Binary Session Types

Hernán Melgratti

ICC University of Buenos Aires-Conicet

Terminology

We say P is *well-typed* if there exists Γ s.t $\Gamma \vdash P$

Example

Is P well-typed?

- $P = x^+?(y{:}\mathtt{int}).x^+!\mathtt{true}.0$
- Yes! take $\Gamma = x^+ : ?\mathtt{int}.!\mathtt{bool}.\mathtt{end}$

$$\dfrac{\dfrac{\emptyset \vdash \mathtt{true} : \mathtt{bool} \qquad \dfrac{x^+ : \mathtt{end}, y : \mathtt{int}\ \text{completed}}{}\,[\text{T-Nil}]}{x^+ : \mathtt{end}, y : \mathtt{int} \vdash 0}\,[\text{T-Out}]}{x^+ : !\mathtt{bool}.\mathtt{end}, y : \mathtt{int} \vdash x^+!\mathtt{true}.0}\,[\text{T-In}]$$

$$x^+ : ?\mathtt{int}.!\mathtt{bool}.\mathtt{end} \vdash x^+?(y{:}\mathtt{int}).x^+!\mathtt{true}.0$$

Example

Is P well-typed?

- $P = x^+?(y{:}!\texttt{bool}.\texttt{end}).y\,!\texttt{true}.0$
- Yes! take $\Gamma = x^+ : ?(!\texttt{bool}.\texttt{end}).\texttt{end}$

$$\dfrac{\dfrac{\dfrac{\emptyset \vdash \texttt{true} : \texttt{bool} \qquad \dfrac{x^+ : \texttt{end}, y : \texttt{end}\ \text{completed}}{}\,[\text{T-Nil}]}{x^+ : \texttt{end}, y : \texttt{end} \vdash 0}\,[\text{T-Out}]}{x^+ : \texttt{end}, y : !\texttt{bool}.\texttt{end} \vdash y\,!\texttt{true}.0}\,[\text{T-In}]}{x^+ : ?(!\texttt{bool}.\texttt{end}).\texttt{end} \vdash x^+?(y{:}!\texttt{bool}.\texttt{end}).y\,!\texttt{true}.0}$$

Example

Is P well-typed?

- $P = x^+?(y{:}!\texttt{int}.\texttt{end}).y\,!\texttt{true}.0$
  No!
  Try with $\Gamma = x^+ : ?(!\texttt{int}.\texttt{end}).\texttt{end}$

$$\dfrac{\dfrac{\dfrac{\emptyset \not\vdash \texttt{true} : \texttt{int} \qquad \dfrac{x^+ : \texttt{end}, y : \texttt{end}\ \text{completed}}{}\,[\text{T-Nil}]}{x^+ : \texttt{end}, y : \texttt{end} \vdash 0}\,[\text{T-Out}]}{x^+ : \texttt{end}, y : !\texttt{int}.\texttt{end} \vdash y\,!\texttt{true}.0}\,[\text{T-In}]}{x^+ : ?(!\texttt{int}.\texttt{end}).\texttt{end} \vdash x^+?(y{:}!\texttt{int}.\texttt{end}).y\,!\texttt{true}.0}$$

Example

Is P well-typed?

- $P = x^+?(y{:}?\mathtt{int}.\mathtt{end}).y\,!1.0$
  No! Try with $\Gamma = x^+ : ?(?\mathtt{int}.\mathtt{end}).\mathtt{end}$

$$\dfrac{\dfrac{\text{There is a mismatch: } y : ?\mathtt{int}.\mathtt{end} \text{ and } y\,!1.0}{x^+ : \mathtt{end}, y : ?\mathtt{int}.\mathtt{end} \vdash y\,!1.0}\,[\text{T-Out}]}{x^+ : ?(?\mathtt{int}.\mathtt{end}).\mathtt{end} \vdash x^+?(y{:}?\mathtt{int}.\mathtt{end}).y\,!1.0}\,[\text{T-In}]$$

Rethinking $(\texttt{int} \to \texttt{bool}) \to \texttt{bool}$

$\texttt{?(?int.!bool.end).!bool.end}$ is not $(\texttt{int} \to \texttt{bool}) \to \texttt{bool}$

$$\begin{aligned} g &: (\texttt{int} \to \texttt{bool}) \to \texttt{bool} \\ \texttt{g f} &= \texttt{f 1} \end{aligned}$$

We may implement g as

$$P_g = x^+\texttt{?}(y\texttt{:t}_\texttt{f})\texttt{.}y\texttt{!1.}y\texttt{?}(z\texttt{:bool})\texttt{.}x^+\texttt{!}z\texttt{.0}$$

where $\texttt{t}_\texttt{f} = \texttt{?int.!bool.end}$

But

$$x^+ : \texttt{?t}_\texttt{f}\texttt{.!bool.end} \not\vdash P_g$$

However

$$x^+ : \texttt{?}\overline{\texttt{t}_\texttt{f}}\texttt{.!bool.end} \vdash P_g$$

What about ?(!int.?bool.end).!bool.end?

$$g : (\texttt{int} \to \texttt{bool}) \to \texttt{bool}$$
$$\texttt{g y} = \texttt{(y 1) \& (y 2)}$$

We may implement g as

$$P_g = x^+?(y{:}\overline{\texttt{t}_\texttt{f}}).y\,!1.y\,?(z_1{:}\texttt{bool}).y\,!2.y\,?(z_2{:}\texttt{bool}).x^+\,!z_1\&z_2.0$$

where $\texttt{t}_\texttt{f} = \texttt{?int.!bool.end}$

However

$$x^+ : ?\overline{\texttt{t}_\texttt{f}}.!\texttt{bool.end} \not\vdash P_g$$

The parameter y **must** be used just for **one** application

On linearity

- Consider $P = x^+!y^+.y^+!1.0$.
- Does the following hold?

$$\Gamma, x^+ : !(!\texttt{int}.\texttt{end}).\texttt{end}, y^+ : !\texttt{int}.\texttt{end} \vdash P$$

$$\frac{\Gamma_1 \vdash v : t \qquad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : !t.S) \vdash x^p!v.P}\ [\text{T-Out}]$$

- No. Why does [T-Out] ban P?
- Take

$$Q = (\nu y{:}!\texttt{int}.\texttt{end})(\nu x{:}!(!\texttt{int}.\texttt{end}).\texttt{end})(\\ P \mid x^-?(z{:}!\texttt{int}.\texttt{end}).z!2.0 \mid y^-?(z{:}\texttt{int}).0)$$

- $Q \xrightarrow{\tau, -} Q'$ where

$$Q' = (\nu y{:}!\texttt{int}.\texttt{end})(\nu x{:}\texttt{end})(y^+!1.0 \mid y^+!2.0 \mid y^-?(z{:}\texttt{int}).0)$$

  where two processes concurrently send on y^+
- A process does not use a session endpoint after *delegating* it (i.e., sending it over a different session endpoint)

Results

Theorem (Type Preservation)

- *If* $\Gamma \vdash P$ *and* $P \xrightarrow{\tau,-} Q$ *then* $\Gamma \vdash Q$.
- *If* $\Gamma, x^p : S, x^{\overline{p}} : \overline{S} \vdash P$ *and* $P \xrightarrow{x,\mathtt{l}} Q$ *then* $S \xrightarrow{\mathtt{l}} T$ *and* $\Gamma, x^p : T, x^{\overline{p}} : \overline{T} \vdash Q$.

Theorem (Type Safety)

Let $\Gamma \vdash P$ *where* Γ *balanced* [2]

- *If* $P \equiv (\nu \tilde{z}{:}\tilde{S})(x^p\,!\,v\,.\,P_1 \mid x^{\overline{p}}?(y{:}\mathtt{t})\,.\,P_2 \mid Q)$ *with* $p \in \{+,-\}$ *then* $x, x^+, x^- \notin \mathrm{fn}(Q)$ *and* $\Gamma, \tilde{z} : \tilde{S} \vdash v : t$
- *If* $P \equiv (\nu \tilde{z}{:}\tilde{S})(x^p \triangleleft \mathtt{l}_j\,.\,R \mid x^{\overline{p}} \triangleright [\mathtt{l}_i : P_i]_{i \in I} \mid Q)$ *with* $p \in \{+,-\}$ *then* $x, x^+, x^- \notin \mathrm{fn}(Q)$ *and* $j \in I$.

[2] Γ is balanced if $x^p : S$ and $x^{\overline{p}} : T$ implies $S = \overline{T}$

Properties

Does the following hold?

$$\vdash (\nu x{:}?\texttt{int}.\texttt{end})(x^{+}?(z{:}\texttt{int}).x^{-}!z.0)$$

Yes!

- The process is well-typed and deadlocked

The type system ensures

- *Type Safety* in communication (e.g., received values are of the expected type)
- *Session Fidelity* (e.g., communication follows the flow described by the session type)
- The type system does not ensure deadlock-freedom

Deadlock

Deadlocked Process

$$P = x^{+}?(z{:}\mathtt{int}).y^{-}!1.0 \quad | \quad y^{+}?(z{:}\mathtt{int}).x^{-}!1.0$$

Is P well-typed?

$$\vdash (\nu x{:}?\mathtt{int}.\mathtt{end})(\nu y{:}?\mathtt{int}.\mathtt{end})P$$

Yes!

- The process is well-typed and deadlocked
- The type system does not check the dependencies between different sessions

Deadlock-freedom by design (linear logic approaches)

- Connection drawn between linear logic and session-typed pi-calculus gave rise to type systems that guarantee deadlock-freedom
  - Luís Caires, Frank Pfenning: Session Types as Intuitionistic Linear Propositions. CONCUR 2010.
  - Philip Wadler: Propositions as sessions. ICFP 2012.
- The type system imposes some structural constraint on programs
  - two processes share at most one channel
  - Hence, there are no circular dependencies