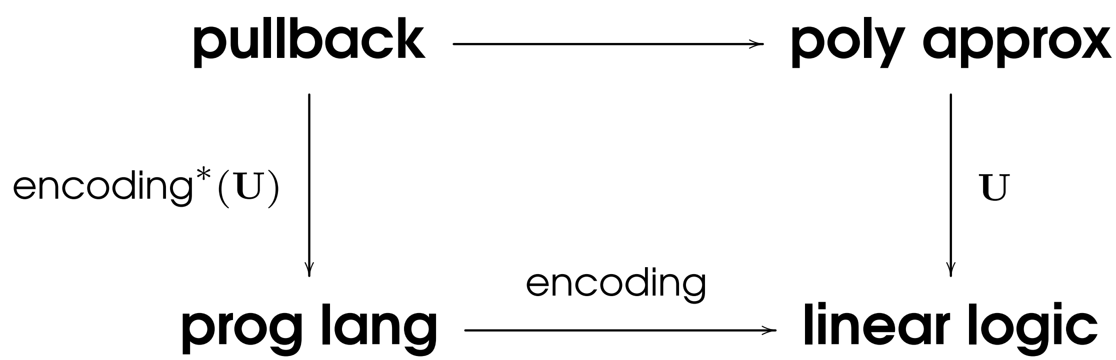
Intersection Types and Runtime Errors in the π-Calculus

Ugo Dal Lago U Bologna / Inria
Marc de Visme ENS Lyon
Damiano Mazza CNRS / U Paris 13
Akira Yoshimizu Inria

MRG, Imperial College London
3 December 2020

Background

- Intersection types (Coppo, Dezani 1980): statically capture dynamics. Challenge: transport ITS technology to concurrent programs.
- Melliès, Zeilberger (POPL 2015): a type system is a functor.
- M., Pellissier, Vial (POPL 2018): ITSs by change of base (pullback)



The point of these observations is not the reduction of the familiar to the unfamiliar [. . .] but the extension of the familiar to cover many more cases. — Saunders MacLane

The π-calculus (polyadic, asynchronous, hyperlocalized)

- Processes and reduction:

$$P, Q ::= \mathbf{0} \;\Big|\; P \mid Q \;\Big|\; \nu x P \;\Big|\; \overline{x}\langle\widetilde{y}\rangle \;\Big|\; \underbrace{x(\widetilde{y}).P \;\Big|\; !x(\widetilde{y}).P}_{\text{no inputs on } \mathsf{fn}(P)}$$

$$\begin{array}{rcll}
\overline{x}\langle\widetilde{y}\rangle \mid x(\widetilde{z}).P & \longrightarrow & P\{\widetilde{y}/\widetilde{z}\} & |\widetilde{y}| = |\widetilde{z}| \\
\overline{x}\langle\widetilde{y}\rangle \mid !x(\widetilde{z}).P & \longrightarrow & P\{\widetilde{y}/\widetilde{z}\} \mid !x(\widetilde{z}).P & |\widetilde{y}| = |\widetilde{z}| \\
\nu x(!x(\widetilde{y}_1).P_1 \mid \cdots \mid !x(\widetilde{y}_n).P_n) & \longrightarrow & \mathbf{0} &
\end{array}$$

- Encodable in (differential) linear logic!

(Honda, Laurent 2010; Ehrhard, Laurent 2010; de Visme, M. 2017)

Expressiveness

- Non-determinism: $\nu x(\overline{x} \mid x.\overline{a} \mid x.\overline{b})$
- Non-deterministic $\lambda\mu$-calculus embeds in $\mathrm{AHL}\pi$.
- Locks:

$$L := !a(z).\nu v(\overline{p}\langle v\rangle \mid v.\overline{z}\langle z\rangle) \qquad \mathsf{Lock} := \nu a(\overline{a}\langle a\rangle \mid L)$$

$$P_i := p(v).Q_i \quad \text{s.t.} \quad Q_i \longrightarrow^* R_i \mid \overline{v},\ v \notin \mathsf{fn}(R_i).$$

We have

$$\begin{aligned} P_1 \mid \cdots \mid P_n \mid \mathsf{Lock} &\longrightarrow^* \ \nu v(P_1 \mid \cdots \mid Q_i \mid \cdots \mid P_n \mid \nu a(v.\overline{a}\langle a\rangle \mid L)) \\ &\longrightarrow^* \ P_1 \mid \cdots \mid R_i \mid \cdots \mid P_n \mid \mathsf{Lock} \end{aligned}$$

Runtime errors and good behavior

- Runtime errors (by example):
  - Arity mismatch: $\overline{x}\langle a, b\rangle \mid x(y).P$
  - Failed send: $\nu x\, \overline{x}\langle a\rangle$
  - Endless wait: $\nu x\, x(y).P$
  - Dependency cycle: $\nu(x, y)(x.\overline{y} \mid y.\overline{x})$

  Definition. A closed process P is well-behaved if, for all $P \longrightarrow^* Q$, Q has no runtime error and $Q \longrightarrow^* \mathbf{0}$.

- Good behavior is more complex than WN or SN (both Σ^0_1):

  Proposition. The set of well-behaved processes is Π^0_2-complete!

Rudimentary types

Types: $A, B, C ::= (A_1, \ldots, A_n)$ Judgments: inp $\vdash P$ :: out

$$\frac{}{\Gamma \vdash \mathbf{0} :: \Delta} \qquad \frac{\Gamma \vdash P :: \Delta \quad \Gamma \vdash Q :: \Delta}{\Gamma \vdash P \mid Q :: \Delta} \qquad \frac{\Gamma, x : A \vdash P :: \Delta, x : A}{\Gamma \vdash \nu x\, P :: \Delta}$$

$$\frac{}{\Gamma \vdash \overline{x}\langle y_1, \ldots, y_n \rangle :: x : (A_1, \ldots, A_n), y_1 : A_1, \ldots, y_n : A_n, \Delta}$$

$$\frac{\vdash P :: y_1 : A_1, \ldots, y_n : A_n, \Delta}{\Gamma, x : (A_1, \ldots, A_n) \vdash \dagger x(y_1, \ldots, y_n).P :: \Delta}$$

Theorem. P typable, $P \longrightarrow^* Q$ implies Q has no arity mismatch.

Simple types

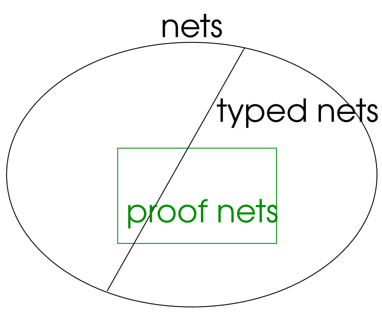
- Types: as before. Typing judgments:

$$\Gamma \vdash P :: \Delta$$

  - Γ = input declarations: $(x; \widetilde{y}) : A$,
    $\widetilde{y}$ = output dependencies;
  - Δ = output declarations: $x : A$

- Example: y depends on x in $x.\overline{y}$. May track dependency cycles.

- Enforced by linear logic correctness:



Simple types: the rules

$$\frac{\widetilde{(x;\widetilde{y})} : \Gamma \vdash P :: \Delta \quad \widetilde{(x;\widetilde{z})} : \Gamma \vdash Q :: \Delta}{\widetilde{(x;\widetilde{y},\widetilde{z})} : \Gamma \vdash P \mid Q :: \Delta}$$

$$\frac{\Gamma, (x;\widetilde{y}) : A, \ldots (w;\widetilde{z}, x) : B \ldots \vdash P :: \Delta, x : A}{\Gamma, \ldots (w;\widetilde{z},\widetilde{y}) : B \ldots \vdash \nu x\, P :: \Delta} \; x \notin \widetilde{y}$$

$$\frac{\vdash P :: y_1 : A_1, \ldots, y_n : A_n, \widetilde{z} : \Delta}{\Gamma, (x;\widetilde{z}) : (A_1, \ldots, A_n) \vdash \dagger x(y_1, \ldots, y_n).P :: \Delta}$$

The rules for $\mathbf{0}$ and output are the same as in rudimentary types, with empty dependencies.

Theorem. P typable, $P \longrightarrow^* Q$ implies that Q has no arity mismatch and no dependency cycle.

Intersection types

- Pre-types:

$$A, B, C ::= \Theta_1 \wedge \cdots \wedge \Theta_k \qquad \text{pre-types}$$

$$\Theta, \Xi ::= * \;\Big|\; (A_1, \ldots, A_n) \qquad \text{sequences}$$

- Types = uniform pre-types, *i.e.*, such that $A \frown A$:

$$\frac{\forall\,\Theta}{* \frown \Theta} \qquad \frac{A_i \frown B_i \quad \forall i \in \{1, \ldots, n\}}{(A_1, \ldots, A_n) \frown (B_1, \ldots, B_n)}$$

$$\frac{\Theta_i \frown \Xi_j \quad \forall i \in \{1, \ldots, k\},\ \forall j \in \{1, \ldots, p\}}{\Theta_1 \wedge \cdots \wedge \Theta_k \frown \Xi_1 \wedge \cdots \wedge \Xi_p}$$

The meaning of types

- Sequences are about arity of channels:

$$\vdash \overline{x}\langle y, z\rangle :: x : (A, B), y : A, z : B$$

- Intersections (non-idempotent!) are about potential use:

$$\vdash \overline{x}\langle y\rangle \mid \overline{x}\langle z\rangle :: x : (A) \wedge (B), y : A, z : B$$

- So we may read back usage information:

$$\Gamma, (x; y) : \boxed{(()) \wedge (\top)} \vdash P :: \Delta \quad \Longrightarrow$$

$\exists$ execution of P s.t. x used twice as unary input, once receiving a name used once for nullary output, once receiving a name which is not used. These receptions unlock sending on y.

The typing rules (by example)

$$\frac{}{\widetilde{w} : \top \vdash \mathbf{0} :: \widetilde{z} : \top}\ \text{zero}$$

$$\frac{}{\vdash \overline{x}\langle x, y\rangle :: x : (A, B) \wedge A, y : B}\ \text{out}$$

$$\frac{\vdash P :: y : A, \Delta}{(x; \mathsf{out}(P) \setminus y) : (A) \vdash x(y).P :: \Delta}\ \text{in}$$

$$\frac{\vdash P :: y : A_1 \quad \ldots \quad \vdash P :: y : A_k}{(x;\,) : (A_1) \wedge \cdots \wedge (A_k) \vdash !x(y).P ::}\ \text{!in}$$

$$\frac{(x; \widetilde{z}) : A \vdash P :: y : C \qquad (x; \widetilde{w}) : B \vdash Q :: y : D}{(x; \widetilde{z}, \widetilde{w}) : A \wedge B \vdash P \mid Q :: y : C \wedge D}\ \text{par}$$

$$\frac{}{(x; z) : * \vdash x.\overline{z} :: z : \top}\ \text{in}_*$$

The rule for restriction is the same as in simple types.

Capturing good behavior

Theorem. A closed process P is typable iff $P \longrightarrow^* \mathbf{0}$.

- A type derivation $\delta :: P$ talks about one possible behavior of P.
- Given $\delta :: P$ and $\rho : P \longrightarrow^* Q$, we may define $\delta \bowtie \rho$ (*matching*).

Definition. A process P is completely typable if

$$\forall \rho : P \longrightarrow^* Q,\ \exists \delta :: P \text{ s.t. } \delta \bowtie \rho.$$

Theorem. Completely typable = well-behaved.

Discussion and perspectives

- Π^0_2-completeness is an insurmountable obstacle: typing is Σ^0_1. Our type system does the best one may hope for.

- There are cases however in which all type derivations for a process are captured by a "parametric derivation" (see the paper).

- This suggests incorporating parameters in derivations, in the style of dependent linear PCF (Dal Lago, Gaboardi 2010), so that one derivation captures every behavior of a process.

- Linear approximations for the π-calculus?

Linear approximations

$$\Xi \vdash p \sqsubset P; \Upsilon \qquad \text{where } \Xi, \Upsilon = \ldots a \sqsubset x \ldots \quad \text{with } a \text{ linear}$$

$$\frac{}{\vdash \mathbf{0} \sqsubset \mathbf{0}}$$

(this is an example of the general rule)

$$\frac{}{\vdash \overline{a}\langle b_1, b_2\rangle \sqsubset \overline{x}\langle y, y\rangle; b_1 \sqsubset y, b_2 \sqsubset y}$$

$$\frac{\vdash p \sqsubset P; \widetilde{\mathbf{b}} \sqsubset \widetilde{y}, \Upsilon}{a \sqsubset x \vdash a(\widetilde{\mathbf{b}}).p \sqsubset x(\widetilde{y}).P; \Upsilon}$$

$$\frac{\vdash p_1 \sqsubset P; \Upsilon_1, \widetilde{\mathbf{b}}_1 \sqsubset \widetilde{y} \quad \ldots \quad \vdash p_n \sqsubset P; \Upsilon_n, \widetilde{\mathbf{b}}_n \sqsubset \widetilde{y}}{a_1 \sqsubset x, \ldots, a_n \sqsubset x \vdash a_1(\widetilde{\mathbf{b}}_1).p_1 \mid \cdots \mid a_n(\widetilde{\mathbf{b}}_n).p_n \sqsubset !x(\widetilde{y}).P; \Upsilon_1, \ldots, \Upsilon_n}$$

$$\frac{\Xi_1 \vdash p \sqsubset P; \Upsilon_1 \quad \Xi_2 \vdash q \sqsubset Q; \Upsilon_2}{\Xi_1, \Xi_2 \vdash p \mid q \sqsubset P \mid Q; \Upsilon_1, \Upsilon_2}$$

$$\frac{\Xi, \mathbf{a}^- \sqsubset x \vdash p \sqsubset P; \Upsilon, \mathbf{a}^+ \sqsubset x}{\Xi \vdash \nu(\mathbf{a}^-, \mathbf{a}^+)\, p \sqsubset \nu x\, P; \Upsilon}$$

$\mathbf{a} \sqsubset x$ means $a_1 \sqsubset x, \ldots, a_n \sqsubset x$ for some $n \geq 0$.
$\widetilde{\mathbf{a}} \sqsubset \widetilde{x}$ means $\mathbf{a}_1 \sqsubset x_1, \ldots, \mathbf{a}_n \sqsubset x_n$ for some $n \geq 0$.