Type Inference in Intuitionistic Linear Logic

Patrick Baillot Martin Hofmann

LIP- CNRS & ENS Lyon LMU Munich

(to appear in PPDP 2010)

10/6/2010
Torino, Concerto-Pics meeting

Introduction

- linear logic (ILL) has inspired linear type systems (e.g. Wadler'91, Mackie'94 . . .)
- automatic type inference ?
- linear decoration of intuitionistic derivations : Danos-Joinet-Shellinx94
  but does not provide all ILL derivations we are looking for
- related issue: type inference in light logics

system	EAL	DLAL	ILL
constraints	linear inequations	mixed boolean/linear constraints	?
inference	P [BT05]	P [ABT07]	?

System F

System F terms will be our source language.

Types:

$$A ::= \alpha \mid A_1 \to A_2 \mid \forall \alpha. A$$

Church-style lambda terms:

$$t ::= x \mid t_1 t_2 \mid \lambda x{:}A.t \mid t[A] \mid \Lambda \alpha.t$$

$FV(t)$: free term variables,
$FTV(T)$, $FTV(t)$: free type variables.

Typing rules for system F

$$\frac{}{\Gamma, x{:}A \vdash x : A}\text{(F-Var)}$$

$$\frac{\Gamma \vdash t_1 : A \to B \qquad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B}\text{(F-App)}$$

$$\frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x{:}A.t : A \to B}\text{(F-Lam)}$$

$$\frac{\Gamma \vdash t : A \qquad \alpha \notin FTV(\Gamma)}{\Gamma \vdash \Lambda\alpha.t : \forall\alpha.A}\text{(F-TLam)}$$

$$\frac{\Gamma \vdash t : \forall\alpha.A}{\Gamma \vdash t[B] : A[B/\alpha]}\text{(F-TApp)}$$

Linear System F: LLF

LLF types:

$$T ::= \alpha \mid T_1 \multimap T_2 \mid \forall \alpha . T \mid\ !T$$

$|.|$ maps LLF types to system F types: forgetting ! and replacing $\multimap$ by $\rightarrow$.

Typing rules for LLF

$$\frac{}{x{:}T \vdash x : T}\text{(LLF-Var)}$$

$$\frac{\Gamma, x{:}S \vdash t : T \qquad \Delta \vdash t' : S \qquad \text{x has at most one free occurrence in t}}{\Gamma, \Delta \vdash t[t'/x] : T}\text{(LLF-Cut)}$$

$$\frac{\Gamma, x{:}S \vdash t : T}{\Gamma \vdash \lambda x{:}|S|.t : S \multimap T}\text{(LLF-Lam)}$$

$$\frac{\Gamma \vdash t_1 : S \multimap T \qquad \Delta \vdash t_2 : S}{\Gamma, \Delta \vdash t_1\, t_2 : T}\text{(LLF-App)}$$

$$\frac{\Gamma, x_1{:}!S, x_2{:}!S \vdash t : T}{\Gamma, x{:}!S \vdash t[x/x_1, x/x_2] : T}\text{(LLF-Contr)}$$

$$\frac{\Gamma \vdash t : T}{\Gamma, x{:}S \vdash t : T}\text{(LLF-Weak)}$$

$$\frac{\Gamma \vdash t : T}{!\Gamma \vdash t : !T}\text{(LLF-Prom)}$$

$$\frac{\Gamma \vdash t : !T}{\Gamma \vdash t : T}\text{(LLF-Der)}$$

$$\frac{\Gamma \vdash t : !T}{\Gamma \vdash t : !!T}\text{(LLF-Dig)}$$

Rules for $\forall$ TLam and TApp are unchanged.

Type checking and type inference for LLF

Type checking for LLF: *Given* t, T, Γ *is* $\Gamma \vdash t : T$ *a valid LLF-judgement?*

Type inference for LLF:
Given a system F type A*, context* Δ *and term* t*, find a concise description of the set of LLF types* T *and contexts* Γ *with* $\Gamma \vdash t : T$ *and* $|T| = A$, $|\Gamma| = \Delta$.

Towards an algorithmic typing system

Goal: remove non syntax-directed rules.

Key issue: LLF-Cut rule.
$\rightarrow$ *decompose* LLF-Prom rule (box) into more basic rules.

For that we need to carry an extra piece of information: natural integers (allow to retrieve the depth).

close to [GuerriniMartiniMasini98] (2-sequents), [MartiniMasini95]
also related to [PfenningWong95]

Algorithmic typing contexts

- an *algorithmic typing context* Γ is a partial map over term and type variables:
  $\Gamma(x) = (T, m), \quad \Gamma(\alpha) = (\star, m')$ where T LLF type and $m, m' \in \mathbb{N}$.
  Denote variables x and α as $X, Y, \ldots$
- Γ is *well-formed* if, for all x:
  $(\Gamma(x) = (T, m) \wedge \alpha \in FTV(T)) \Rightarrow \Gamma(\alpha) = (\star, m')$ with $m' \geq m$.

  If $c \in \mathbb{Z}$ then

  $$\Gamma^{+c} \text{ ok means: if } \Gamma(X) = (U, m), \text{ then } m + c \geq 0.$$

  Then Γ^{+c} is defined by:

  $$\text{if } \Gamma(X) = (U, m), \text{ then } (\Gamma^{+c})(X) = (U, m + c).$$
- ALLF typing judgements:
  $\Gamma \vdash_a t : T$ where Γ well-formed.

ALLF typing rules

$$\frac{\Gamma(x) = (T, 0)}{\Gamma \vdash_{\boldsymbol{a}} x : T}(\mathsf{Var}) \qquad \frac{\Gamma, x{:}(S, 0) \vdash_{\boldsymbol{a}} t : T}{\Gamma \vdash_{\boldsymbol{a}} \lambda x{:}|S|.t : S \multimap T}(\mathsf{Lam})$$

$$\frac{\Gamma \vdash_{\boldsymbol{a}} t_1 : S \multimap T \qquad \Gamma \vdash_{\boldsymbol{a}} t_2 : S \qquad x \in FV(t_1) \cap FV(t_2) \Rightarrow \exists T.\Gamma[x] = !T}{\Gamma \vdash_{\boldsymbol{a}} t_1 t_2 : T}(\mathsf{App})$$

$$\frac{\Gamma, \alpha{:}(\star, 0) \vdash_{\boldsymbol{a}} t : T \qquad \alpha \notin FTV(\Gamma)}{\Gamma \vdash_{\boldsymbol{a}} \Lambda\alpha.t : \forall\alpha.T}(\mathsf{TLam}) \qquad \frac{\Gamma \vdash_{\boldsymbol{a}} t : \forall\alpha.T \qquad FTV(S) \subseteq dom(\Gamma)}{\Gamma \vdash_{\boldsymbol{a}} t[S] : T[|S|/\alpha]}(\mathsf{TApp})$$

$$\frac{\Gamma \vdash_{\boldsymbol{a}} t : !T}{\Gamma^{+1}, \Delta^0 \vdash_{\boldsymbol{a}} t : T}(\mathsf{Enter}) \qquad \frac{\Gamma \vdash_{\boldsymbol{a}} t : T \qquad \Gamma^{-1}\ \mathsf{ok}}{\Gamma^{-1} \vdash_{\boldsymbol{a}} t : !T}(\mathsf{Leave})$$

Rules Der and Dig : as in LLF.

If $\Delta = x_1 : T_1, \ldots, x_n : T_n$, then $\Delta^0 = x_1 : (T_1, 0), \ldots, x_n : (T_n, 0)$.

From LLF to ALLF, and back

Theorem (Completeness of ALLF)

If $\Gamma \vdash t : T$ and $\mathcal{E} = FTV(T) \cup FTV(t)$, then we have $(\Gamma + \mathcal{E})^0 \vdash_a t : T$.

where $(\Gamma + \mathcal{E})^0 = \Gamma^0 \cup \bigcup_{\alpha \in \mathcal{E}} \{\alpha : (\star, 0)\}$.

Note that as a particular case, if $FTV(T) \cup FTV(t) \subseteq FTV(\Gamma)$ then $\Gamma^0 \vdash_a t : T$.

Theorem (Soundness of ALLF)

Let T be an LLF type and Γ an LLF context. If $\Gamma^0 \vdash_a t : T$ then $\Gamma \vdash t : T$.

Towards type checking ? ALLF is not enough . . .

The system ALLF is not yet ready for type checking/inference.

Indeed: we have removed the Cut rule, but . . .
the rules handling ! are still not syntax-directed.

idea : we will group sequences of such rules into clusters.
For that we define one generic !-rule.

A generic ! rule

$$\frac{\Gamma \vdash_a t : !^p T \qquad FV(t) = FV(\Gamma) \qquad \text{condition } (*)}{\Gamma^{+c}, \Delta \vdash_a t : !^q T} \text{(ALL-!)}$$

with

$$(*) \begin{cases} T \text{ is not of the form } !T', \\ p \geq 0, q \geq 0 \\ \Gamma^{+c} \text{ ok} \\ (\Gamma^{-1} \text{ ok}) \vee (p \neq 0) \vee (q = c = 0) \end{cases}$$

Proposition

- *The rule* ALL-! *is derivable in ALLF.*
- *Any sequence of rules* Enter, Leave, Der, Dig *in ALLF can be represented by **one** instance of rule* ALL-!.

Parameterizing types

We will use the idea of *typing by decoration*, using parameters (used e.g. in [CoppolaMartini01], [B02], ...)

For that, we define LLF-*type schemas*:

$$T ::= A \mid T_1 \multimap T_2 \mid \forall\alpha.T \mid !^q T$$

where qs are formal parameters.

Free parameterization:

$$\begin{array}{ccc} \text{System F types} & & \text{LLF type schemas} \\ A & \longrightarrow & A^T \end{array}$$

Example:

$$\begin{array}{rcl} A & = & \forall\alpha.\alpha \to \alpha \\ A^T & = & !^a(\forall\alpha.!^b(!^c\alpha \multimap !^d\alpha)) \end{array}$$

Type checking algorithm: (i) constraints generation

input: in LLF, (Γ, t, T)
we will construct a *parameterized derivation* (parameters in types and for levels).

- schemas $|T|^T$ and $|\Gamma(x)|^T$
- initial constraints
- start from $(|\Gamma|^T + \mathcal{E})^0 \vdash_a t : |T|^T$ and apply bottom-up, alternatively:
  - ALL-! rule, and introduce fresh parameters,
  - syntax-directed logical rule.

  At each step collect side-conditions and unification conditions, $\rightarrow$ system $\mathcal{C}$ of arithmetic constraints.

Constraints

constraints $\mathcal{C}$ generated:

$a \geq 0$	(nonnegativity)
$a \neq 0$	(shared variables in App)
$a + b = c$	(coupling of premise and conclusion)
$a \neq 0 \vee b \neq 0 \vee c = 0$	((ALL-!) rule)

$\rightarrow$ We need a generalization of (conjunctive) linear constraints.

Horn disjunctive linear relations (Horn DLR):

$$\wedge_{i=1}^{N}(l_{i,1} \vee l_{i,2} \vee \cdots \vee l_{i,n_i})$$

where for each i, among the $l_{i,j}$ linear conditions there are:

disequations ($\neq$) and **at most one** inequality ($\leq$).

Theorem (Jonsson and Bäckström '96)

Satisfiability of Horn DLR over $\mathbb{Q}$ is decidable in polynomial time.

Type checking algorithm: (ii) constraints resolution

$\mathcal{C}$ constraint system generated.

Proposition

If ϕ is a solution of $\mathcal{C}$ and $k \in \mathbb{N}^*$, then $\phi' = k.\phi$ is a solution too.

As $\mathcal{C}$ belongs to the class of Horn DLR:

- by the previous Theorem it can be decided in polynomial time over $\mathbb{Q}$,
- hence by this Proposition it can be decided in polynomial time over $\mathbb{Z}$.

Example

does $\quad x{:}!(!B \multimap C), y{:}!(A \multimap B), z{:}!A \vdash x(yz) : !C$ hold ?

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}(!^{a_1}(!^{b_1}B \multimap C), c_3) \vdash_a x : !^{j_1}(!^{i_1}B \multimap !^{h_2}C)}\ \mathrm{Var}_1}{x{:}(!^{a_1}(!^{b_1}B \multimap C), c_2) \vdash_a x : !^{i_1}B \multimap !^{h_2}C}\ \text{ALL-}!_2 \qquad \dfrac{\dfrac{\vdots}{y{:}(!^{d_1}(A \multimap B), e_3), z{:}(!^{f_1}A, g_3) \vdash_a yz : !^{i_2}B}\ \mathrm{App}}{y{:}(!^{d_1}(A \multimap B), e_2), z{:}(!^{f_1}A, g_2) \vdash_a yz : !^{i_1}B}\ \text{ALL-}!_3}{x{:}(!^{a_1}(!^{b_1}B \multimap C), c_2), y{:}(!^{d_1}(A \multimap B), e_2), z{:}(!^{f_1}A, g_2) \vdash_a x(yz) : !^{h_2}C}\ \mathrm{App}}{x{:}(!^{a_1}(!^{b_1}B \multimap C), c_1), y{:}(!^{d_1}(A \multimap B), e_1), z{:}(!^{f_1}A, g_1) \vdash_a x(yz) : !^{h_1}C}\ \text{ALL-}!_1$$

Constraints:

Var_1	$j_1 = a_1, c_3 = 0, i_1 = b_1, h_2 = 0$
$!_3$	$e_2 = e_3 + C_3, g_2 = g_3 + C_3, (e_3 \neq 0 \wedge g_3 \neq 0) \vee i_2 \neq 0 \vee i_1 = C_3 = 0$
$!_2$	$\begin{cases} c_2 = c_3 + C_2, \\ c_3 \neq 0 \vee j_1 \neq 0 \vee C_2 = 0 \end{cases}$
$!_1$	$\begin{cases} c_1 = c_2 + C_1, e_1 = e_2 + C_1, g_1 = g_2 + C_1, \\ (c_2 \neq 0 \wedge e_2 \neq 0 \wedge g_2 \neq 0) \vee h_2 \neq 0 \vee h_1 = C_1 = 0 \end{cases}$
Positivity	$\{a \ldots k\}_{\{1 \ldots 4\}} \geq 0.$
Initial	$c_1 = e_1 = g_1 = 0, a_1 \neq 0, b_1 \neq 0, d_1 \neq 0, f_1 \neq 0, h_1 \neq 0$

Type checking and type inference

Theorem (Decidability of LLF type checking)

Let Γ be an LLF-context, t a term, T an LLF-type. One can decide in polynomial time whether $\Gamma \vdash t : T$ holds.

Type inference:
Given a system F judgement $\Gamma \vdash t : A$, the system of constraints generated from $(\Gamma^T + \mathcal{E})^0 \vdash_a t : A^T$ gives a polynomial-sized description of all LLF decorations of this judgement.
$\Rightarrow$ solution of the type inference problem.

Conclusion and perspectives

- we have given efficient, constraints-based algorithms for type-checking and type inference of system F Church terms in LLF.
  algorithmic system analogous to [GuerriniMartiniMasini08]
  method to build the algorithmic system:
  "decompose and clusterize" the ! rules.
- application also to S4 type systems (e.g. Pfenning-Wong'95)
- future work:
  remove restriction on the cut-rule
  could we infer also sharing of subterms ?