Continuations, mobile processes, all the things...

Julien Gabet

Mars-June, 2018

Memo/garbage part

$M, N ::= x; \lambda x.M; MN$
$(\lambda x.M)N \to_\beta M[N/x]$ terms tend to get bigger

If $C[\ ]$ is a context, and $M \to_\beta N$
then $C[M] \to_\beta C[N]$

$P, Q ::= u(xy).P; \bar{u}xy.P; P|Q; (\nu x)P|!P$
$u(xy).P|\bar{u}ab.Q \to P[a/x, b/y]|Q$

If $P \to Q$ then $C[P] \to C[Q]$ (with necessary hypothesis on context C)
If $P \equiv P' \to Q \equiv Q'$ then $P \to Q$

Krivine Abstract Machine (KAM)
$M \star \Pi \star \mathcal{E}$

$$\begin{aligned} MN \star \Pi \star \mathcal{E} &\to M \star (N, \mathcal{E}).\Pi \star \mathcal{E} \\ \lambda x.M \star (N, \mathcal{E}).\Pi \star \mathcal{F} &\to M \star \Pi \star \mathcal{F}, s \mapsto (N, \mathcal{E}) \\ x \star \Pi \star \mathcal{E}, x \mapsto (M, \mathcal{F}) &\to M \star \Pi \star \mathcal{F} \end{aligned}$$

For exponentials:
$!P \simeq !P|!P$
$(\nu u)!u(x).P \simeq 0$
idea: $!P|Q \simeq !P|!P|Q \ \ \forall Q$

$$\begin{aligned} [\![(M, \mathcal{E}).\Pi]\!]_u &= (\nu m)(\nu v)(\bar{u}mv|!m(x)[\![M, \mathcal{E}]\!]_x|[\![\Pi]\!]_v) \\ [\![M, (x_i \mapsto (M_i, \mathcal{E}_i))_{i=1..k}]\!]_u &= (\nu x_1) \dots (\nu x_k)([\![M]\!]_u|!x_1(u).[\![M_1, \mathcal{E}_1]\!]_u| \dots) \\ [\![MN]\!]_u &= (\nu v)(\nu n)([\![M]\!]_v|\bar{v}nu|!n(x).[\![N]\!]_x) \\ [\![\lambda x.M]\!]_u &= u(xv).[\![M]\!]_v \\ [\![x]\!]_u &= \bar{x}u \end{aligned}$$

We want $M \star \Pi \star \mathcal{E} \to M' \star \Pi' \star \mathcal{E}'$ iff $[\![M, \mathcal{E}]\!]_u|[\![\Pi]\!]_u \to [\![M', \mathcal{E}']\!]_v|[\![\Pi']\!]_v$

- equiv $\simeq$ bisimulation
- the traduction goes well

Definition

A binary relation S is a reduction bisimulation if, forall $(P, Q) \in S$

(1) $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ for some Q' with $(P', Q') \in S$

(2) $Q \xrightarrow{\tau} Q'$ implies $P \xrightarrow{\tau} P'$ for some P' with $(P', Q') \in S$

Definition (*Observability:*)

$P \downarrow_x$ if P can make an input action of subject x
$P \downarrow_{\bar{x}}$ if P can make an output action of subject x.

Definition (*Image-finite process:*)

P is image-finite if, for all derivative Q of P and any action $\alpha, \exists n \geq 0$ and $Q_1, \ldots Q_n$ such that $Q \stackrel{\alpha}{\Rightarrow} Q'$ implies $Q' = Q_i$ for some i.
where $\Rightarrow$ is the reflexive transitive closure of $\xrightarrow{\tau}$ and $\stackrel{\alpha}{\Rightarrow}$ is $\Rightarrow\xrightarrow{\alpha}\Rightarrow$ for some action α.

Rules for base-π

Value-typing

$$\frac{}{\Gamma \vdash basval : B}\ \text{TV-BASVAL} \qquad \frac{}{\Gamma, x : T \vdash x : T}\ \text{TV-NAME}$$

Process typing

$$\frac{\Gamma \vdash P : \diamondsuit \quad \Gamma \vdash Q : \diamondsuit}{\Gamma \vdash P|Q : \diamondsuit}\ \text{T-PAR} \qquad \frac{\Gamma \vdash P : \diamondsuit \quad \Gamma \vdash Q : \diamondsuit}{\Gamma \vdash P + Q : \diamondsuit}\ \text{T-SUM}$$

$$\frac{\Gamma \vdash v : \sharp T \quad \Gamma \vdash w : \sharp T \quad \Gamma \vdash P : \diamondsuit}{\Gamma \vdash [v = w]P : \diamondsuit}\ \text{T-MAT} \qquad \frac{}{\Gamma \vdash 0 : \diamondsuit}\ \text{T-NIL}$$

$$\frac{\Gamma \vdash P : \diamondsuit}{\Gamma \vdash !P : \diamondsuit}\ \text{T-REP} \qquad \frac{\Gamma, x : L \vdash P : \diamondsuit}{\Gamma \vdash (\nu x : L)P : \diamondsuit}\ \text{T-RES} \qquad \frac{\Gamma \vdash P : \diamondsuit}{\Gamma \vdash \tau.P : \diamondsuit}\ \text{T-TAU}$$

$$\frac{\Gamma \vdash v : \sharp T \quad \Gamma, x : T \vdash P : \diamondsuit}{\Gamma \vdash v(x).P : \diamondsuit}\ \text{T-INP} \qquad \frac{\Gamma \vdash v : \sharp T \quad \Gamma \vdash w : T \quad \Gamma \vdash P : \diamondsuit}{\Gamma \vdash \bar{v}w.P : \diamondsuit}\ \text{T-OUT}$$

Types: $S, T ::= V$ value type
$\quad | L$ link type
$\quad | \diamond$ behaviour type

Value types: $V ::= B$ basic type
Link types: $L ::= \sharp V$ connexion type
Environments: $\Gamma ::= \Gamma, x : L \mid \Gamma, x : V \mid \emptyset$

Transitions for base-π

$$\frac{}{\bar{a}w.P \xrightarrow{\bar{a}w} P}\ \text{OUT} \qquad \frac{}{a(x).P \xrightarrow{aw} P\{w/x\}}\ \text{INP} \qquad \frac{}{\tau P \xrightarrow{\tau} P}\ \text{TAU}$$

$$\frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}\ \text{MAT} \qquad \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}\ \text{SUM-L} \qquad \frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ \text{PAR-L } (bn(\alpha) \cap fn(Q) = \emptyset)$$

$$\frac{P \xrightarrow{(\nu \widetilde{z}:\widetilde{T})\bar{a}v} P' \quad Q \xrightarrow{av} Q'}{P|Q \xrightarrow{\tau} (\nu \widetilde{z} : \widetilde{T})(P'|Q')}\ \text{COMM-L } (\widetilde{z} \cap fn(Q) = \emptyset)$$

$$\frac{P \xrightarrow{\alpha} P'}{(\nu x : T)P \xrightarrow{\alpha} (\nu x : T)P'}\ \text{RES } (x \notin n(\alpha)) \qquad \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'|!P}\ \text{REP-ACT}$$

$$\frac{P \xrightarrow{(\nu \widetilde{z}:\widetilde{T})\bar{a}v} P'}{(\nu x : T)P \xrightarrow{(\nu \widetilde{z}:\widetilde{T}, x:T)} P'}\ \text{OPEN } (x \in fn(v), x \notin \{\widetilde{z}, a\})$$

$$\frac{P \xrightarrow{(\nu \widetilde{z}:\widetilde{T})\bar{a}v} P' \quad P \xrightarrow{av} P''}{!P \xrightarrow{\tau} (\nu \widetilde{z} : \widetilde{T})(P'|P'')|!P}\ \text{REP-COMM } (\widetilde{z} \cap fn(P) = \emptyset)$$

$$\frac{\text{Si } v \text{ n'est pas un nom}}{\bar{v}w.P \xrightarrow{\tau} wrong}\ \text{OUTERR} \qquad \frac{\text{Si } v \text{ n'est pas un nom}}{v(x).P \xrightarrow{\tau} wrong}\ \text{INPERR}$$

$$\frac{\text{Si } v \text{ ou } w \text{ n'est pas un nom}}{[v = w]P \xrightarrow{\tau} wrong}\ \text{MATERR}$$

simply-typed π-calculus: same but with value types $V ::= B$ basic type
$\quad | L$ link type, allowing to pass links

i/o types

Grammar: same + $L ::= iV \mid oV$ (input and output capabilities)

Subtyping rules

$$\frac{}{T \leq T}\ \text{SUB-REFL} \qquad \frac{S \leq S' \quad S' \leq T}{S \leq T}\ \text{SUB-TRANS}$$

$$\frac{}{\sharp T \leq iT}\ \text{SUB-}\sharp\text{I} \qquad \frac{}{\sharp T \leq oT}\ \text{SUB-}\sharp\text{O}$$

$$\frac{S \leq T}{iS \leq iT}\ \text{SUB-II} \qquad \frac{S \leq T}{oT \leq oS}\ \text{SUB-OO}$$

$$\frac{S \leq T \quad T \leq S}{\sharp T \leq \sharp S}\ \text{SUB-BS}$$

Typing rules

$$\frac{\Gamma \vdash a : iS \quad \Gamma, x : S \vdash P : \diamondsuit}{\Gamma \vdash a(x).P : \diamondsuit}\ \text{T-INPS}$$ replaces T-INP

$$\frac{\Gamma \vdash a : oT \quad \Gamma \vdash w : T \quad \Gamma \vdash P : \diamondsuit}{\Gamma \vdash \bar{a}w.P : \diamondsuit}\ \text{T-OUTS}$$ replaces T-OUT

$$\frac{\Gamma \vdash v : S \quad S \leq T}{\Gamma \vdash v : T}\ \text{SUBSUMPTION}$$

Linear types

Grammar: $L ::= l_\sharp V \mid l_i V \mid l_o V$

Combination of types

$l_i T \uplus l_o T = l_\sharp T$

$T \uplus T = T$ if T is non-linear

$T \uplus U = error$ otherwise

Combination of environments

If for some $x, \Gamma_1(x)$ and $\Gamma_2(x)$ are defined and $\Gamma_1(x) \uplus \Gamma_2(x) = error$ then $\Gamma_1 \uplus \Gamma_2$ is undefined.

Otherwise, $(\Gamma_1 \uplus \Gamma_2)(x) = \begin{cases} \Gamma_1(x) \uplus \Gamma_2(x) \text{ if both are defined} \\ \Gamma_i(x) \text{ if defined but } \Gamma_{3-i}(x) \text{ is not defined} \\ \text{undefined otherwise} \end{cases}$

Extraction

$Lin(\Gamma) = \{x | \Gamma(x) = l_I T \text{ for } I \in \{i, o, \sharp\} \text{ and some type } T\}$

$Lin_i(\Gamma) = \{x | \Gamma(x) = l_i S \text{ or } \Gamma(x) = l_\sharp S \text{ for some } S\}$

Value-typing

$$\frac{}{\Gamma, x : T \vdash x : T}\ \text{LIN-NAME } (Lin(\Gamma) = \emptyset) \qquad \frac{}{\Gamma \vdash \star : unit}\ \text{LIN-UNIT } (Lin(\Gamma) = \emptyset)$$

\+ SUBSUMPTION and subtyping rules

Process typing

$$\frac{\Gamma_1 \vdash v : mS \quad (m \in \{i, l_i\}) \qquad \Gamma_2, x : S \vdash P}{\Gamma_1 \uplus \Gamma_2 \vdash v(x).P : \diamondsuit}\ \text{LIN-INP}$$

$$\frac{\Gamma_1 \vdash v : mS \quad (m \in \{o, l_o\}) \qquad \Gamma_2 \vdash w : S \qquad \Gamma_3 \vdash P : \diamondsuit}{\Gamma_1 \uplus \Gamma_2 \uplus \Gamma_3 \vdash \bar{v}w.P : \diamondsuit}\ \text{LIN-OUT}$$

$$\frac{\Gamma_1 \vdash P_1 : \diamondsuit \qquad \Gamma_2 \vdash P_2 : \diamondsuit}{\Gamma_1 \uplus \Gamma_2 \vdash P_1 | P_2 : \diamondsuit}\ \text{LIN-PAR} \qquad \frac{\Gamma \vdash P_1 : \diamondsuit \qquad \Gamma \vdash P_2 : \diamondsuit}{\Gamma \vdash P_1 + P_2 : \diamondsuit}\ \text{LIN-SUM}$$

$$\frac{\Gamma \vdash P : \diamondsuit}{\Gamma \vdash \tau.P : \diamondsuit}\ \text{LIN-TAU} \qquad \frac{\Gamma \vdash P : \diamondsuit}{\Gamma \vdash !P : \diamondsuit}\ \text{LIN-REP } (Lin(\Gamma) = \emptyset)$$

$$\frac{}{\Gamma \vdash 0 : \diamondsuit}\ \text{LIN-NIL } (Lin(\Gamma) = \emptyset) \qquad \frac{\Gamma_1 \vdash v : \sharp T \qquad \Gamma_1 \vdash w : \sharp T \qquad \Gamma_2 \vdash P : \diamondsuit}{\Gamma_1 \uplus \Gamma_2 \vdash [v = w]P : \diamondsuit}\ \text{LIN-MAT}$$

$$\frac{\Gamma, x : L \vdash P : \diamondsuit}{\Gamma \vdash (\nu x : L)P : \diamondsuit}\ \text{LIN-RES} \qquad \frac{\Gamma \vdash P : \diamondsuit}{\Gamma \vdash (\nu x : L)P : \diamondsuit}\ \text{LIN-RES2}$$

TODO: results on i/o and i/o-lin

We take the following rules over simplified π-calculus with only parallelization:

$$\frac{}{\bot \vdash 0} \qquad \frac{E \vdash P \qquad F \vdash Q}{E \parr F \vdash P|Q} \qquad \frac{}{\alpha \vdash A}$$

along with subsumption written as such $\dfrac{E \leq F \qquad F \vdash P}{E \vdash P}$

where $E \leq F \iff \vdash E^{\bot}, F$ in MLL.
The goal is to completely write the elimination of subsumption in this system (in order to generalize to more expressive ones later).

Remark: the definition of $\leq$ implies one can dualize the connectors (*ie.* turn $\parr$s in $\otimes$s), given the addition of enough neutrals in the formula. This is not bad per se, but can lead to unfriendly types that don't reflect the way things behave.
For example, under the assumption that $\vdash P_i^{\bot}$ for each $i \in \{1, \ldots, n\}$, one can prove:
$(P_1 \parr 1 \parr \ldots \parr P_n \parr 1) \otimes \alpha_1 \otimes \ldots \otimes \alpha_n \leq \alpha_1 \parr \ldots \parr \alpha_n$
One way to mitigate this kind of behaviour would be to annotate the parallel rule with some variable, and to allow for context in the typing rules:

$$\frac{}{\Gamma, \bot \vdash 0} \qquad \frac{\Gamma, x : E \vdash P \qquad \Delta, x : F \vdash Q}{\Gamma, \Delta, E \parr_x F \vdash P|_x Q} \qquad \frac{}{\Gamma, \alpha \vdash A}$$

Contextualization allows for parts we don't deal with when typing parallel behaviours, but it needs to be carefully handled with the subsumption and subtyping in general. Giving contexts to the axiom rules (both null behaviour and atoms) allows the "trash" parts of subtyping to be moved up to the top of the tree (and thus eliminate subsumptions under axioms), but dealing with parallel is an other story...
Specifically, we need to be able to type the following:
$(P_1 \parr 1 \parr \ldots \parr P_n \parr 1) \otimes \alpha_1 \otimes \ldots \otimes \alpha_n \vdash A_1| \ldots |A_n$
Let's add a tensor rule that does "nothing" on the π-terms to deal with the $\alpha_1 \otimes \ldots \otimes \alpha_n$ part:

$$\frac{\Gamma, E, F \vdash P}{\Gamma, E \otimes F \vdash P}$$

This rule might need some more syntax though (such as annotations), as it mostly reflects the non-deterministic part of the behaviour of the π-term that is given a type.
Let's then transform our example by this rule and the context usage:
$P_1 \parr 1 \parr \ldots \parr P_n \parr 1, \alpha_1 \otimes \ldots \otimes \alpha_n \vdash A_1| \ldots |A_n$
This is much more appealing. We can now annotate on the $P_i \parr 1$ types and use them for parallelizing, and get the α_i together with the tensor rule.

What we get here by pushing the sub rules upwards is atoms of the form $\dfrac{}{\alpha_i, P_i \parr 1 \vdash A_i}$ and we use the parallelization rule and the tensor rule to put them together like so:

$$\frac{\dfrac{\alpha_i, x : (P_i \parr 1) \vdash A_i \qquad \alpha_j, x : (P_j \parr 1) \vdash A_j}{\alpha_i, \alpha_j, (P_i \parr 1) \parr_x (P_j \parr 1) \vdash A_i|_x A_j}}{\alpha_i \otimes \alpha_j, (P_i \parr 1) \parr_x (P_j \parr 1) \vdash A_i|_x A_j}$$