Pony

PAUL LIÉTAR

Text of abstract ....

1 ACTORS

$$
\begin{array}{rclcl}
\texttt{P} & \in & \mathit{Program} & ::= & \overline{\texttt{CT}}\ \overline{\texttt{AT}} \\
\texttt{CT} & \in & \mathit{ClassDef} & ::= & \texttt{class}\ \texttt{C}\ \overline{\texttt{F}} \\
\texttt{AT} & \in & \mathit{ActorDef} & ::= & \texttt{actor}\ \texttt{A}\ \overline{\texttt{F}}\ \overline{\texttt{B}} \\
\texttt{F} & \in & \mathit{Field} & ::= & \texttt{var}\ f : \texttt{S}\ \kappa \\
\texttt{B} & \in & \mathit{Behv} & ::= & \texttt{be}\ b(\overline{x : \texttt{S}\ \kappa}) \Rightarrow e \\
e & \in & \mathit{Expr} & ::= & x \mid x = e \mid e.f \mid e.f = e \\
 & & & \mid & e.b(\overline{e}) \mid e; e \mid \texttt{new}\ \texttt{S}
\end{array}
$$

Fig. 1. Syntax

$$
\begin{array}{rclcl}
\chi & \in & \mathit{Heap} & = & \mathit{Addr} \rightarrow (\mathit{Actor} \cup \mathit{Object}) \\
\nu & \in & \mathit{Value} & = & \mathit{Addr} \cup \{\mathit{null}\} \\
\iota & \in & \mathit{Addr} & = & \mathit{ActorAddr} \cup \mathit{ObjectAddr} \\
\alpha & \in & \mathit{ActorAddr} & & \\
\omega & \in & \mathit{ObjectAddr} & & \\
 & & \mathit{Object} & = & \mathit{ClassID} \times (\mathit{FieldID} \rightarrow \mathit{Value}) \\
 & & \mathit{Actor} & = & \mathit{ActorID} \times (\mathit{FieldID} \rightarrow \mathit{Value}) \\
 & & & & \times\ \overline{\mathit{Message}} \times \mathit{Locals} \times \mathit{Expr} \\
\varphi & \in & \mathit{Locals} & = & \mathit{LocalID} \rightarrow \mathit{Value} \\
 & & \mathit{LocalID} & = & \mathit{SourceID} \times \mathit{TempID} \\
\mu & \in & \mathit{Message} & = & \mathit{MethodID} \times \overline{\mathit{Value}}
\end{array}
$$

Fig. 2. Runtime entities

$$\boxed{\chi, \varphi, e \rightsquigarrow \chi, \varphi, e}$$

$$\frac{\varphi' = \varphi[t_a \mapsto \varphi(x)]}{\chi, \varphi, x \rightsquigarrow \chi, \varphi', t_a}\ \textsc{Local} \qquad \frac{\varphi' = \varphi[t_a \mapsto \varphi(x), x \mapsto \varphi(t_p)]}{\chi, \varphi, x = t_p \rightsquigarrow \chi, \varphi', t_a}\ \textsc{AsnLocal}$$

$$\frac{\varphi' = \varphi[t_a \mapsto \chi(t_a, f)]}{\chi, \varphi, t_a.f \rightsquigarrow \chi, \varphi', t_a}\ \textsc{Fld} \qquad \frac{\begin{array}{c}\iota = \varphi(t_a) \\ \chi' = \chi[\iota, f \mapsto \varphi(t_p)] \\ \varphi' = \varphi[t_a \mapsto \chi(\iota, f)]\end{array}}{\chi, \varphi, t_a.f = t_p \rightsquigarrow \chi, \varphi', t_a}\ \textsc{AsnFld}$$

$$\frac{\begin{array}{c}\alpha \notin \chi \qquad \overline{f} = \mathcal{F}s(\texttt{A}) \\ \chi' = \chi[\alpha \mapsto (\texttt{A}, \overline{f} \mapsto \mathit{null}, \epsilon, \epsilon, \epsilon)] \\ \varphi' = \varphi[t_a \mapsto \alpha]\end{array}}{\chi, \varphi, \texttt{new A} \rightsquigarrow \chi, \varphi, t_a}\ \textsc{Ator} \qquad \frac{\begin{array}{c}\omega \notin \chi \qquad \overline{f} = \mathcal{F}s(\texttt{C}) \\ \chi' = \chi[\alpha \mapsto (\texttt{C}, \overline{f} \mapsto \mathit{null})] \\ \varphi' = \varphi[t_a \mapsto \omega]\end{array}}{\chi, \varphi, \texttt{new C} \rightsquigarrow \chi, \varphi, t_a}\ \textsc{Ctor}$$

$$\frac{}{\chi, \varphi, t_a; e \rightsquigarrow \chi, \varphi, e}\ \textsc{Seq} \qquad \frac{\begin{array}{c}\alpha = \varphi(t_p) \qquad \overline{\mu} = \chi(\alpha)\downarrow_3 \\ \chi' = \chi[\alpha \mapsto \overline{\mu} \cdot (b, \overline{\varphi(t_p)})]\end{array}}{\chi, \varphi, t_p.b(\overline{t_p}) \rightsquigarrow \chi', \varphi, t_p}\ \textsc{Async}$$

Fig. 3. Local Execution

2 REFERENCE CAPABILITIES

$$\begin{aligned}\kappa &::= \texttt{iso} \mid \texttt{trn} \mid \texttt{ref} \mid \texttt{val} \mid \texttt{box} \mid \texttt{tag} \\ \lambda &::= \kappa \mid \texttt{iso}- \mid \texttt{trn}-\end{aligned}$$

$$\texttt{iso}- \leq \{\texttt{iso}, \texttt{trn}-\} \qquad \texttt{trn}- \leq \{\texttt{trn}, \texttt{ref}, \texttt{val}\} \leq \texttt{box} \qquad \{\texttt{iso}, \texttt{box}\} \leq \texttt{tag} \qquad \lambda \leq \lambda$$

$$\frac{\lambda \leq \lambda'' \qquad \lambda'' \leq \lambda'}{\lambda \leq \lambda'}$$

3 TYPING RULES

4 RUNTIME

4.1 Active and Passive Temporaries

4.2 Visibility

$$\begin{array}{lcll} p & \in & \mathit{Path} & ::= \texttt{this} \mid x \mid t_a.f \mid p.f \\ pe & \in & \mathit{PathExt} & ::= p \mid t_a \\ \lambda s & \in & \mathit{CapSet} & ::= \cdots \end{array}$$

$$\boxed{\Gamma \vdash e : \mathtt{S}\ \lambda}$$

$$\frac{x \in \Gamma}{\Gamma \vdash x : \Gamma(x)}\ \textsc{T-Local} \qquad \frac{\Gamma(x) = \mathtt{S}\ \kappa \qquad \Gamma \vdash_{\mathcal{A}} e : \mathtt{S}\ \kappa}{\Gamma \vdash x = e : \mathcal{U}(\mathtt{S}\ \kappa)}\ \textsc{T-AsnLocal}$$

$$\frac{\Gamma \vdash e : \mathtt{S}\ \lambda \qquad \mathit{Field}(\mathtt{S}, f) = \mathtt{S}'\kappa}{\Gamma \vdash e.f : \mathtt{S}'\ (\lambda \rhd \kappa)}\ \textsc{T-Fld} \qquad \frac{\begin{array}{c}\Gamma \vdash e : \mathtt{S}\ \lambda \qquad \mathit{Field}(\mathtt{S}, f) = \mathtt{S}'\kappa \\ \Gamma \vdash_{\mathcal{A}} e' : \mathtt{S}'\ \kappa \\ \Gamma \vdash e' : \mathtt{S}'\ \lambda' \qquad \lambda \lhd \mathcal{A}(\lambda')\end{array}}{\Gamma \vdash e.f = e' : \mathtt{S}'\ (\lambda \blacktriangleright \kappa)}\ \textsc{T-AsnFld}$$

$$\frac{\mathtt{C} \in \mathtt{P}}{\mathtt{new\ C} : \mathtt{C\ iso}-}\ \textsc{T-Ctor} \qquad \frac{\mathtt{A} \in \mathtt{P}}{\mathtt{new\ A} : \mathtt{A\ tag}}\ \textsc{T-Ator} \qquad \frac{\Gamma \vdash e : \mathtt{S}\ \lambda \qquad \Gamma \vdash e' : \mathtt{S}'\ \lambda'}{\Gamma \vdash e; e' : \mathtt{S}'\ \lambda'}\ \textsc{T-Seq}$$

$$\frac{\begin{array}{c}\Gamma \vdash e : \mathtt{A}\ \lambda \qquad \overline{\Gamma \vdash_{\mathcal{A}} e : \mathtt{S}\ \lambda} \\ \mathit{Behv}(A, b) = \overline{\mathtt{S}\ \lambda}\end{array}}{\Gamma \vdash e.b(\overline{e}) : \mathtt{A}\ \lambda}\ \textsc{T-Behv}$$

Fig. 4. Typing rules

$$\boxed{\Delta, \chi, \iota \vdash \iota : \lambda s, pe}$$

$$\frac{\iota \in \chi}{\Delta, \chi, \iota \vdash \iota : \{\mathtt{ref}\}, \mathtt{this}}\ \textsc{V-This} \qquad \frac{\begin{array}{c}\Delta, \chi, \iota \vdash \iota' : \lambda s, pe \qquad \chi(\iota', f) = \iota'' \\ \mathit{Field}(\chi(\iota) \downarrow_1, f) = \mathtt{S}\ \kappa\end{array}}{\Delta, \chi, \iota \vdash \iota'' : \lambda s \circ \kappa, pe.f}\ \textsc{V-Fld}$$

$$\frac{\chi(\alpha, x) = \iota \qquad \Delta(\alpha, x) = \mathtt{S}\ \kappa}{\Delta, \chi, \alpha \vdash \iota : \{\kappa, \mathcal{U}(\kappa)\}, x}\ \textsc{V-Local} \qquad \frac{\chi(\alpha, t_a) = \iota \qquad \Delta(\alpha, t_a) = \mathtt{S}\ \lambda}{\Delta, \chi, \alpha \vdash \iota : \{\lambda\}, t_a}\ \textsc{V-Active}$$

Fig. 5. Visibility

4.3 Well-formed visibility

$\mathit{WFV}(\Delta, \chi)$ iff $\forall \alpha, \alpha', \iota, p, p', \lambda s, \lambda s', \iota_0, p_0, \lambda_0$

(1) If $\alpha \neq \alpha'$ and $\Delta, \chi, \alpha \vdash \iota : \lambda s, p$ and $\Delta, \chi, \alpha' \vdash \iota : \lambda s', p'$ then $\lambda s \sim_g \lambda s'$

(2) If $\Delta, \chi, \alpha \vdash \iota_0 : \lambda s_0, p_0$ and $\mathit{Writable}(\lambda s_0)$ and $\Delta, \chi, \iota_0 \vdash \iota : \lambda s, p$ and $\Delta, \chi, \iota_0 \vdash \iota : \lambda s', p'$ then either
   (a) $\lambda s \sim_l \lambda s'$ or
   (b) $\chi, \alpha \vdash \mathit{Overlap}(p, p')$

(3) If $\Delta, \chi, \alpha \vdash \iota : \lambda s, p$ and $\Delta, \chi, \alpha \vdash \iota : \{\lambda\}, t_a$ then $\lambda s \sim_l \mathcal{A}(\lambda)$

$\chi, \alpha \vdash \mathit{Overlap}(p, p')$ iff $p = p'$ or $\exists pe, pe', f, f', \overline{f}, \overline{f'}$ such that $p = pe.f.\overline{f}$ and $p' = pe'.f'.\overline{f'}$ and $\chi(\alpha, pe) = \chi(\alpha, pe')$.

5 RECOVERY

5.1 Regioned capabilities

$$\begin{aligned}\lambda ::= \;& \texttt{iso-} \mid \texttt{iso} \mid \texttt{val} \mid \texttt{tag} \\ & \mid (\texttt{trn-}, r_{id}) \mid (\texttt{trn}, r_{id}) \mid (\texttt{ref}, r_{id}) \mid (\texttt{box}, r_{id})\end{aligned}$$

5.2 Well-formed regions

$WFR(\Delta, \chi)$ iff $\forall \alpha, \iota, z, z', \overline{f}, \overline{f'}, \lambda, \lambda'$,
If $z \neq z'$ and $\Delta, \chi, \alpha \vdash \iota : \lambda s, z.\overline{f}$ and $\Delta, \chi, \alpha \vdash \iota : \lambda s', z'.\overline{f'}$ then either

(1) $Region(\lambda s) = Region(\lambda s')$ or
(2) $\lambda s \sim_g \lambda s'$