Session-Typed Concurrent Programming

Stephanie Balzer
Carnegie Mellon University

July 21, 2018

1 CC0 - Concurrency / Tests / Sharing

$A, B, C ::= A \otimes B \mid A \multimap B \mid \&\{l : \bar{A}\} \mid \oplus\{l : \bar{A}\} \mid !A$
$\Psi, \Delta \vdash P :: (x : A)$
Also have: $\texttt{cut}!$, $\texttt{copy}$, $!R$, $!L$
And process replication $!P$ with $!P \equiv P \mid !P$
$Q \longleftrightarrow_u !P \xrightarrow{\texttt{copy}} (Q \longleftrightarrow_u !P \text{ and } Q \longleftrightarrow_x P)$
$(C_1 \longleftrightarrow_u !P \text{ and } C_2 \longleftrightarrow_u !P) \xrightarrow{\texttt{copy}}$
$(C_1 \longleftrightarrow_u !P; C_2 \longleftrightarrow_u !P; C_1 \longleftrightarrow_x P; C_2 \longleftrightarrow_y P')$
where P, P' are individual copies of P

2 Manifest Sharing

What is the type of q?

$$\begin{aligned}\texttt{queue}A = \&\{&\texttt{enq} : A \multimap \texttt{queue}A,\\ &\texttt{deq} : \oplus\{\texttt{none} : \texttt{queue}A, \texttt{some} : A \otimes \texttt{queue}A\}\}\end{aligned}$$

Must acquire after the equal sign, and release before each $\texttt{queue}A$.
What about session types?
Shared space:

$$A_S := \uparrow^S_L A_L$$

Linear space:

$$\begin{aligned}A_L, B_L := \oplus\{l : \bar{A}_L\} \mid A_L \otimes B_L \mid 1 \mid \exists x : A_S.B_S\\ \mid \&\{l : \bar{A}_L\} \mid A_L \multimap B_L \mid \downarrow^S_L A_S \mid \Pi x : A_S.B_S\end{aligned}$$

The shared layer is higher up in the hierarchy than the linear layer. This means a shared proposition cannot rely on linear sources when typing.
Now we can rewrite the $\texttt{queue}$.

$$\begin{aligned}\texttt{queue}A_S = \uparrow^S_L \&\{&\texttt{enq} : \Pi x : A_S.\downarrow^S_L \texttt{queue}A_S,\\ &\texttt{deq} : \oplus\{\texttt{none} : \downarrow^S_L \texttt{queue}A_S, \texttt{some} : \exists x : A_S.\downarrow^S_L \texttt{queue}A_S\}\}\end{aligned}$$

2.1 Processes

Linear Process: $\Gamma; \Delta \vdash P : (x_L : A_L)$
Shared Process: $\Gamma \vdash P : (x_S : A_S)$
Δ: Linear
Γ: Structural

$$\frac{\Gamma, x_S :\uparrow^S_L A_L; \Delta, x_L : A_L \vdash Q_{x_L} :: (z_L : C_L)}{\Gamma, x_s :\uparrow^S_L; \Delta \vdash x_L \leftarrow \texttt{acquire}\ x_S; Q_{x_L} :: (z_L : C_L)}(\uparrow^S_L L)$$

$$\frac{\Gamma; \cdot \vdash P_{x_L} :: (x_L : A_L)}{\Gamma \vdash x_L \leftarrow \texttt{accept}\ x_S; P_{x_L} :: (x_S :\uparrow^S_L A_L)}(\uparrow^S_L R)$$

$$\frac{\Gamma, x_S : A_S; \Delta \vdash Q_{x_S} :: (z_L : C_L)}{\Gamma; \Delta, x_L :\downarrow^S_L A_S \vdash x_S \leftarrow \texttt{release}\ x_L; Q_{x_S} :: (z_L : C_L)}(\downarrow^S_L L)$$

$$\frac{\Gamma \vdash P_{x_S} :: (x_S : A_S)}{\Gamma; \cdot \vdash x_S \leftarrow \texttt{detach}\ x_L; P_{x_S} :: (x_L :\downarrow^S_L A_S)}(\downarrow^S_L R)$$

2.2 Preservation?

Not yet. We need well formed session types: equi-synchronizing.

2.3 Progress

With sharing we get a weaker form of progress, possible to have cycles.
Processes can now try to acquire themselves.

3 Concluding Thoughts

Deadlock is part of the current research.
With sharing we are back into full blow π-calculus with all of its expressiveness, but also all of the issues.
New paper (September presentation) presents an encoding of an asynchronous, untyped π-calculus into shared session-typed process calculus.
Linear propositions correspond to session types, and proofs correspond to processes. Linear propositions are an interleaving of proof construction, proof reduction, and proof deconstruction. These correspond to acquire, communication, and release respectively. Deadlock corresponds to proof construction.