Type Inference in Elementary LL and cœrcions

Vincent Atassi

February 2006

Linear logic and implicit complexity

In linear logic, controle of ressources is made explicit, thanks to the ! and ? modalities : $A \Rightarrow B$ becomes $!A \multimap B$.
This opened the way to subsystems designed to have *intrinsic* and *implicit* complexity bounds : ELL and LLL.

Implicit complexity via λ-calculus and ELL

An implicit (no visible bound) characterisation of elementary time complexity class.

- Proof system with elementary time cut elimination, elementary counterpart to LLL ;
- Applied to λ-calculus as a type system.

$\Rightarrow$ Extensionnaly complete : enconding of elementary time turing machines.
$\Rightarrow$ Intensionnaly correct : a typeable term reduces in elementary time (tower fo exponentials of fixed height).

Implicit Computational Complexity & ELL

A *logical* characterisation of Elementary time.

- Might lead to a useful semantics of a bounded time type system
- System admits higher-order types
- Applies to Optimal Reduction : simplified algorithm

Some results on ELL

- If $\vdash_{\mathsf{ELL}} t : A$ then t reduces t a normal form in elementary time.
- Elementary recursive functions are typable in ELL
- If $\vdash_{\mathsf{ELL}} t : A$ then t can be reduced by Lamping's *abstract* algorithm.
- Type inference is decidable (for a subsystem)

Intensionnality problem

Despite the extensional completeness, an implicit complexity system can be too poor to be actually used, due to a lack of *expressive power*.

- Intensionnality , expressive power : accepting more algorithms
- In particular, ability to write *natural* algorithms for a function.

$\Rightarrow$ Extending the set of typable terms, witheout losing decidebility or bounded termination.

Example

The square function on integers :

- Natural algorithm : $\lambda n.(mult)n\ n$
- Typable algorithm : $\lambda n.(mult)\ (iter\ n\ succ\ 0)\ n$

$\Rightarrow$ Coercion problem : conversion from type N to $!N$ necessary as soon as polynomials are involved.

Outline

Issues adressed in this presentation :

- A polynomial algorithm for typing in propositional ELL
- Its implementation
- System extension with cœrcion rules

Core multiplicative affine logic

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash : B}\ cut$$

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ weak$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap R \qquad \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C} \multimap L$$

Fig.: Core implicative fragment of eal

Affine logic

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ cut$$

$$\frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ weak \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash: B}\ contr$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \multimap R \qquad \frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C} \multimap L$$

$$\frac{A_1, \ldots, A_n \vdash B}{!A_1, \ldots, !A_n \vdash !B}\ !$$

Fig.: Affine logic with exponential and contraction

$$\frac{}{x : A \vdash x : A}\ (\text{var}) \qquad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\ (\text{weak}) \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ (\text{abs})$$

$$\frac{\Gamma_1 \vdash M_1 : A \multimap B \qquad \Gamma_2 \vdash M_2 : A}{\Gamma_1, \Gamma_2 \vdash (M_1)\ M_2 : B}\ (\text{appl})$$

$$\frac{x_1 : !C, \ldots, x_n : !C, \Delta \vdash M : B}{x : !C, \Delta \vdash M\{x/x_1, \ldots, x/x_n\} : B}\ (\text{contr})$$

$$\frac{\Gamma_1 \vdash M_1 : !A_1 \ \ldots \Gamma_n \vdash M_n : !A_n \qquad x_1 : A_1, \ldots, x_n : A_n \vdash M : B}{\Gamma_1, \ldots, \Gamma_n \vdash M\{M_i/x_i\} : !B}\ (\text{prom})$$

Fig.: EAL^* type system

Syntax extension

Clings more closely to ELL derivations :

Pseudo-terms $t = x \mid \lambda x.t \mid (t_1)\, t_2 \mid !t \mid \bar{!}t$

Regular pseudo-terms $a = x|\lambda x.t|(t_1)t_2$; $t = !^n t$ with $n \in Z$

Parametrized types $u = \alpha|u \multimap u$ and $v = !^n u$

$\Rightarrow$ $Type(!^n t) = !^{n+m} T$ if $Type(t) = !^m T$

$\Rightarrow$ Since we will do type inference in the propositional fragment, we add restricted system-T constants (integers, iterator and addition).

$$\frac{}{x : A \vdash x : A}\ (\text{var}) \qquad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\ (\text{weak}) \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ (\text{abs})$$

$$\frac{\Gamma_1 \vdash M_1 : A \multimap B \qquad \Gamma_2 \vdash M_2 : A}{\Gamma_1, \Gamma_2 \vdash (M_1)\ M_2 : B}\ (\text{appl})$$

$$\frac{x_1 : !C, \ldots, x_n : !C, \Delta \vdash M : B}{x : !C, \Delta \vdash M\{x/x_1, \ldots, x/x_n\} : B}\ (\text{contr})$$

$$\frac{\Gamma_1 \vdash M_1 : !A_1 \ \ldots \Gamma_n \vdash M_n : !A_n \qquad x_1 : A_1, \ldots, x_n : A_n \vdash M : B}{\Gamma_1, \ldots, \Gamma_n \vdash !M\{\bar{!}M_i/x_i\} : !B}\ (\text{prom})$$

$$\frac{(t, c) \in \mathfrak{U}}{\vdash t : c}\ (\text{cst})$$

Fig.: EAL^* type system with constants on pseudo-term

Typability on regular pseudo-terms

Typability problem boils down to :

- Simple type assignement
- Types decoraton correction : no negative exponentials
- Linear typing condtion : ensuring linear decorations of simple types are consistent with respect to applications or abstractions
- Bracketing condition : well formdness of boxing, according to promotion rule of ELL
- Contraction condition : variables with several occurrences must be of a bang type

$\Rightarrow$ Rephrasing typability : simple type derivation + global criteria on integer parameters.

Typability to type inference

- Simple type inference
- Term and types free decoration : fresh parameters at all term nodes, fresh parameters for each subtype of occuring types
- Term explorations for *linear contraints* generation on those parameters
- Solvability $\Rightarrow$ typability and a solution is a type assignation

$\Rightarrow$ A regular pseudo-term t is typable in the restricted EAL* type system iff it has a simple type derivation and the generated constraints are satisfiable.

Constraints number analysis

For a lambda-term t et its simple type dérivation $\mathfrak{D}$. $|max(\mathfrak{D})|$ is size of the biggest types appearing in the type derivation.
(In)néquations generated for well-formdness of box placement :

brracketing Linear in $|t|$

Types consistency contraints :

types $\sum_{\forall T, T \in \mathfrak{D}} |T|$

Application $|t| \times |max(\mathfrak{D})|$

Others linear in $|t|$

$\Rightarrow$ Polynomial number of constraints

About constraints resolution

Solving the resulting linear program.

- Result : the set of solutions of the resulting constraints is closed under mulitplication by a positive integers : polynomial time solving
- In practice : simplex has proven to be the fastest
- Objective function ? Handling negative parameters and minimizing box nesting depth

$\Rightarrow$ This problem arises for typing in any bounded time subsystem of LL

$\Rightarrow$ Result interpretation gives a instantiated ELL type and box placement the input term

Illustration

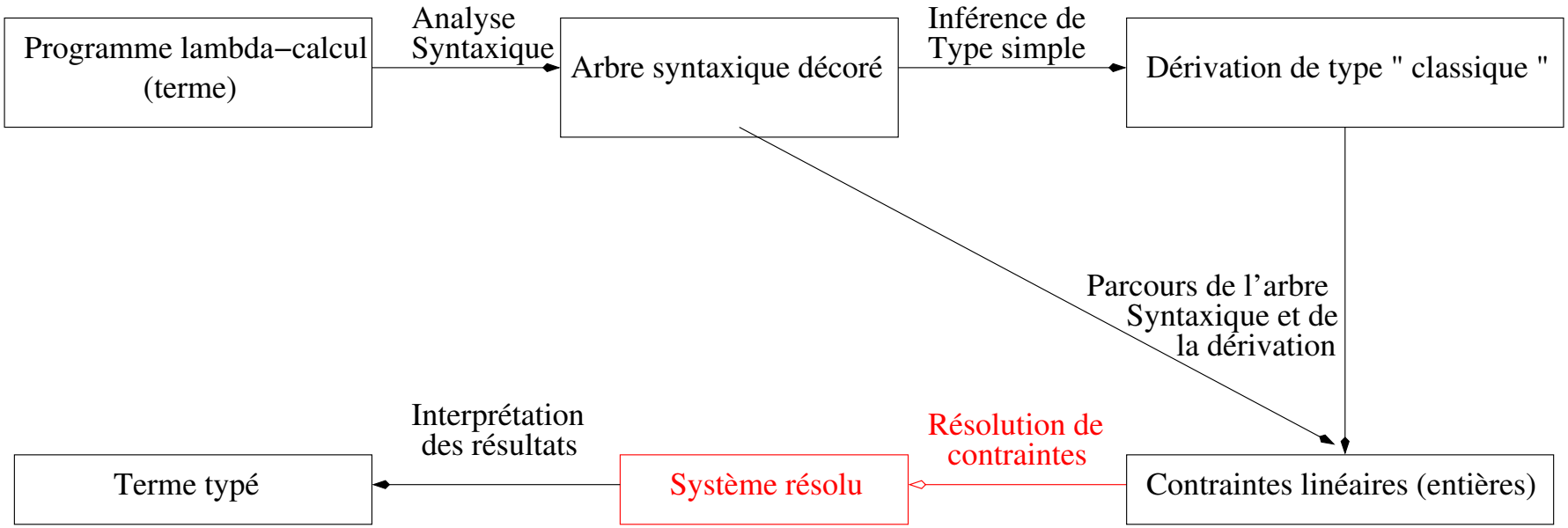

Fig.: Program overview

Exemple : $\lambda n.\lambda f.(n\ (n\ f))$

Illustration

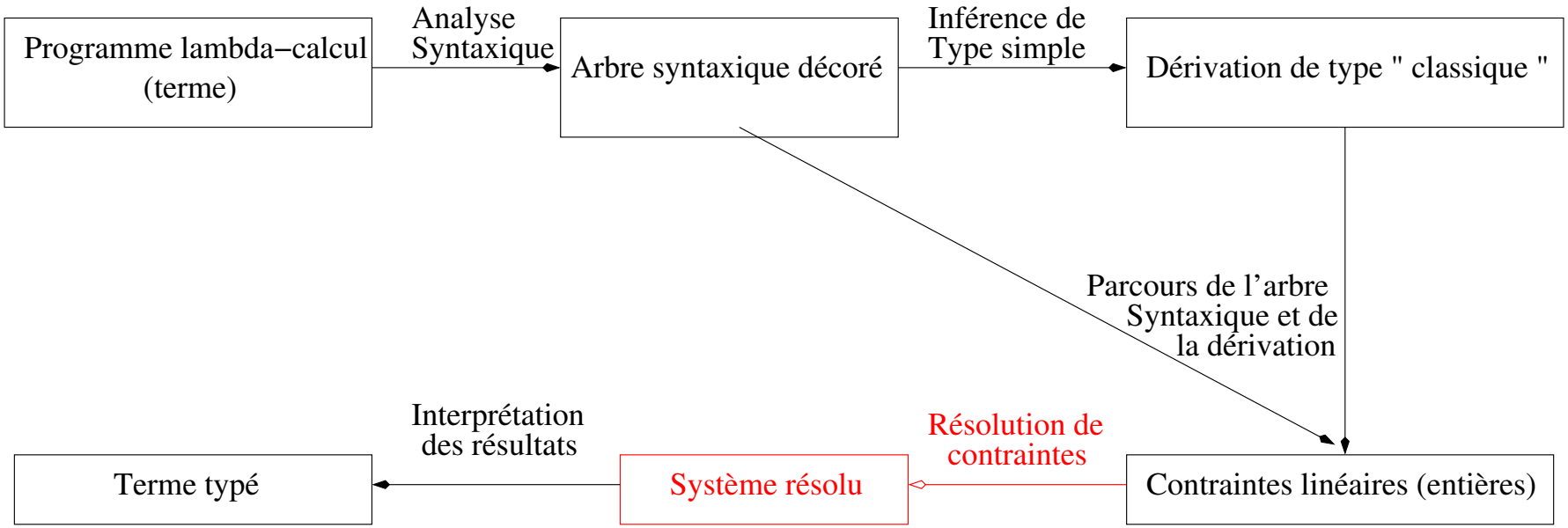

Fig.: Program overview

$$\lambda n.\lambda f.(n\ (n\ f)) : (A \to A) \to (A \to A)$$

Illustration

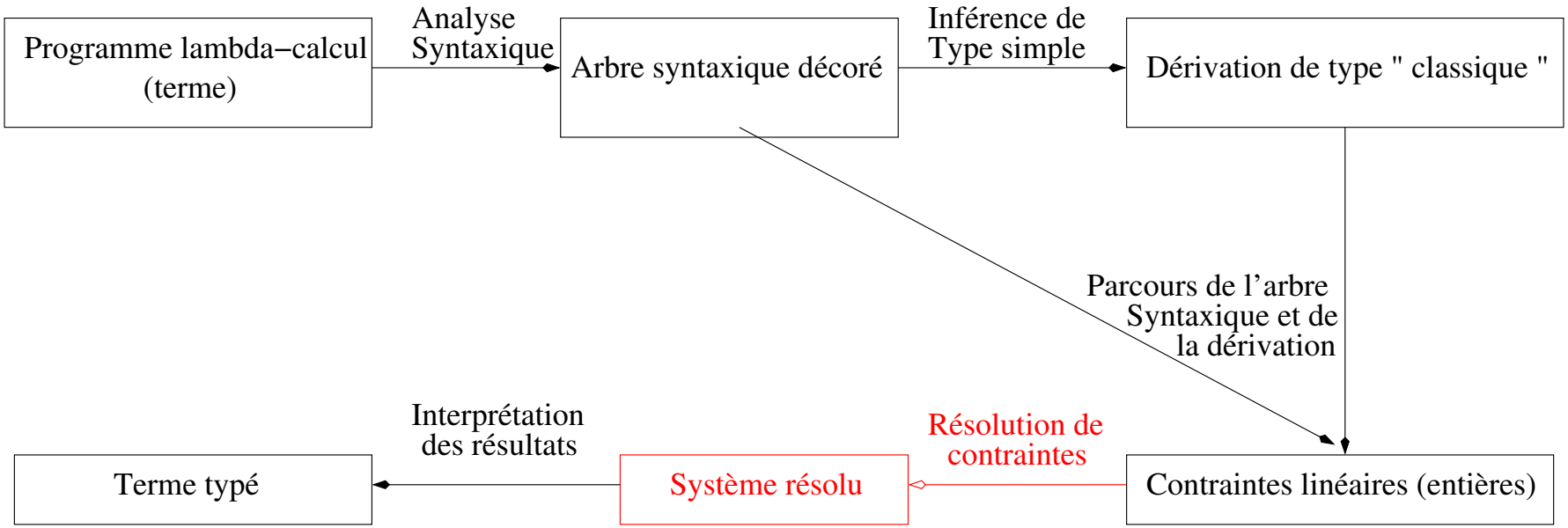

Fig.: Program overview

$$!^{n1}\lambda n.!^{n2}\lambda f.!^{n3}(!^{n4}n\ !^{n5}(!^{n6}n\ !^{n7}f)) :!^{n1}(!^{p1}(!^{p2}A \multimap !^{p3}A) \multimap !^{n2}(!^{p4}A \multimap !^{p3+n3}A))$$

Illustration

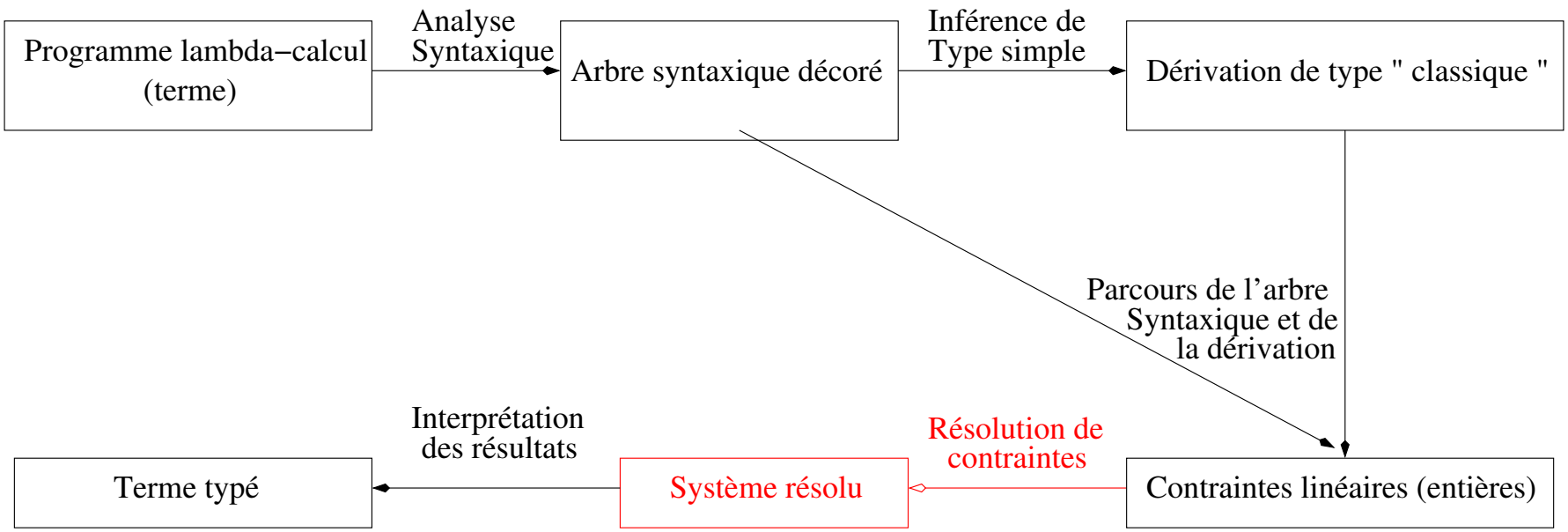

Fig.: Program overview

$$n3 + n4 \geq 0$$

$$\ldots$$

$$p2 - p6 - n6 = 0$$

$$\ldots$$

Illustration

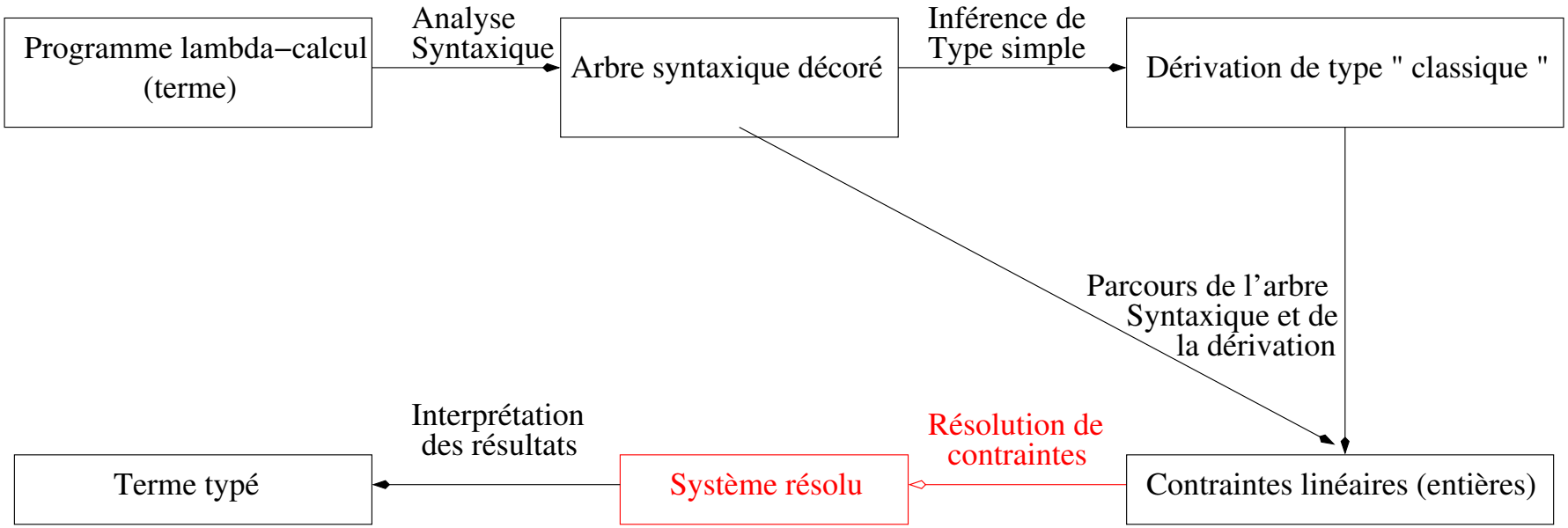

Fig.: Program overview

$$\lambda n.!\lambda f.(\bar{!}n\ (\bar{!}n\ f)) :!(A \to A) \to !(A \to A)$$

ELL and cœrcions

$\Rightarrow$ Using ELL proofs for encodings, placing, not solely promotion rules (boxes) with a special discipline, but also cœrcions, with the form of another explicit rule, is necessary.
$\Rightarrow$ Now, in the context of type inference, we automatized boxes placement. A further step is to automatize cœrcions placement.
$\Rightarrow$ The same way we enriched the input terms with boxes, we will enrich them with cœrcive subterms.

Typing with cœrcions

We have a derivable term t suc that $x : N \vdash t : !N$ is derivable, and another one for, say, booleans.
Placing cœrcions is adding *cuts* in proofs : sounds problematic for type inference.
But those cuts have *no computational value* : their only effect is to add modalities on positive sides of arrows, and to remove some on the negative side.
Hence the idea of a subtyping relation on types for expressing cœrcions : $N \leq !N$ and $!N \multimap N \leq N \multimap !N$.

A subtyping relation for cœrcions

We define first the subtyping relation for base types which admit cœrcions.

Definition

$\preceq$ *is defined by :*

$$\forall n, m \text{ st } 0 \leq n \leq m, !^n N \preceq !^m N \text{ and } !^n B \preceq !^m B$$

Proposition

$\preceq$ *is reflexive and transitive.*

Lemma (Correction of $\preceq$)

If $A \preceq B$, *then* $\exists t$ *such that* $x : A \vdash t : B$ *is derivable and* $t \longrightarrow_\beta^* x$.

Subtyping relation

$$\text{(base)}\ \frac{}{A \leq B}\ A \preceq B \qquad \text{(var)}\ \frac{}{\alpha \leq \alpha}$$

$$\text{(prom)}\ \frac{A \leq B}{!A \leq !B} \qquad \text{(arrow)}\ \frac{A_1 \geq A_2 \qquad B_1 \leq B_2}{A_1 \multimap B_1 \leq A_2 \multimap B_2}$$

Fig.: Subtyping system.

Lemma

$\leq$ *is reflexive and transitive.*

Subtyping relation

$$\text{(base)}\ \frac{}{A \leq B}\ A \preceq B \qquad \text{(var)}\ \frac{}{\alpha \leq \alpha}$$

$$\text{(prom)}\ \frac{A \leq B}{!A \leq !B} \qquad \text{(arrow)}\ \frac{A_1 \geq A_2 \qquad B_1 \leq B_2}{A_1 \multimap B_1 \leq A_2 \multimap B_2}$$

Fig.: Subtyping system.

Proposition ($\leq$'s correction)

If $A \leq B$, then exists t such that $x : A \vdash t : B$ is derivable in EAL *and $t \longrightarrow^*_{\beta\eta} x$.*

Overview of the proofs

- Base relation : successive cuts on a $coerc^{j,j+1}$ term
- Type relation : build a context to place the cuts correctly

$\Rightarrow$ gives a *costly* (in terms of evaluation time and boxe nesting) translation in eal

A cœrcions enriched type system

You add the wo following rules to EAL to obtain $\vdash_{\leq}$:

$$\frac{\Gamma, x : B \vdash t : C \quad A \leq B}{\Gamma, x : A \vdash t : C}\ coerc - L \qquad \frac{\Gamma \vdash t : A \quad A \leq B}{\Gamma \vdash t : B}\ coerc - R$$

The correction is given by :

Theorem

If $\Gamma \vdash_{\leq} t : A$ *then exists* t' *such that* $\Gamma \vdash_{\mathsf{EAL}} t' : A$ *is derivable and* $t' \longrightarrow^{*}_{\beta\eta} t$.

Démonstration.

Left side or right side cut with the term built from the subtyping derivation. □

We seek a equivalent type system suited for type inference.
Equivalence proofs :

- Coerc-l elimination : the contraction issue
- Coerc-r : not only at application level because of the promotion rule. Then commutes with other rules

Type inference with cœrcions

$$\frac{A \leq B}{x : A \vdash x : B}\ (\text{var}) \qquad \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}\ (\text{weak}) \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ (\text{abs})$$

$$\frac{\Gamma_1 \vdash M_1 : B \multimap C \qquad \Gamma_2 \vdash M_2 : A \qquad A \leq B \quad C \leq D}{\Gamma_1, \Gamma_2 \vdash (M_1\ M_2) : D}\ (\text{appl})$$

$$\frac{x_1 : !C, \ldots, x_n : !C, \Delta \vdash M : B \qquad A \leq !C}{x : A, \Delta \vdash M\{x/x_1, \ldots, x/x_n\} : B}\ (\text{contr})$$

$$\frac{\Gamma_1 \vdash M_1 : !A_1 \ \ldots \Gamma_n \vdash M_n : !A_n \qquad x_1 : A_1, \ldots, x_n : A_n \vdash M : B}{\Gamma_1, \ldots, \Gamma_n \vdash !M\{\bar{!}M_i/x_i\} : !B}\ (\text{prom})$$

Fig.: EAL^*_A inference-driven type system

Related Works

Fields in time-bounded subsytems of linear logic.

- Linear Logic ' !' and application to Optimal reduction : [Asp94]
- Decidability of type inference in EAL and application to Optimal Reduction study : [ACM00, CM03, CRDR05].
- Call-by-value and eal : another system derived from ELL, with CBV reduction, enjoying type inference decidability, subject reduction and completness via a flexible system of constants : [CDLRDR05]
- Light Logics : LLL/LAL/DLAL systems of polynomial time, with decidable system F type decoration : [BT04].
- Light Logics : SLL : [Laf04]

Conclusion & perspectives

From EAL to LLL/LAL/DLAL

- Feasible algorithm for Light logics
- Coercions for light logics

Inference enhancement :

- Modularity for type inference
- Exploiting relation with Optimal Reduction


Andrea Asperti, Paolo Coppola, and Simone Martini.
(optimal) duplication is not elementary recursive.
In *POPL '00 : Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 96–107, New York, NY, USA, 2000. ACM Press.


Andrea Asperti.
Linear logic, commonads, and optimal reduction.
Fundamenta Informaticæ, 22(1), 1994.


Patrick Baillot and Kazushige Terui.
Light types for polynomial time computation in lambda-calculus.
In *LICS '04 : Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)*, pages 266–275, Washington, DC, USA, 2004. IEEE Computer Society.


Paolo Coppola, Ugo Dal Lago, and Simona Ronchi Della Rocca.

Elementary affine logic and the call by value lambda calculus.
In Pawel Urzyczyn, editor, *TLCA'05*, volume 3461 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2005.


Paolo Coppola and Simone Martini.
Optimizing optimal reduction : A type inference algorithm for elementary affine logic.
CoRR, cs.LO/0305011, 2003.


P. Coppola and Simona Ronchi Della Rocca.
Principal typing for lambda calculus in elementary affine logic.
Fundamenta Informaticae, 65(1-2) :87–112, 2005.


Vincent Danos and Jean-Baptiste Joinet.
1st international workshop on implicit computational complexity (icc'99).
Santa Barbara, California, 1999.


Yves Lafont.
Soft linear logic and polynomial time.
Theor. Comput. Sci., 318(1-2) :163–180, 2004.