On Type Inference in the Intersection Type Discipline

ITRS'04 – July 13th, 2004

Gérard Boudol and Pascal Zimmer

INRIA Sophia Antipolis

History

- system $\mathcal{D}$: Coppo, Dezani, 1980; Pottinger, 1980
- principal typing: Coppo, Dezani and Venneri, 1980; Ronchi della Rocca and Venneri, 1984
- inference: Ronchi della Rocca, 1988
- system $\mathbb{I}$: Kfoury and Wells, 1999 (expansion variables)
- system E: Carlier, Kfoury, Polakow and Wells, 2004

Types syntax

$$\begin{array}{rcl} \tau, \sigma \ldots & ::= & t \mid \pi \rightarrow \sigma \\ \pi, \kappa \ldots & ::= & \omega \mid \tau \mid \pi \wedge \kappa \end{array}$$

- conjunction only at the left of an arrow
- empty sequence denoted by ω
- types considered modulo the congruence $\equiv_{UACI}$:

$$\begin{array}{rcll} \omega \wedge \pi & \equiv & \pi & (U) \\ (\pi_0 \wedge \pi_1) \wedge \pi_2 & \equiv & \pi_0 \wedge (\pi_1 \wedge \pi_2) & (A) \\ \pi_0 \wedge \pi_1 & \equiv & \pi_1 \wedge \pi_0 & (C) \\ \pi \wedge \pi & \equiv & \pi & (I) \end{array}$$

and written $\tau_1, \ldots, \tau_n \rightarrow \sigma$.

INRIA

Typing rules

$$\frac{}{x : \tau \vdash x : \tau}\text{(Typ Id)}$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma \setminus x \vdash \lambda x M : \Gamma(x) \rightarrow \tau}\text{(Typ } \lambda\text{)}$$

$$\frac{\Gamma \vdash M : \tau_1, \ldots, \tau_n \rightarrow \sigma \quad \forall i,\ \Delta_i \vdash N : \tau_i}{\Gamma \wedge \Delta_1 \wedge \ldots \wedge \Delta_n \vdash MN : \sigma}\text{(Typ Appl Gen)} \qquad (n \geq 1)$$

$$\frac{\Gamma \vdash M : \omega \rightarrow \sigma \quad \Delta \vdash N : \tau}{\Gamma \wedge \Delta \vdash MN : \sigma}\text{(Typ Appl } \omega\text{)}$$

$$\frac{\Gamma \vdash M : \tau \quad \Gamma \equiv_{UACI} \Delta}{\Delta \vdash M : \tau}\text{(Typ Congr)}$$

INRIA

Properties

- Subject reduction: If $M \to M'$, then

$$\Gamma \vdash M : \tau \implies \Gamma \vdash M' : \tau$$

- Theorem: A term M is typable in $\mathcal{D}$ if and only if M is strongly normalising.

Properties

- Subject reduction: If $M \to M'$, then

$$\Gamma \vdash M : \tau \implies \Gamma \vdash M' : \tau$$

- Theorem: A term M is typable in $\mathcal{D}$ if and only if M is strongly normalising.
- Trivial algorithm: try to strongly normalise, then type.
- Problem: does not work for an extended calculus (recursion...)
- We have the type, but not the typing tree...

Algorithm: general idea

Mimick β-reduction on types:

$$(\lambda x M)N \rightarrow_\beta M\{x \mapsto N\} = M[\ldots N \ldots N \ldots]$$

$$\tau_N \rightarrow t \perp t_1, \ldots, t_n \rightarrow \tau_M$$

Copy n times the type variables and constraints of N.

$\Rightarrow$ territory ($=$ set of type variables)

Identify t with τ_M, and t_i with the i^{th} copy of τ_N.

Algorithm: general idea

Mimick β-reduction on types:

$$(\lambda x M)N \to_\beta M\{x \mapsto N\} = M[\ldots N \ldots N \ldots]$$

$$\tau_N \to t \perp t_1, \ldots, t_n \to \tau_M$$

Copy n times the type variables and constraints of N.

$\Rightarrow$ territory ($=$ set of type variables)

Identify t with τ_M, and t_i with the i^{th} copy of τ_N.

Take care of β_K-redexes: $(\lambda x M)N \to_\beta M$

$\Rightarrow$ special rule for $n = 0$

$\Rightarrow$ extended λ-calculus

INRIA

Implementation – Notations

- Implementation of the algorithm: TYPI

  `http://www-sop.inria.fr/mimosa/Pascal.Zimmer/typi.html`

- Notations slightly different from the paper in the proceedings.

INRIA

Example

$$M = F(\lambda u\ \Delta(uu))$$

with $F = \lambda x\ I = \lambda x \lambda y\ y$ and $\Delta = \lambda x\ (xx)$

Example

$$M = F(\lambda u\, \Delta(uu))$$

with $F = \lambda x\, I = \lambda x \lambda y\, y$ and $\Delta = \lambda x\, (xx)$

- First step:
  annotate every variable and application with a fresh type variable.

$$M' = (F'\, (\lambda u\, (\Delta'(u^{t_4}\, u^{t_5})^{t_6})^{t_7}))^{t_8}$$

where $F' = \lambda x \lambda y\, y^{t_0}$
and $\Delta' = \lambda x\, (x^{t_1}\, x^{t_2})^{t_3}$

Example - $F(\lambda u\ \Delta(uu))$

- Second step:
  for every application $(M'N')^t$, build the constraint:

$$Typ(N') \to t \ \overset{\perp}{=} \ Typ(M') \ \ [ftv(N')]$$

Example - $F(\lambda u\ \Delta(uu))$

- Second step:
  for every application $(M'N')^t$, build the constraint:

$$Typ(N') \rightarrow t \ \underline{\perp}\ Typ(M') \ \ [ftv(N')]$$

$$\left\{ \begin{array}{rcll} (t_4, t_5 \rightarrow t_7) \rightarrow t_8 & \underline{\perp} & \omega \rightarrow t_0 \rightarrow t_0 & [t_1, \ldots, t_7], \\ t_6 \rightarrow t_7 & \underline{\perp} & t_1, t_2 \rightarrow t_3 & [t_4, t_5, t_6], \\ t_5 \rightarrow t_6 & \underline{\perp} & t_4 & [t_5], \\ t_2 \rightarrow t_3 & \underline{\perp} & t_1 & [t_2] \end{array} \right\}$$

Example - $F(\lambda u\ \Delta(uu))$

Decomposition of:

$$t_6 \to t_7 \perp t_1, t_2 \to t_3 \ [t_4, t_5, t_6]$$

corresponding to $\Delta(uu) \to (uu)(uu)$.

- $D(2, \{t_4, t_5, t_6\})$: duplicate the equations whose node is in (uu), duplicate the type variables occurring in (uu)
- substitute $\{t_7 \mapsto t_3, \emptyset\}$
- replace the x in Δ by the two copies:

$$\{t_1 \mapsto t_6^1, \{t_4^1, t_5^1, t_6^1\}\} \ ; \ \{t_2 \mapsto t_6^2, \{t_4^2, t_5^2, t_6^2\}\}$$

Example - $F(\lambda u\ \Delta(uu))$

Updated system:

$$\left\{\begin{array}{rcll}
(t^1_4, t^2_4, t^1_5, t^2_5 \to t_3) \to t_8 & \perp\!\!\!\!\!= & \omega \to t_0 \to t_0 & [T], \\
t^2_6 \to t_3 & \perp\!\!\!\!\!= & t^1_6 & [t^2_4, t^2_5, t^2_6] \\
t^1_5 \to t^1_6 & \perp\!\!\!\!\!= & t^1_4 & [t^1_5], \\
t^2_5 \to t^2_6 & \perp\!\!\!\!\!= & t^2_4 & [t^2_5]
\end{array}\right.$$

where $T = \{t_3, t^1_4, t^2_4, t^1_5, t^2_5, t^1_6, t^2_6\}$

Those equations correspond to the term:

$$F(\lambda u\ (uu)(uu))$$

Example - $F(\lambda u\ \Delta(uu))$

Decomposition of:

$$(t_4^1, t_4^2, t_5^1, t_5^2 \to t_3) \to t_8 \perp\!\!\!= \omega \to t_0 \to t_0 \;\; [T]$$

We should not “erase” the argument, since it must be typable ! Updated system:

$$\left\{ \begin{array}{llll} t_6^2 \to t_3 & \perp\!\!\!= & t_6^1 & [t_4^2, t_5^2, t_6^2], \\ t_5^1 \to t_6^1 & \perp\!\!\!= & t_4^1 & [t_5^1], \\ t_5^2 \to t_6^2 & \perp\!\!\!= & t_4^2 & [t_5^2] \end{array} \right\}$$

Those equations correspond to the terms:

$$I \;\; \text{and} \;\; \lambda u\ (uu)(uu)$$

(no equation for I) and not to I alone.

INRIA

$\Lambda_\mathcal{K}$-calculus

- Inspired by Klop, 1980.
- Syntax:

$$M, N ::= x \mid M\,N \mid \lambda x M \mid [M, N]$$

- Semantics:

For $x \in fv(M)$:

$$[\lambda x M, N_1, \ldots, N_n]\, N \longrightarrow_\mathcal{K} [M\{x \mapsto N\}, N_1, \ldots, N_n]$$

For $x \notin fv(M)$:

$$[\lambda x M, N_1, \ldots, N_n]\, N \longrightarrow_\mathcal{K} [M, N_1, \ldots, N_n, N]$$

$\Lambda_{\mathcal{K}}$-calculus

- $\mathcal{WN}_{\mathcal{K}} = \mathcal{SN}_{\mathcal{K}}$: normalising terms are strongly normalising
- $\mathcal{SN}_{\Lambda} = \Lambda \cap \mathcal{SN}_{\mathcal{K}}$: they correspond to strongly normalising terms in λ-calculus
- We add the typing rule:

$$\frac{\Gamma_1 \vdash M_1 : \tau \qquad \Gamma_2 \vdash M_2 : \sigma}{\Gamma_1 \wedge \Gamma_2 \vdash [M_1, M_2] : \tau}\text{(Typ Forget)}$$

Reduction rules

System state: $(\mathcal{E}, \Pi)$ where

- $\mathcal{E}$ is a set of constraints
- Π is a proof skeleton, that will evolve to a valid typing tree

Reduction rules

System state: $(\mathcal{E}, \Pi)$ where

- $\mathcal{E}$ is a set of constraints
- Π is a proof skeleton, that will evolve to a valid typing tree

Rule for $n \geq 1$:

$$(\{\tau \to t \perp t_1, \ldots, t_n \to \sigma \;\; [T]\} \cup \mathcal{E},\ \Pi) \longrightarrow (S(\mathcal{E}),\ S(\Pi))$$

with $S = \{t_i \mapsto \langle\tau\rangle^i, \langle T\rangle^i\}_{1 \leq i \leq n}\ ;\ \{t \mapsto \sigma, \emptyset\}\ ;\ D(n, T)$

(R_n)

Reduction rules

Rule for $n = 0$:

$$(\{\tau \to t \perp \omega \to \sigma \ [T]\} \cup \mathcal{E},\ \Pi) \ \longrightarrow\ (S(\mathcal{E}),\ S(\Pi))$$

with $S = \{t \mapsto \sigma, \emptyset\}$

(R_0)

Final rule:

$$(\{\tau \perp t\} \cup \mathcal{E},\ \Pi) \ \longrightarrow_f\ (S(\mathcal{E}),\ S(\Pi)) \quad \text{with } S = \{t \mapsto \tau\}$$

(R_f)

Results

- A term M is in normal form if and only if the corresponding system $\mathcal{E}_M$ is irreducible.

Results

- A term M is in normal form if and only if the corresponding system $\mathcal{E}_M$ is irreducible.
- Operational correspondence...

Results

- A term M is in normal form if and only if the corresponding system $\mathcal{E}_M$ is irreducible.
- Operational correspondence...
- Theorem: A term M is typable if and only if the initial system corresponding to M converges.

INRIA

Results

- A term M is in normal form if and only if the corresponding system $\mathcal{E}_M$ is irreducible.
- Operational correspondence...
- Theorem: A term M is typable if and only if the initial system corresponding to M converges.
- Theorem: If M is typable, then the final proof skeleton is a valid typing tree for M. Moreover, the final *typing* is *principal*, in the sense of Coppo, Dezani, Venneri 80.

Other results and ongoing work

- Strong conjecture: The *typing tree* is *principal*.
- Rank: Syntactic definition on types; to evaluate the “level” of polymorphism.

  Property: The finite-rank algorithm *always stops*.
  Consequence: Finite-rank inference is *decidable*.

INRIA

Other results and ongoing work

- Variant: by replacing the rule (R_0) with the general rule (R_n); related to the type system $\mathcal{D}\Omega$, with the rule:

$$\frac{}{\vdash M : \omega}\text{(Typ }\omega\text{)}$$

  (if the algorithm converges, then the term is typable).
- Extension to references (introducing conjunction only for values, as in ML; less liberty on the order of resolution)
- Extension to recursion $\mu x M$ (additional unification step at the end of the algorithm)

Comparison

- The $\Lambda_{\mathcal{K}}$-calculus is made explicit; easier proofs.

Ronchi della Rocca 88

- complex definition to compute the expansion

System $\mathbb{I}$

- expansion variables vs territories, different type systems
- different atomicity of operations (1 step $\Rightarrow n + 2$ steps)

System E

- similar to the variant with ω; system $\mathcal{D}\Omega$ with expansion variables

The end

Rank

$$inc(0) = 0$$
$$inc(n) = n + 1 \quad \text{for } n > 0$$

$$rank(t) = 0$$
$$rank(\tau \rightarrow \sigma) = \max(inc(rank(\tau)), rank(\sigma))$$
$$rank(\tau_1, \ldots, \tau_n \rightarrow \sigma) =$$
$$\max(inc(\max(1, rank(\tau_1), \ldots, rank(\tau_n))), rank(\sigma))$$
$$\text{for } n \neq 1$$

Rank

Syntactic definition on types...

- rank 0: usual types without intersection
- rank 1: empty
- rank $r \geq 2$: there is a non-trivial conjunction under $r - 1$ arrows
  Example:
  $(t_1 \to t_2), (\omega \to t_3) \to t_1 \to t_3$ has rank 3

Finite-rank algorithm

- Choose a maximal allowed rank r.
- For every intermediate step $(\mathcal{E}, \Pi)$, check that $rank(\Pi) \leq r$.
- Otherwise, the term is not typable at rank r.

Finite-rank algorithm

- Choose a maximal allowed rank r.
- For every intermediate step $(\mathcal{E}, \Pi)$, check that $rank(\Pi) \leq r$.
- Otherwise, the term is not typable at rank r.

Property: The finite-rank algorithm *always stops*.

Consequence: Finite-rank inference is *decidable*.

INRIA

Variant

What happens if we use the general rule also for $n = 0$?

$$(\{\tau \to t \stackrel{?}{=} t_1, \ldots, t_n \to \sigma \ \ [T]\} \cup \mathcal{E},\ \Pi) \ \longrightarrow\ (S(\mathcal{E}),\ S(\Pi))$$
$$\text{with } S = \{t_i \mapsto \langle\tau\rangle^i, \langle T\rangle^i\}_{1\leq i\leq n}\ ;\ \{t \mapsto \sigma, \emptyset\}\ ;\ D(n, T)$$

$$(R_n)$$

- Leads to "erase" constraints or sub-trees by $D(0, T)$
- Correspondence with the type system $\mathcal{D}\Omega$ (Krivine) or $\lambda\cap$ (Barendregt)

$$\frac{}{\vdash M : \omega}\,(\text{Typ } \omega)$$

Variant

- Property: The variant of the algorithm converges iff the term is normalising.
- Proposition: A term is typable in $\mathcal{D}\Omega$ with a non-trivial type iff it has a head-normal form.
- Caracterisation of normalising terms.
- Corollary: If the algorithm converges, then the term is typable.
- Reciprocal property: not true (example: $x\Omega$)

System $\mathbb{I}$

- System proposed by Kfoury and Wells (variant: System E with Carlier)
- Types contain *expansion variables*:

$$\begin{array}{rcl} \psi & ::= & \alpha \mid (\psi \rightarrow \psi) \\ \psi & ::= & \psi \mid (\psi \wedge \psi') \mid (F\psi) \end{array}$$

- Algorithm for solving similar constraints and returning a typing tree

INRIA

System $\mathbb{I}$

- Correspondence expansion variables / territory:

$$F_T \quad \longleftrightarrow \quad T = \{v \mid F_T \in \text{E-path}(v, \Gamma_{\mathbb{I}}(M))\}$$

- Both algorithms perform the same operations, not necessarily in the same order, if we ignore expansion variables
$\longrightarrow$ operational correspondence
- Used to avoid redoing the proofs of some results (principality, finite rank)

INRIA

References

The expression

$$(\lambda r\,(r := [\texttt{"a string"}]\,;\ \texttt{hd}(!\,r) + 1))\,(\texttt{ref}\,[\,])$$

is typable, but its execution leads to an error...

References

The expression

$$(\lambda r\,(r := [\texttt{"a string"}]\,;\ \texttt{hd}(!\,r) + 1))\,(\texttt{ref}\,[\,])$$

is typable, but its execution leads to an error...

Solution similar to the one for polymorphism in ML: introducing conjunction only for *values* (Davies and Pfenning).

$$\frac{\Gamma \vdash V : A \qquad \Gamma \vdash V : B}{\Gamma \vdash V : A \wedge B}$$

$$\frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

References

- Distinguish the types of terms-variables and applications: t_v and $t_@$
- Extended syntax for types:

$$t_b ::= t_v \mid t_b\, ref \mid cte \mid t_b\, list$$

$$\tau, \sigma ::= t_v \mid \tau\, ref \mid cte \mid \tau\, list \mid t_@ \mid t_b, \ldots, t_b \rightarrow \tau$$

- Decomposible equations:

$$\tau \rightarrow t_@ \ \perp\!\!\!\!\!= \ t_{b_1}, \ldots, t_{b_n} \rightarrow \sigma \ \ [T]$$

References

$$(\{\tau \to t_{@} \stackrel{\perp}{=} t_{b_1}, \ldots, t_{b_n} \to \sigma \ [T]\} \cup \mathcal{E},\ \Pi) \longrightarrow (S(\mathcal{E}),\ S(\Pi))$$

$$\text{with } S = \begin{cases} mgu(t_{b_i}, \langle\tau\rangle^i, \langle T\rangle^i)_{1\leq i\leq n} \ ;\ \{t_{@} \mapsto \sigma, \emptyset\} \ ;\ D(n,T) & \text{if } ValueType(\tau) \\ mgu(t_{b_i}, \tau, T)_{1\leq i\leq n} \ ;\ \{t_{@} \mapsto \sigma, \emptyset\} & \text{otherwise} \end{cases}$$

INRIA

References

$$(\{\tau \to t_{@} \overset{\perp}{=} t_{b_1}, \ldots, t_{b_n} \to \sigma\ [T]\} \cup \mathcal{E},\ \Pi) \longrightarrow (S(\mathcal{E}),\ S(\Pi))$$

$$\text{with } S = \begin{cases} mgu(t_{b_i}, \langle\tau\rangle^i, \langle T\rangle^i)_{1\leq i\leq n}\ ;\ \{t_{@} \mapsto \sigma, \emptyset\}\ ;\ D(n,T) & \text{if } ValueType(\tau) \\ mgu(t_{b_i}, \tau, T)_{1\leq i\leq n}\ ;\ \{t_{@} \mapsto \sigma, \emptyset\} & \text{otherwise} \end{cases}$$

but we also need to impose an order for solving the constraints, corresponding more or less to call-by-value...

Recursion

- We add an operator $\mu x M$
- Solution: infer types as for M, then additional unification algorithm
- Modify the type system:

$$\frac{\Gamma, x : \sigma_1, \ldots, x : \sigma_n \vdash M : \tau}{\Gamma \vdash \mu x\, M : \tau}\text{(REC)} \quad \text{with } \forall i\ \sigma_i \equiv \tau$$

- Equality modulo commutativity and contraction:

$$\ldots, \tau_1, \tau_2, \ldots \longrightarrow \sigma \quad \equiv \quad \ldots, \tau_2, \tau_1, \ldots \longrightarrow \sigma$$

$$\ldots, \tau, \tau, \ldots \longrightarrow \sigma \quad \equiv \quad \ldots, \tau, \ldots \longrightarrow \sigma$$

INRIA