Geometry of Interaction and Principal Types

Furio Honsell
(20/8/58-14/9/2044 (14/8/2040))
University of Udine (leave of absence)
elected Member of the Assembly of the
Friuli Venezia Giulia Autonomous Region (ITALY)

joint work with Marina Lenisa and Ivan Scagnetto

IFIP WG 2.2 Bologna 6-8/9/7E7, (9+8)x7x(6x5-4x3-2+1)

Mathematics and Politics: the issue of **inequity**

In 2022, the Region FVG has had a significant increase in fiscal revenues. Last December we had over 750 million Euros more than expected. How should we use them?

Wait a minute! How much of this comes from VAT, which is an indirect tax, and hence it has an inversely proportional impact on household incomes, therefore unconstitutional, *strictu sensu*?
Actually, how much of this comes just from inflation?
I asked the question and computed the answer: 150 million Euros!
MORAL: the cost of bread increases because of inflation, the government gets richer?
How should the Region use this extra profit?

A little history and my IFIP WG 2.2. CV

- “Formal Language description Languages” 1966
  - Corrado Böhm, CUCH Machine,
  - LCF, ISWIM, OWHY Dana Scott,
  - Peter Landin, The Next 700 Programming Languages
  - ...
- first invitation - **Palo Alto 1989**
- member - **Hamilton (CA) 1993**
- local organizer IFIP WG 2.2- **Udine 1999** and **Udine 2006**

A question at the Semantics of Data Types Symposium, Sophia-Antipolis, June 27-29, 1984.

- I asked D.Scott, G.Plotkin, etc. :"**Are any two λ-terms equated in all Scott Domain Models β-convertible**?"
- The *Completeness* issue: let $\mathcal{C}$ be a class of models of $\mathcal{T}$; and let $\mathcal{F}$ be a set of sentences, $\mathcal{C}$ is $\mathcal{F}$-complete if

$$\forall \phi \in \mathcal{F}.(\forall \mathcal{M} \in \mathcal{C}.\mathcal{M} \models \phi) \implies (\forall \mathcal{M} \models \mathcal{T}.\mathcal{M} \models \phi)$$

- The reverse issue is $\mathcal{F}$-consistency of $\mathcal{C}$.
- Inconsistency has been gradually settled in the negative: F.Honsell, S.Ronchi (quantifier-free 1992), P.Selinger (2003), F.H. G.Plotkin ($\Pi_2(Eq)$, 2009), A.Salibra, A.Carraro (equational 2013, using unsolvable terms);
- $\Pi_1(POS)$-completeness holds (F.H.-G.P. 2009) $\Sigma_1(Eq)$-completeness fails (P.Selinger 2001);
- **Equational completeness, original question, is still open!** (YES, for ω_1-continuous functions in NJC1992 P.Di Gianantonio-F.H.-G.P. .)

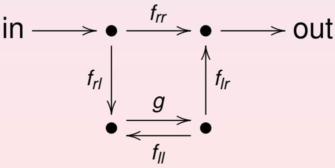
Girard's "Geometry of Interaction" Semantics

- Linear Logic: make multiple occurrences of variables explicit;
- decompose application $[\![MN]\!] = [\![M]\!] \bullet_{linear} !([\![N]\!])$;
- introduce two abstractions $\lambda x.x$ and $\lambda !x.xx$ and pattern matching reduction,
  i.e. $(\lambda !x.M)N$ is stuck but $(\lambda !x.M)!N \to M[N/x]$

Many *game (interaction) models* were developed since the early '90's.
Equational Completeness is even more problematic, in TLCA 1999 P.D.G.- G.Franco - F.H. essentially only one theory is modeled.

Girard's GoI - Abramsky's version in MSCS (2002)

- derived from approach using *traced monoidal categories*;
- T_Σ, the language of *moves*, is defined by the signature $\Sigma_0 = \{\epsilon\} \cup Vars$, $\Sigma_1 = \{\mathrm{l},\mathrm{r}\}$, $\Sigma_2 = \{\langle\ ,\ \rangle\}$;
  $r(t)$ are *output words*, terms $l(t)$ are *input words* ;
- $\mathcal{I}$ is the set of *strategies i.e. partial involutions* over T_Σ: partial functions $f : T_\Sigma \rightharpoonup T_\Sigma$ such that
  $$f(t) = t' \Leftrightarrow f(t') = t;$$
- the operation of *replication* is defined by
  $!f = \{(\langle t, t_1\rangle, \langle t, t_2\rangle) \mid t \in T_\Sigma \wedge (t_1, t_2) \in f\}$;
- the notion of *linear application* is defined by
  $f \cdot g = f_{rr} \cup (f_{rl}; g; (f_{ll}; g)^*; f_{lr})$, where
  $f_{ij} = \{(t_1, t_2) \mid (i(t_1), j(t_2)) \in f\}$, for $i, j \in \{r, l\}$



Models: Affine, Light, Elementary, Linear λ-Calculi via Combinators

- **affine** λ^A: no !-abstraction no multiple occurrences of vars; the calculus terminates in linear time and models Grzegorczyk $\mathcal{E}^1$.

$$(\mathbf{B})_\lambda = \lambda xyz.x(yz) \quad (\mathbf{C})_\lambda = \lambda xyz.(xz)y \quad (\mathbf{I})_\lambda = \lambda x.x \quad (\mathbf{K})_\lambda = \lambda xy.x$$

- **light** λ^L … terminates in polynomial time and models $\mathcal{E}^2$;
- **elementary** λ^E: variables !-abstracted only if they occur in the scope of a single !; terminates in elementary time and models $\mathcal{E}^3$

$$(\mathbf{W})_\lambda = \lambda x!y.x!y!y \qquad (\mathbf{F})_\lambda = \lambda!x!y.!(xy)$$

- **full** $\lambda^!$: no restrictions on !-abstractions

$$(\mathbf{D})_\lambda = \lambda!x.x \qquad (\delta)_\lambda = \lambda!x.!!x$$

The interpretation of Combinators in $\mathcal{I}$

$$
\begin{array}{lcl}
[\![\mathbf{B}]\!] & = & \{r^3x \leftrightarrow lrx\ ,\ l^2x \leftrightarrow rlrx\ ,\ rl^2x \leftrightarrow r^2lx\} \\
[\![\mathbf{I}]\!] & = & \{lx \leftrightarrow rx\} \\
[\![\mathbf{C}]\!] & = & \{l^2x \leftrightarrow r^2lx\ ,\ lrlx \leftrightarrow rlx\ ,\ lr^2x \leftrightarrow r^3x\} \\
[\![\mathbf{K}]\!] & = & \{lx \leftrightarrow r^2x\} \\
[\![\mathbf{F}]\!] & = & \{l\langle i, rx\rangle \leftrightarrow r^2\langle i, x\rangle\ ,\ l\langle i, lx\rangle \leftrightarrow rl\langle i, x\rangle\} \\
[\![\mathbf{W}]\!] & = & \{r^2x \leftrightarrow lr^2x\ ,\ l^2\langle i, x\rangle \leftrightarrow rl\langle li, x\rangle\ ,\ lrl\langle i, x\rangle \leftrightarrow rl\langle ri, x\rangle\} \\
[\![\delta]\!] & = & \{l\langle\langle i, j\rangle, x\rangle \leftrightarrow\ r\langle i, \langle j, x\rangle\rangle\} \\
[\![\mathbf{D}]\!] & = & \{l\langle \epsilon, x\rangle \leftrightarrow rx\}
\end{array}
$$

where $\{u_i[x] \leftrightarrow v_i[x] \mid i \in I\}$ denotes the partial involution $\{(u_i[t], v_i[t]) \mid i \in I, t \in T_\Sigma\}$

What is this all about? What are strategies? What are moves? What is G.o.I.? (λ^A-case)

- understanding this is essential for studying the fine structure of models
- **strategies** are **principal types**,
- **moves** are **occurrences of variables** in principal types;
- *Types* T_Σ are binary trees whose leaves are variables $\alpha, \beta, \ldots \in$ *TVar*, and nodes are denoted by $\multimap$, *i.e.*

$$(T_\Sigma \ni)\ \sigma, \tau ::= \ \alpha \mid \beta \mid \ \ldots \ \mid \sigma \multimap \tau \ .$$

- A type σ is *binary* if each variable in σ occurs at most twice.
- *Occurrences* of variables in types are denoted by:

$$(O_\Sigma \ni)\ u[\alpha] ::= [\alpha] \mid lu[\alpha] \mid ru[\alpha]\ ,$$

  - $[\alpha]$ denotes the occurrence of the variable α in the type α,
  - if $u[\alpha]$ denotes an occurrence of α in σ_1 (σ_2), then $lu[\alpha]$ ($ru[\alpha]$) denotes the corresponding occurrence of α in $\sigma_1 \multimap \sigma_2$.

Correspondence between types and partial involutions

- A type τ gives rise to a set of variable occurrences

$$\mathcal{O}(\tau) = \{u[\alpha] \mid u[\alpha] \text{ is an occurrence of } \alpha \text{ in } \tau\}.$$

- A *binary* type τ gives rise to a partial involution on O_Σ

$$\mathcal{R}(\tau) = \{\langle u[\alpha], v[\alpha]\rangle \mid u[\alpha], v[\alpha] \text{ are different occurrences of } \alpha \text{ in } \tau\}$$

- Vice versa, from a set of variable occurrences, such that no path is the initial prefix of any other path of a different occurrence, we can build the tree of a type, by tagging possible missing leaves with fresh variables in $Z = \{\zeta_1, \ldots, \zeta_i, \ldots\}$.
-

$$\mathcal{T}_Z(S) = \begin{cases} \zeta & \text{if } \mathcal{S} = \emptyset \\ \alpha & \text{if } \mathcal{S} = \{[\alpha]\} \\ \mathcal{T}_Z(\{u \mid lu \in \mathcal{S}\}) \multimap \mathcal{T}_Z(\{u \mid ru \in \mathcal{S}\}) & \text{otherwise,} \end{cases}$$

- For all type σ, we have $\mathcal{T}_Z(\mathcal{O}(\sigma)) = \sigma$;

Two Perspectives on Unification - the top-down perspective of types

The perspectve of Types

A unification algorithm *à la* Martelli Montanari: it unifies simultaneously sets of pairs of types.

Let E be a set of pairs of types:

$$
\begin{array}{lcl}
\mathcal{U}(\{\langle \sigma_1 \multimap \sigma_2, \tau_1 \multimap \tau_2 \rangle\} \cup E) & \rightarrow & \mathcal{U}(\{\langle \sigma_1, \tau_1 \rangle, \langle \sigma_2, \tau_2 \rangle\} \cup E) \\
\mathcal{U}(\{\langle \alpha, \alpha \rangle\} \cup E) & \rightarrow & E \\
\mathcal{U}(\{\langle \sigma_1 \multimap \sigma_2, \alpha \rangle\} \cup E) & \rightarrow & \mathcal{U}(\{\langle \alpha, \sigma_1 \multimap \sigma_2 \rangle\} \cup E) \\
\mathcal{U}(\{\langle \alpha, \sigma \rangle\} \cup E) & \rightarrow & \mathcal{U}(\{\langle \alpha, \sigma \rangle\} \cup E[\sigma/\alpha]), \text{ if } \alpha \notin \sigma \wedge \alpha \in Var(E) \\
\mathcal{U}(\{\langle \alpha, \sigma \rangle\} \cup E) & \rightarrow & \text{fail, if } \alpha \in \sigma \ \wedge \ \alpha \neq \sigma
\end{array}
$$

The Bottom-up Perspective of Type-Variable Occurrences

Definition (Occurrence Unifiers)

Let σ, τ be types.
(i) Two occurrences $u[\alpha] \in \sigma$ and $v[\beta] \in \tau$ are *unifiable* if u is a prefix of v, *i.e.* there exists w such that $uw = v$, or vice versa.
(ii) If two occurrences $u[\alpha] \in \sigma$ and $v[\beta] \in \tau$ are *unifiable*, their *occurrence unifier* (occ-unifier) is the most general unifier of $\mathcal{T}_Z(\{u[\alpha]\})$ and $\mathcal{T}_Z(\{v[\beta]\})$.

The Bottom-up Perspective of Type Variable Occurrences

GoI application gives rise to a *variable-occurrence oriented* alternate characterization of unification. (Case of binary types where each variable occurs exactly twice).

Definition (GoI-unification)

Let $\sigma, \tau \in T_\Sigma$ be types. The types σ and τ *GoI-unify* if
(i) for every $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)$ there exists $\langle u'[\gamma], v'[\gamma]\rangle \in \mathcal{R}(\tau)\hat{;}(\mathcal{R}(\sigma)\hat{;}\mathcal{R}(\tau))^*$, such that $uw = u'$ and $vw = v'$, and
(ii) for every $(u[\alpha], v[\alpha]) \in \mathcal{R}(\tau)$ there exists $\langle u'[\gamma], v'[\gamma]\rangle \in \mathcal{R}(\sigma)\hat{;}(\mathcal{R}(\tau)\hat{;}\mathcal{R}(\sigma))^*$, such that $uw = u'$ and $vw = v'$.
I.e.:

$$\mathcal{R}(\tau)\widehat{\subseteq}\mathcal{R}(\sigma)\hat{;}(\mathcal{R}(\tau)\hat{;}\mathcal{R}(\sigma))^* \quad \text{and} \quad \mathcal{R}(\sigma)\widehat{\subseteq}\mathcal{R}(\tau)\hat{;}(\mathcal{R}(\sigma)\hat{;}\mathcal{R}(\tau))^*$$

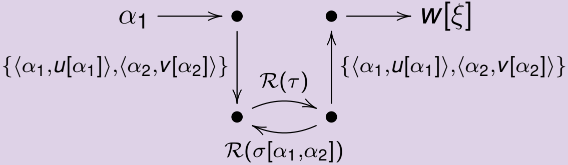
where $\widehat{\subseteq}$ denotes “inclusion up-to substitution”.

Proposition

Let $\sigma, \tau \in T_\Sigma$ be binary types where each variable occurs exactly twice. Then σ, τ unify if and only if σ, τ GoI unify.

Proof.

Let $\sigma[\alpha, \alpha]$ be σ where we have highlighted the two occurrences of a variable α, $u[\alpha]$, $v[\alpha]$, and consider the new type $\sigma[\alpha_1, \alpha_2] \multimap \alpha_1 \multimap \alpha_2$. Now compute $\mathcal{R}(\sigma[\alpha_1, \alpha_2] \multimap \alpha_1 \multimap \alpha_2) \cdot \mathcal{R}(\tau)$. Then σ and τ unify with unifier U if and only if, by Propositions below, for some fresh variable ξ,



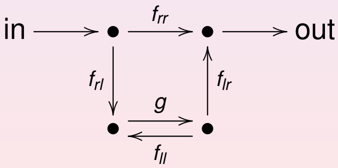
If S is the collection of all such possible outcomes we have $U(\alpha) = \mathcal{T}_Z(S)$, for a suitable set Z of fresh variables. □

GoI using binary types - http://158.110.146.197:31780/automata/

(i) $\mathcal{I}$ is the set of *partial involutions* induced by binary types, *i.e.* $\mathcal{I} = \{\mathcal{R}(\tau) \mid \tau \in T_\Sigma \ \wedge \ \tau \text{ binary}\}$. (ii) The notion of *linear application* is defined, for $f, g \in \mathcal{I}$, by

$$f \cdot g = \ f_{rr} \cup (f_{rl} \,\hat{;}\, g \,\hat{;}\, (f_{ll} \,\hat{;}\, g)^* \,\hat{;}\, f_{lr}) \ ,$$

where $f_{ij} \ = \ \{\langle u, v\rangle \mid \langle i(u), j(v)\rangle \in f\}$, for $i, j \in \{r, l\}$. Variables in different pairs of $f \cdot g$ to be disjoint.
(iii) $\mathcal{O}(f \cdot g) = \{u \mid \exists v. \langle u, v\rangle \in f \cdot g\}$.



Principal Type Assignment System $\Vdash$ for λ^A

$$\frac{}{x : \alpha \Vdash_A x : \alpha} \quad \text{(var)}$$

$$\frac{\Gamma, x : \sigma \Vdash_A M : \tau}{\Gamma \Vdash_A \lambda x.M : \sigma \multimap \tau} \quad \text{(abs)} \qquad \frac{\Gamma \Vdash_A M : \sigma \quad \alpha \text{ fresh}}{\Gamma \Vdash_A \lambda x.M : \alpha \multimap \sigma} \quad (\text{abs}_\emptyset)$$

$$\frac{\begin{array}{cc} \Gamma \Vdash_A M : \sigma & \Delta \Vdash_A N : \tau \\ (dom(\Gamma) \cap dom(\Delta)) = \emptyset & (TVar(\Gamma) \cap TVar(\Delta)) = \emptyset \\ (TVar(\sigma) \cap TVar(\tau)) = \emptyset & U' = \mathcal{U}(\sigma, \alpha \multimap \beta) \\ U = \mathcal{U}(U'(\alpha), \tau) & \alpha, \beta \textit{ fresh} \end{array}}{U \circ U'(\Gamma, \Delta) \Vdash_A MN : U \circ U'(\beta)} \quad \text{(app)}$$

Principal Types for Affine Combinators

$$
\begin{aligned}
[\![\mathbf{I}]\!] &= \alpha \multimap \alpha \\
[\![\mathbf{B}]\!] &= (\alpha \multimap \beta) \multimap (\gamma \multimap \alpha) \multimap \gamma \multimap \beta \\
[\![\mathbf{C}]\!] &= (\alpha \multimap \beta \multimap \gamma) \multimap \beta \multimap \alpha \multimap \gamma \\
[\![\mathbf{K}]\!] &= \alpha \multimap \beta \multimap \alpha
\end{aligned}
$$

Resolution of principal types is G.o.I. application

Proposition

Let $f, g \in \mathcal{I}$. *Then* $\langle u, v\rangle \in f \cdot g$ *if and only if there exists an even sequence,* $\langle u_1[\alpha_1], u'_1[\alpha_1]\rangle, \ldots, \langle u_{n+1}[\alpha_{n+1}], u'_{n+1}[\alpha_{n+1}]\rangle$*:*

- *either* $n = 0$ *and* $\langle u_1[\alpha_1], u'_1[\alpha_1]\rangle \in f_{rr}$ *or* $n > 0$, $\langle u_1[\alpha_1], u'_1[\alpha_1]\rangle \in f_{rl}$, $\langle u_{n+1}[\alpha_{n+1}], u'_{n+1}[\alpha_{n+1}]\rangle \in f_{lr}$, $\langle u_i[\alpha_i], u'_i[\alpha_i]\rangle \in g$, *for* $i < n$, i *even, and* $\langle u_i[\alpha_i], u'_i[\alpha_i]\rangle \in f_{ll}$, *for* $1 < i < n+1$, i *odd;*
- *the set of types* $\Pi = \{\langle \mathcal{T}_Z(u'_i[\alpha_i]), \mathcal{T}_Z(u_{i+1}[\alpha_{i+1}])\rangle \mid 1 \leq i \leq n\}$ *(where* Z*-variables used in different types are different) is unifiable*

The sequence $\langle u_1[\alpha_1], u'_1[\alpha_1]\rangle, \ldots, \langle u_{n+1}[\alpha_{n+1}], u'_{n+1}[\alpha_{n+1}]\rangle$ *is called a trajectory and* $\langle u, v\rangle$ *its output.*

Resolution of principal types is G.o.I. application

Proposition

Let $\sigma_1 \multimap \sigma_2$ and τ be binary types, let Θ be the set of type-variables in $TVar(\sigma_1 \multimap \sigma_2, \tau)$ which are not involved in any trajectory of $\mathcal{R}(\sigma_1 \multimap \sigma_2) \cdot \mathcal{R}(\tau)$, and let $U_{\pi_1}, \ldots, U_{\pi_n}$ be the unifiers arising from all trajectories $\pi_1, \ldots, \pi_n$ of

$$\mathcal{R}(\sigma_1 \multimap \sigma_2) \cdot \mathcal{R}(\tau).$$

If σ_1 and τ are unifiable with m.g.u. $\overline{U}$, then:

1. $U_{\pi_i} \leq \overline{U}$ *for all* i;
2. $\bigoplus_i U_{\pi_i} = \overline{U}_{\restriction (TVar \setminus \Theta)}$;
3. $\mathcal{T}_Z(\mathcal{O}(\mathcal{R}(\sigma_1 \multimap \sigma_2) \cdot \mathcal{R}(\tau))) = \overline{U}(\sigma_2)$.

G.o.I. application is Resolution of (ancestral) types

Proposition

Let $\sigma_1 \multimap \sigma_2$ and τ be binary types such that σ_1 and τ are unifiable with m.g.u. $\overline{U}$, then

$$\mathcal{R}(\sigma_1 \multimap \sigma_2) \cdot \mathcal{R}(\tau) = \mathcal{R}(\overline{U}(\sigma_2)) \ .$$

Theorem (A.C. - F.H.- M.L. - I.S. 2018)

*For any closed term M of affine combinatory logic, $\{$**I**, **B**, **C**, **K** $\}$, we have: $[\![M]\!]^{\mathcal{I}} = \mathcal{R}(\sigma)$, where σ is the principal type of $(M)_\lambda$.*

The above theorem provides a partial answer to a problem raised by Abramsky in *Structural Approach to Reversible Computation (2011)*

Ancestral types

No full resolution can apply to the following two types

$\tau \equiv ((\alpha \multimap \beta) \multimap (\gamma \multimap (\gamma \multimap \delta) \multimap \delta)) \multimap \alpha \multimap \beta$

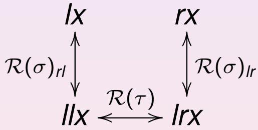
$\sigma \equiv (\alpha \multimap \alpha) \multimap (\gamma \multimap \gamma)$

but,

$\mathcal{R}(\sigma) = \{\mathit{rlx} \leftrightarrow \mathit{lllx}, \mathit{rrx} \leftrightarrow \mathit{llrx}, \mathit{lrlx} \leftrightarrow \mathit{lrrlx}, \mathit{lrrlrx} \leftrightarrow \mathit{lrrrx}\}$

$\mathcal{R}(\tau) = \{\mathit{llx} \leftrightarrow \mathit{lrx}, \mathit{rlx} \leftrightarrow \mathit{rrx}\}$

$\mathcal{R}(\sigma) \cdot \mathcal{R}(\tau) = \{\mathit{lx} \leftrightarrow \mathit{rx}\}$



Exact resolution can be obtained considering *ancestral types*:

$\sigma' \equiv ((\alpha \multimap \beta) \multimap \gamma) \multimap \alpha \multimap \beta$

$\tau' \equiv (\alpha \multimap \alpha) \multimap \gamma$

$\mathcal{R}(\sigma') = \{\mathit{rlx} \leftrightarrow \mathit{lllx}, \mathit{rrx} \leftrightarrow \mathit{llrx}\}$

$\mathcal{R}(\tau') = \{\mathit{llx} \leftrightarrow \mathit{lrx}\}$

Beyond λ^A: Coppo -Dezani Intersection Types with Modalities

- Finitary logical description of λ-models: M. Coppo, M. Dezani-Ciancaglini, F. Honsell, G. Longo. Extended type structures and filter lambda models. **Logic Colloquium '82**, North Holland, Amsterdam, (1984).
- P. Di Gianantonio, F. Honsell, M. Lenisa. A type assignment system for game semantics. *Theor. Comput. Sci.* **398** (1-3), (2008).
- A. Ciaffaglione, P. Di Gianantonio, F. Honsell, M. Lenisa, I. Scagnetto, λ!-calculus, Intersection Types, and Involutions. *FSCD 2019*, LIPIcs **131**, (2019).
- Antonio Bucciarelli, Delia Kesner, Simona Ronchi Della Rocca - Inhabitation for Non-idempotent Intersection Types, *Logical Methods in Computer Science*, **14**(3), (2018).
- Many papers by Delia Kesner *et al*.

The language of types and indices

The language of types, *Type*$^!$, is a two sorted language given by the following grammars

$$(\mathit{Type}^! \ni)\ \sigma, \tau \ ::=\ \omega \mid \alpha \mid \ldots \mid \sigma \multimap \tau \mid \widehat{\sigma}$$
$$(\widehat{\mathit{Type}^!} \ni)\ \widehat{\sigma}, \widehat{\tau} \ ::= !_u\sigma \mid \widehat{\sigma} \wedge \widehat{\tau}$$
$$(\mathit{Index}^! \ni)\ u, v ::= \epsilon \mid i \mid j \mid \ldots \mid \langle u, v \rangle.$$

- i, j, ... are index variables.
- The syntactic category $\widehat{\sigma}$-types isolates types whose main constructor is ! or $\wedge$.
- The equivalence relation on types $\sim$ is induced by $\omega \sim \sigma$, for any type σ which contains only the constant ω and no type variables.

Principal Types for non-linear Combinators

$$
\begin{aligned}
[\![\mathbf{F}]\!] &= \;!_i(\alpha \multimap \beta) \multimap !_i\alpha \multimap !_i\beta \\
[\![\mathbf{W}]\!] &= (!_i\alpha \multimap !_j\beta \multimap \gamma) \multimap (!_i\alpha \wedge !_j\beta) \multimap \gamma \\
[\![\mathbf{D}]\!] &= \;!_\epsilon\alpha \multimap \alpha \\
[\![\delta]\!] &= \;!_{\langle j,i\rangle}\alpha \multimap !_i!_j\alpha
\end{aligned}
$$

General Principal Type Assignment System $\Vdash$

$$\frac{}{x : \alpha \Vdash x : \alpha} \quad \text{(var)}$$

$$\frac{\Gamma, x : \sigma \Vdash M : \tau}{\Gamma \Vdash \lambda x.M : \sigma \multimap \tau} \quad \text{(abs)} \qquad \frac{\Gamma \Vdash M : \sigma \quad \alpha \text{ fresh}}{\Gamma \Vdash \lambda x.M : \alpha \multimap \sigma} \quad \text{(abs}_\emptyset\text{)}$$

$$\frac{\Gamma \Vdash M : \sigma}{\widehat{!}_i(\Gamma) \Vdash !M : !_i(\sigma)} \quad \text{(!-Intro)} \qquad \frac{\Gamma \Vdash !M : !_\epsilon \sigma \quad !(\Gamma)}{\Gamma \Vdash M : \sigma} \quad \text{(!-Elim)}$$

$$\frac{\Gamma, !x : \widehat{\sigma}, !y : \widehat{\tau} \Vdash M[x, y] : \rho}{\Gamma, !x : \widehat{\sigma} \wedge \widehat{\tau} \Vdash M[x, x] : \rho} \quad (\wedge\text{-Intro-L})$$

$$\frac{\begin{array}{cc} \Gamma \Vdash M : \sigma & \Delta \Vdash N : \tau \\ (dom(\Gamma) \cap dom(\Delta)) = \emptyset & (TVar(\Gamma) \cap TVar(\Delta)) = \emptyset \\ (TVar(\sigma) \cap TVar(\tau)) = \emptyset & U' = \mathcal{M}(\langle \ldots, \sigma, \alpha \multimap \beta) \\ U = \mathcal{M}(\ldots, U'(\alpha), \tau) & \alpha, \beta \textit{ fresh} \end{array}}{U \circ U'(\Gamma, \Delta) \Vdash MN : U \circ U'(\beta)} \quad \text{(app)}$$

Notation

- $!(\Gamma)$ means that all variables in Γ are banged (Γ is possibly empty);
- $\widehat{!}_u(\Gamma, x:\tau) = \widehat{!}_u(\Gamma), !x :!_u\tau$ and
  $\widehat{!}_u(\Gamma, !x:\widehat{\tau}) = \widehat{!}_u(\Gamma), !x : \widehat{!}_u(\widehat{\tau})$, where
  $$\widehat{!}_u(\widehat{\tau}) = \begin{cases} !_{\langle u,v\rangle}\tau & \text{if } \widehat{\tau} = !_v\tau \\ \widehat{!}_u(\widehat{\tau}_1) \wedge \widehat{!}_u(\widehat{\tau}_2) & \text{if } \widehat{\tau} = \widehat{\tau}_1 \wedge \widehat{\tau}_2 \end{cases}$$

Principal Type for $2 \equiv \lambda !x.\lambda !y.x!(x!y)$

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{z : \delta \Vdash z : \delta}{!z :!_{\epsilon}\delta \Vdash !z :!_{\epsilon}\delta}
}{!z :!_{\epsilon}\delta \Vdash z : \delta}
\qquad
\dfrac{
\dfrac{
w : \beta \Vdash w : \beta
\qquad
\dfrac{y : \alpha \Vdash y : \alpha}{!y :!_i\alpha \Vdash !y :!_i\alpha}
}{w :!_i\alpha \multimap \gamma,\ !y :!_i\alpha \Vdash w!y : \gamma}
}{!w :!_j(!_i\alpha \multimap \gamma),\ !y :!_{\langle j,i\rangle}\alpha \Vdash !(w!y) :!_j\gamma}
}{!z :!_{\epsilon}(!_j\gamma \multimap \delta),\ w :!_j(!_i\alpha \multimap \gamma), !y :!_{\langle j,i\rangle}\alpha \Vdash z!(w!y) : \delta}
}{!x :!_k(!_j\gamma \multimap \delta) \wedge !_j(!_i\alpha \multimap \gamma),\ !y :!_{\langle j,i\rangle}\alpha \Vdash x!(x!y) : \delta}
}{\langle\rangle \Vdash \lambda !x.\lambda !y.x!(x!y) :!_k(!_j\gamma \multimap \delta) \wedge !_j(!_i\alpha \multimap \gamma) \multimap (!_{\langle j,i\rangle}\alpha \multimap \delta)}
$$

Principal Type for $2!2 = 4$

$$\dfrac{\dfrac{\cdots}{\langle\rangle \Vdash 2 :!_k(!_j\gamma \multimap \delta)\wedge!_j(!_i\alpha \multimap \gamma) \multimap (!_{\langle j,i\rangle}\alpha \multimap \delta)} \qquad \dfrac{\dfrac{\cdots}{\langle\rangle \Vdash 2 :!_k(!_j\gamma \multimap \delta)\wedge!_j(!_i\alpha \multimap \gamma) \multimap (!_{\langle j,i\rangle}\alpha \multimap \delta)}}{\langle\rangle \Vdash !2 :!_m(!_k(!_j\gamma \multimap \delta)\wedge!_j(!_i\alpha \multimap \gamma) \multimap (!_{\langle j,i\rangle}\alpha \multimap \delta))}}{\langle\rangle \Vdash 2!2 :??}$$

Duplication at work

$$!_m\tau \equiv !_m(!_k(!_j\gamma \multimap \delta)\wedge !_j(!_i\alpha \multimap \gamma) \multimap (!_{\langle j,i\rangle}\alpha \multimap \delta)))$$
$$!'_k(!_{j'}\gamma' \multimap \delta')\wedge !_{j'}(!_{i'}\alpha' \multimap \gamma') \qquad \multimap \qquad (!_{\langle i',j'\rangle}\alpha' \multimap \delta')$$

duplicate $!_m$ *i.e.* $!_m\tau \mapsto !_{m_l}\tau_l \wedge !_{m_l}\tau_r$

$$\tau_l \equiv \quad !_{k_l}(!_{j_l}\gamma_l \multimap \delta_l)\wedge !_{j_l}(!_{i_l}\alpha_l \multimap \gamma_l) \quad \multimap \quad (!_{\langle i_l,j_l\rangle}\alpha_l \multimap \delta_l)$$
$$!_{j'}\gamma' \quad \multimap \quad \delta'$$

duplicate $!_{j'}$ *i.e.* $!_{j'}\gamma' \mapsto !_{j'_l}\gamma'_l \wedge !_{j'_r}\gamma'_r$

$$!_{k_l}(!_{j_l}\gamma_l \multimap \delta_l)\wedge !_{j_l}(!_{i_l}\alpha_l \multimap \gamma_l)$$
$$!_{j^l}\gamma^1_l \wedge !_{j^r}\gamma^1_r$$

but since on the right hand side we had

$$!_{m_r}\tau_r \equiv !_{m_r}(\quad !_{k_r}(!_{j_r}\gamma_r \multimap \delta_r)\wedge !_{j_r}(!_{i_r}\alpha_r \multimap \gamma_r) \quad \multimap \quad (!_{\langle i_r,j_r\rangle}\alpha_r \multimap \delta_r) \quad)$$
$$!_{j'}(\quad !_{i'}\alpha' \quad \multimap \quad \gamma' \quad)$$

and $!_{j'}$ has been duplicated we have, in fact,

$$!_{m_r}(\quad !_{k_r}(!_{j_r}\gamma_r \multimap \delta_r)\wedge !_{j_r}(!_{i_r}\alpha_r \multimap \gamma_r) \quad \multimap \quad (!_{\langle i_r,j_r\rangle}\alpha_r \multimap \delta_r) \quad)$$
$$!_{j'_l}(!_{i'_l}\alpha'_l \multimap \gamma'_l) \quad \wedge \quad !_{j'_r}(!_{i'_r}\alpha'_r \multimap \gamma'_r)$$

hence we have to duplicate also $!_{m_r}\tau_r$, namely we have to meta-unify

$$(!_{m_{rl}}(!_{k_{rl}}(!_{j_{rl}}\gamma_{rl} \multimap \delta_{rl})\wedge !_{j_{rl}}(!_{i_{rl}}\alpha_{rl} \multimap \gamma_{rl}) \multimap (!_{\langle i_{rl},j_{rl}\rangle}\alpha_{rl} \multimap \delta_{rl}))) \qquad \wedge$$
$$!_{j'_l}(!_{i'_l}\alpha'_l \multimap \gamma'_l) \qquad \wedge$$

$$(!_{m_{rr}}(!_{k_{rr}}(!_{j_{rr}}\gamma_{rr} \multimap \delta_{rr})\wedge !_{j_{rr}}(!_{i_{rr}}\alpha_{rr} \multimap \gamma_{rr}) \multimap (!_{\langle i_{rr},j_{rr}\rangle}\alpha_{rr} \multimap \delta_{rr})))$$
$$!_{j'_r}(!_{i'_r}\alpha'_r \multimap \gamma'_r)$$

...

Duplication at work

Duplications triggered by $!_{j'}$ are reflected also in

$$!_{\langle j', i' \rangle}\alpha' \multimap \delta'$$

which is meta-substituted into

$$(!_{\langle j'_l, i'_l \rangle}\alpha'_l \wedge !_{\langle j'_r, i'_r \rangle}\alpha'_r) \multimap \delta'$$

Metaunification (in progress)

Two issues

- duplication of !-indices trigger, even at-a-distance, duplication of subtypes which they encapsulate;
- the connection with the encapsulating !-indices has to be maintained:

we need a new meta-unification judgment, $\mathcal{M}$, which rewrites sets of *t*-ples consisting of appropriate parameters to take care of the above issues and two types, $\langle \ldots, \sigma, \tau \rangle$.