g(v)=[\![\delta';\,\eta]\!]\to\delta \qquad \Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e' \;\vdash^{\omega}_{\texttt{pl}} p \;:[\![\delta';\,\eta]\!]}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p.v \;:\delta}$$

$$\textsc{T-Pl-Idx}\quad \frac{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p \;:[\delta;\,\eta] \qquad \eta_i<\eta}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p[\eta_i] \;:\delta}$$

$$\textsc{T-Pl-Select}\quad \frac{y:\varepsilon\vdash e:\varepsilon' \qquad e=e^{\square}[e'.\texttt{forall}(d)] \qquad \text{not\_contains\_forall}(e^{\square}) \qquad \Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e' \;\vdash^{\omega}_{\texttt{pl}} p \;:[\![\delta;\,\eta]\!]}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\vdash^{\omega}_{\texttt{pl}} p[\![e]\!] \;:\delta}$$

Figure 7: Place expression typing rules

$\boxed{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma'}$

$$\textsc{RR-Refl}\quad \frac{}{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_1 \dashv \Gamma}$$

$$\textsc{RR-Array}\quad \frac{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma'}{\Delta\,;\Gamma \;\vdash^{\nu}\; [\delta_1;\,\eta]\rightsquigarrow[\delta_2;\,\eta] \dashv \Gamma'}$$

$$\textsc{RR-ArrayView}\quad \frac{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma'}{\Delta\,;\Gamma \;\vdash^{\nu}\; [\![\delta_1;\,\eta]\!]\rightsquigarrow[\![\delta_2;\,\eta]\!] \dashv \Gamma'}$$

$$\textsc{RR-At}\quad \frac{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma'}{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1@\mu\rightsquigarrow\delta_2@\mu \dashv \Gamma'}$$

$$\textsc{RR-Tuple}\quad \frac{\forall i\in\{\,1\ldots n\,\}.\Delta\,;\Gamma_{i-1} \;\vdash^{\nu}\; \delta_i\rightsquigarrow\delta'_i \dashv \Gamma_i}{\Delta\,;\Gamma_0 \;\vdash^{\nu}\; (\delta_1,\ldots,\delta_n)\rightsquigarrow(\delta'_1,\ldots,\delta'_n) \dashv \Gamma_n}$$

$$\textsc{RR-Reference}\quad \frac{\Delta\,;\Gamma \;\vdash^{\nu}\; \rho_1 :> \rho_2 \dashv \Gamma' \qquad \Delta\,;\Gamma' \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma''}{\Delta\,;\Gamma \;\vdash^{\nu}\; \&\rho_1\;\omega\;\mu\;\delta_1\rightsquigarrow\&\rho_2\;\omega\;\mu\;\delta_2 \dashv \Gamma''}$$

$$\textsc{RR-Dead}\quad \frac{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2 \dashv \Gamma'}{\Delta\,;\Gamma \;\vdash^{\nu}\; \delta_1\rightsquigarrow\delta_2^{\dagger} \dashv \Gamma}$$

Figure 8: Region rewriting

$\boxed{\Delta\,;\Gamma \;\vdash^{\nu}\; \rho_1 :> \rho_2 \dashv \Gamma'}$

$$\textsc{Ol-Refl}\quad \frac{}{\Delta\,;\Gamma \;\vdash^{\nu}\; \rho :> \rho \dashv \Gamma}$$

$$\textsc{Ol-BothAbstract}\quad \frac{\imath_1:\textbf{rgn}\in\Delta \qquad \imath_2:\textbf{rgn}\in\Delta \qquad \imath_1 :> \imath_2\in\Delta}{\Delta\,;\Gamma \;\vdash^{\nu}\; \imath_1 :> \imath_2 \dashv \Gamma}$$

$$\textsc{Ol-CombineConcrete}\quad \frac{\Gamma\vdash r_1\;\texttt{rnrb} \qquad \Gamma\vdash r_1\;\texttt{rnrb} \qquad r_1\;\texttt{occurs before}\;r_2\;\texttt{in}\;\Gamma \qquad \{\,\overline{\ell}\,\}=\Gamma(r_1)\cup\Gamma(r_2)}{\Delta\,;\Gamma \;\vdash^{+}\; r_1 :> r_2 \dashv \Gamma[r_2\mapsto\{\,\overline{\ell}\,\}]}$$

$$\textsc{Ol-CheckConcrete}\quad \frac{\Gamma\vdash r_1\;\texttt{rnrb} \qquad \Gamma\vdash r_1\;\texttt{rnrb} \qquad r_1\;\texttt{occurs before}\;r_2\;\texttt{in}\;\Gamma}{\Delta\,;\Gamma \;\vdash^{=}\; r_1 :> r_2 \dashv \Gamma}$$

$$\textsc{Ol-AbstractConcrete}\quad \frac{\imath:\textbf{rgn}\in\Delta \qquad r\in\mathrm{dom}(\Gamma)}{\Delta\,;\Gamma \;\vdash^{\nu}\; \imath :> r \dashv \Gamma}$$

$\boxed{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\mid\; \mathrm{A} \;\vdash^{\omega}\; \textbf{borrow}^{+}\;p\Rightarrow\{\,{}^{\omega}p_i\,\}}$

$$\frac{\text{narrowable}(\Gamma,\omega,e,p) \qquad \mathrm{A}\vdash^{\omega}p \qquad \Gamma\;\vdash^{\omega}\;\textbf{borrow}\;p\Rightarrow\{\,{}^{\omega}p_i\,\}}{\Delta\,;\Gamma \;\mid\; y:\varepsilon\,;\,e \;\mid\; \mathrm{A} \;\vdash^{\omega}\; \textbf{borrow}^{+}\;p\Rightarrow\{\,{}^{\omega}p_i\,\}}$$

4 Auxiliary Definitions

$\boxed{\text{narrowable}(\Gamma,\omega,e,p)}$

The predicate narrowable(Γ,ω,e,p) inidicates whether place expression p narrows parallel accesses such that execution resource e can borrow or access p if $\omega = \texttt{uniq}$:

$$\text{narrowable}(\Gamma, \omega, e, p) := \quad (\omega = \texttt{uniq}) \Rightarrow \text{exec}(\Gamma, p) = e$$

$\boxed{\pi = \sigma \cdot \sigma}$: Path
A path π is a list of σ, where $\sigma \in \{\ .\texttt{j} \mid .x \mid * \mid [i] \mid [\![e]\!] \mid f_v{::}\langle\overline{\tau}\rangle(\overline{v})\ \}$.

$\boxed{p = x|\pi}$
The equation $p = x|\pi$ describes how a place expression p is decomposed into it's root, the variable x, and it's path:

$$\begin{aligned}
x &= x \mid \bullet \\
p.\texttt{j} &= x \mid (\pi \cdot .\texttt{j}) \quad \text{for } p = x \mid \pi \\
p.x' &= x \mid (\pi \cdot .x') \quad \text{for } p = x \mid \pi \\
*p &= x \mid (\pi \cdot *) \quad \text{for } p = x \mid \pi \\
p[i] &= x \mid (\pi \cdot [i]) \quad \text{for } p = x \mid \pi \\
p[\![e]\!] &= x \mid (\pi \cdot [\![e]\!]) \quad \text{for } p = x \mid \pi \\
p.f_v{::}\langle\overline{\tau}\rangle(\overline{v}) &= x \mid (\pi \cdot f_v{::}\langle\overline{\tau}\rangle(\overline{v})) \quad \text{for } p = x \mid \pi
\end{aligned}$$

$\boxed{\text{race-free}_O(\pi_1, \pi_2)}$

$$\frac{}{\text{race-free}_{\text{Eq}}(nil, \pi)} \qquad \frac{\text{race-free}_{\text{Eq}}(\pi, nil) \qquad \sigma \neq [\![e]\!]}{\text{race-free}_{\text{Eq}}(\sigma \cdot \pi, nil)} \qquad \frac{\text{race-free}_{\text{Eq}}(\pi_{\text{prev}}, \pi_{\text{current}})}{\text{race-free}_{\text{Eq}}(* \cdot \pi_{\text{prev}}, * \cdot \pi_{\text{current}})}$$

$$\frac{.\mathsf{i} \neq .\mathsf{j}}{\text{race-free}_{\text{Eq}}(.\mathsf{i} \cdot \pi_{\text{prev}}, .\mathsf{j} \cdot \pi_{\text{current}})_{\text{None}}} \qquad \frac{\text{race-free}_{\text{Eq}}(\pi_{\text{prev}}, \pi_{\text{current}})}{\text{race-free}_{\text{Eq}}(.\mathsf{j} \cdot \pi_{\text{prev}}, .\mathsf{j} \cdot \pi_{\text{current}})} \qquad \frac{\text{race-free}_{\text{Eq}}(\pi_{\text{prev}}, \pi_{\text{current}})}{\text{race-free}_{\text{Eq}}([\![e]\!] \cdot \pi_{\text{prev}}, [\![e]\!] \cdot \pi_{\text{current}})}$$

$$\frac{\text{race-free}_O(\pi_{\text{prev}}, \pi_{\text{current}})}{\text{race-free}_O([\mathsf{i}] \cdot \pi_{\text{prev}}, [\mathsf{j}] \cdot \pi_{\text{current}})} \qquad \frac{\begin{array}{c}\text{race-free}_{O'}(\pi_{\text{prev}}, \pi_{\text{current}}) \\ \text{racy-overlap}(v_1 \cdot \ldots \cdot v_m, v_1' \cdot \ldots \cdot v_n') = O' \qquad \sigma \notin v\end{array}}{\text{race-free}_{\text{Eq}}(v_1 \cdot \ldots v_m \cdot \sigma \cdot \pi_{\text{prev}}, v_1' \cdot \ldots v_n' \cdot \sigma \cdot \pi_{\text{current}})}$$

$$\frac{\text{race-free}_{\text{Overlap}}(\pi_{\text{prev}}, \pi_{\text{current}}) \qquad \sigma \notin v}{\text{race-free}_{\text{Overlap}}(v_1 \cdot \ldots v_m \cdot \sigma \cdot \pi_{\text{prev}}, v_1' \cdot \ldots v_n' \cdot \sigma \cdot \pi_{\text{current}})}$$

$\boxed{\text{A} \vdash^{\omega} p}$ Access Conflict Check:

$$\frac{p = x|\pi \qquad \forall\ ^{\omega'}p_{\text{A}} \in \text{A}.\ \mathit{if}\ ((\omega = \texttt{uniq} \vee \omega_{\text{A}} = \texttt{uniq}) \wedge p_{\text{A}} = x|\pi_{\text{A}})\ \mathit{then}\ \text{race-free}_{\text{Eq}}(\pi_{\text{A}}, \pi)}{\text{A} \vdash^{\omega} p}$$

$\boxed{\pi \bowtie \pi'}$: Paths π and π' are in a memory conflict,
i.e., they can point to the same memory location if applied to the same variable.
(This substitutes the standard conflict check in Oxide's borrowing rules.)

$$\frac{\pi \bowtie \pi'}{\sigma \cdot \pi \bowtie \sigma \cdot \pi'} \qquad \frac{\sigma \neq \sigma' \qquad \neg(\sigma, \sigma' \in \{\ .\texttt{fst}, .\texttt{snd}\ \})}{\sigma \cdot \pi \bowtie \sigma' \cdot \pi'} \qquad \frac{}{\epsilon \bowtie \epsilon}$$

$\boxed{\text{isCopyable}(\delta)}$

$$\begin{aligned}
&\text{isCopyable}(\delta) = \text{true} \quad \text{for } \delta \in \{\ \texttt{bool},\ \texttt{int},\ \texttt{float},\ \texttt{AtomicU32},\ \&\varrho\ \texttt{shrd}\ \mu\ \delta\ \} \\
&\text{isCopyable}((\delta_1, \ldots, \delta_n)) = \forall i.\ \text{isCopyable}(\delta_i) \\
&\text{isCopyable}(s\{\ x_1 : \delta_1, \ldots, x_n : \delta_n\ \}) = \forall i.\text{isCopyable}(\delta_i) \\
&\text{isCopyable}([\delta;\ \eta]) = \text{isCopyable}(\delta) \\
&\text{isCopyable}(\delta) = \text{false} \quad \text{else}
\end{aligned}$$

$\boxed{\text{isPlace}(p)}$

$$\text{isPlace}(p) := (p = x \mid \pi) \Rightarrow \forall \sigma \in \pi.\ \sigma \in \{\ .\mathtt{j}, .x\ \}$$

$\boxed{\text{release-ownership}(e, p)}$

$$\text{release-ownership}(e, p_1) = \begin{cases} p_2 & \text{if there exist } \pi_1 \text{ and } \pi_2 \text{ such that } p_1 = x \mid \pi_1 \cdot [\![e]\!] \cdot \pi_2 \text{ and } p_2 = x \mid \pi_1 \cdot [\![e]\!] \\ p_1 & \text{else} \end{cases}$$

$\boxed{\text{remove-anon}(\mathcal{F})}$

$$\begin{aligned} &\text{remove-anon}(\bullet) = \bullet \\ &\text{remove-anon}(\mathcal{F}, \_ : e\ \tilde{\delta}) = \text{remove-anon}(\mathcal{F}) \\ &\text{remove-anon}(\mathcal{F}, x : e\ \tilde{\delta}) = \text{remove-anon}(\mathcal{F}), x : e\ \tilde{\delta} \\ &\text{remove-anon}(\mathcal{F}, r \mapsto \{\ \bar{\ell}\ \}) = \text{remove-anon}(\mathcal{F}), r \mapsto \{\ \bar{\ell}\ \} \end{aligned}$$

$\boxed{\text{remove-anon}(\Gamma)}$

$$\begin{aligned} &\text{remove-anon}(\bullet) = \bullet \\ &\text{remove-anon}(\Gamma, (\mathcal{F})) = \text{remove-anon}(\Gamma, (\text{remove-anon}(\mathcal{F}))) \end{aligned}$$

$\boxed{A +{}^{\omega} p}$

$$\begin{aligned} A_1, {}^{\omega'}p', A_2 +{}^{\omega}p &= A_1, A_2, {}^{\omega}p \quad \text{if } \exists^{\omega'} p' \in A.\ p' \text{ is prefix of } p \\ A +{}^{\omega}p &= A, {}^{\omega}p \quad \text{else} \end{aligned}$$

$\boxed{A_1 \Cup A_2}$

$$\begin{aligned} A_1 \Cup (A_2, {}^{\omega}p) &= (A_1 +{}^{\omega}p) \Cup A_2 \\ A \Cup \bullet &= A \end{aligned}$$