Verification of Ptime Reducibility for System F terms Via Dual Light Affine Logic

Kazushige Terui

National Institute of Informatics, Japan

joint work with

Vincent Atassi $\otimes$ Patrick Baillot

LIPN CNRS University Paris 13

Characterizations of Ptime

- Explicit characterization of Ptime functions:

$$f \in \mathbf{FP} \iff \exists \text{ Turing Machine } M$$

$$M \text{ is P-clocked and } M \text{ computes } f$$

Characterizations of Ptime

- Explicit characterization of Ptime functions:

$$f \in \mathbf{FP} \iff \exists \text{ Turing Machine } M$$

$$M \text{ is P-clocked and } M \text{ computes } f$$

- Implicit characterizations of Ptime functions:

Characterizations of Ptime

- Explicit characterization of Ptime functions:

$$f \in \mathbf{FP} \iff \exists \text{ Turing Machine } M$$

$$M \text{ is P-clocked and } M \text{ computes } f$$

- Implicit characterizations of Ptime functions:
  - Replace Turing Machines with higher model of computation

Characterizations of Ptime

- Explicit characterization of Ptime functions:

$$f \in \mathbf{FP} \iff \exists \text{ Turing Machine } M$$

$$M \text{ is P-clocked and } M \text{ computes } f$$

- Implicit characterizations of Ptime functions:
  - Replace Turing Machines with higher model of computation
  - Replace P-clocked with structural/logical conditions

$$f \in \mathbf{FP} \iff \exists\, \mathcal{PROG}\ M$$

$$M \text{ satisfies } \mathcal{COND} \text{ and } M \text{ computes } f$$

Characterizations of Ptime

- Implicit characterizations of Ptime functions:

$$f \in \mathbf{FP} \iff \exists\, \mathcal{PROG}\ M$$

M satisfies $\mathcal{COND}$ and M computes f

- Various approaches:

$\mathcal{PROG}$	$\mathcal{COND}$	
Primitive Recursion	Safety	(Bellantoni-Cook, Leivant, ...)
Term Rewriting	PO + Quasi-Interpretation	(Marion-Moyen, Bonfante, ...)
System T	Safe-Linear Types	(Hofmann, Schwichtenberg...)
Proof Nets	LLL/LAL Types	(Girard, Asperti, ...)
System F	DLAL Types	(Baillot-T. LICS04)

Complexity Verification

- In complexity verification, more relevant is

Complexity Verification

- In complexity verification, more relevant is
- Explicit characterization of Ptime programs:

$$M \in \mathbf{P}\text{-}\mathcal{PROG} \iff \exists \text{ degree } n \; \forall \text{ input } w$$
$$M(w) \text{ terminates in time } O(|w|^n)$$

Complexity Verification

- In complexity verification, more relevant is
- Explicit characterization of Ptime programs:

$$M \in \mathbf{P}\text{-}\mathcal{PROG} \iff \exists \text{ degree } n \; \forall \text{ input } w$$
$$M(w) \text{ terminates in time } O(|w|^n)$$

- $\mathbf{P}\text{-}\mathcal{PROG}$ is Σ_2-complete!

Complexity Verification

- In complexity verification, more relevant is
- Explicit characterization of Ptime programs:

$$M \in \mathbf{P}\text{-}\mathcal{PROG} \iff \exists \text{ degree } n \; \forall \text{ input } w$$
$$M(w) \text{ terminates in time } O(|w|^n)$$

- $\mathbf{P}\text{-}\mathcal{PROG}$ is Σ_2-complete!
- Impossible to characterize by, e.g., a “natural” type system (which is usually Σ_1).

Complexity Verification

- Nevertheless, ICC is useful to provide a good approximation of $\mathbf{P}$-$\mathcal{PROG}$:

$$M \in \mathbf{P}\text{-}\mathcal{PROG} \quad \Longleftarrow \quad M \text{ satisfies } \mathcal{COND}$$

- To be practically useful,
  - $\mathcal{PROG}$ must be a natural, expressive model of computation.
  - $\mathcal{COND}$ must admit many algorithms.
  - Complexity of $\mathcal{COND}$ is in question.

In This Talk

- $\mathcal{PROG}$ = System F:
  - reference system for studying polymorphic functional programming languages
  - various data types are uniformly definable

In This Talk

- $\mathcal{PROG}$ = System F:
  - reference system for studying polymorphic functional programming languages
  - various data types are uniformly definable
- $\mathcal{COND}$ = Dual Light Affine Logic (DLAL)
  - A refinement of Light Linear Logic. A type system for System F lambda terms that ensures Ptime normalization.

In This Talk

- $\mathcal{PROG}$ = System F:
  - reference system for studying polymorphic functional programming languages
  - various data types are uniformly definable
- $\mathcal{COND}$ = Dual Light Affine Logic (DLAL)
  - A refinement of Light Linear Logic. A type system for System F lambda terms that ensures Ptime normalization.
- Main Result: Given a system F term M, it is decidable in Ptime whether M is typable in DLAL.
  - Typing guarantees that M works in Ptime.
  - The algorithm is already implemented.

Outline

- Background: System F
- From Linear Logic to DLAL
- Main Result
- Proof Idea
- Example/Implementation
- Conclusion

Background: System F typing

- Types of System F:

$$T, U ::= \alpha \mid T \to U \mid \forall \alpha . T$$

- (Explicitly-typed) terms of system F:

$$x^T \qquad (\lambda x^T . M^U)^{T \to U} \qquad ((M^{T \to U}) N^T)^U$$

$$(\Lambda \alpha . M^U)^{\forall \alpha . U} \qquad ((M^{\forall \alpha . U}) T)^{U[T/\alpha]}$$

with condition: in $\Lambda \alpha . M^U$, α may not occur freely in the types of free term variables of M (the eigenvariable condition).

Linear Logic

- Decomposition of $A \to B$ into $!A \multimap B$.

Linear Logic

- Decomposition of $A \to B$ into $!A \multimap B$.
- $t : A \multimap B$ uses data $a : A$ exactly once.

Linear Logic

- Decomposition of $A \to B$ into $!A \multimap B$.
- $t : A \multimap B$ uses data $a : A$ exactly once.
- $a : !A$ can be duplicated.

$$dup : !A \multimap !A \otimes !A$$

Linear Logic

- Decomposition of $A \to B$ into $!A \multimap B$.
- $t : A \multimap B$ uses data $a : A$ exactly once.
- $a : !A$ can be duplicated.

$$dup : !A \multimap !A \otimes !A$$

- Modality ! is S4.

Linear Logic $\approx$ linear lambda calculus +

duplication controlled by S4-modality.

Light Linear Logic

- Change the modality from S4 to K-bounded monotone one.

$$\frac{A \multimap B \multimap C}{\S A \multimap \S B \multimap \S C} \qquad \frac{A \multimap B}{!A \multimap !B} \qquad \frac{}{!A \multimap \S A}$$

Light Linear Logic $\approx$ linear lambda calculus +

K-bounded monotone duplication.

Light Linear Logic

- Change the modality from S4 to K-bounded monotone one.

$$\frac{A \multimap B \multimap C}{\S A \multimap \S B \multimap \S C} \qquad \frac{A \multimap B}{!A \multimap !B} \qquad \frac{}{!A \multimap \S A}$$

Light Linear Logic $\approx$ linear lambda calculus +

K-bounded monotone duplication.

- Why is § a K-modality?
  - It enforces a stratified structure on proofs.
  - Layers are strictly separated:

$$\S A \not\multimap A \qquad \S A \not\multimap \S\S A$$

  - It allows "layer-by-layer normalization"

Light Linear Logic

- Why isn't ! a K-modality?

$$\dfrac{\dfrac{t : A \multimap A \multimap A}{t : !A \multimap !A \multimap !A}}{\lambda x.txx : !A \multimap !A}$$

$(\lambda x.txx)^n a \longrightarrow^*$ exponentially many a's

Light Linear Logic

- Why isn't ! a K-modality?
  - If it were, iteration would cause exponential blow-up at one layer

$$\dfrac{\dfrac{t : A \multimap A \multimap A}{t : !A \multimap !A \multimap !A}}{\lambda x.txx : !A \multimap !A}$$

$(\lambda x.txx)^n a \longrightarrow^*$ exponentially many a's

Light Linear Logic

- Why isn't ! a K-modality?
  - If it were, iteration would cause exponential blow-up at one layer

$$\dfrac{\dfrac{t : A \multimap A \multimap A}{t : !A \multimap !A \multimap !A}}{\lambda x.txx : !A \multimap !A}$$

$(\lambda x.txx)^n a \longrightarrow^*$ exponentially many a's

- With ! non-K, the blow-up is at most quadratic.

Light Linear Logic

- Why isn't ! a K-modality?
  - If it were, iteration would cause exponential blow-up at one layer

$$\dfrac{\dfrac{t : A \multimap A \multimap A}{t : !A \multimap !A \multimap !A}}{\lambda x.txx : !A \multimap !A}$$

$(\lambda x.txx)^n a \longrightarrow^*$ exponentially many a's

- With ! non-K, the blow-up is at most quadratic.
- Main Property: Proof net π of depth d normalizes in $|\pi|^{2^d}$ steps.

Light Linear Logic

- Why isn't ! a K-modality?
  - If it were, iteration would cause exponential blow-up at one layer

$$\dfrac{\dfrac{t : A \multimap A \multimap A}{t : !A \multimap !A \multimap !A}}{\lambda x.txx : !A \multimap !A}$$

$(\lambda x.txx)^n a \longrightarrow^*$ exponentially many a's

- With ! non-K, the blow-up is at most quadratic.
- Main Property: Proof net π of depth d normalizes in $|\pi|^{2^d}$ steps.
- Light Affine Logic (Asperti 98): (Intuitionistic) LLL with full weakening.

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap !B \quad a : A}{fa : !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap !B \quad a : A}{fa : !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

- Terms of type $!B$ are sharable, but not all of them are duplicable.

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap !B \quad a : A}{fa : !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

- Terms of type $!B$ are sharable, but not all of them are duplicable.
- λ-calculus confuses them. So it leads to bad exponential blow-up by β-reduction.

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap\, !B \quad a : A}{fa :\, !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

- Terms of type $!B$ are sharable, but not all of them are duplicable.
- λ-calculus confuses them. So it leads to bad exponential blow-up by β-reduction.
- Two solutions:

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap !B \quad a : A}{fa : !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

- Terms of type $!B$ are sharable, but not all of them are duplicable.
- λ-calculus confuses them. So it leads to bad exponential blow-up by β-reduction.
- Two solutions:
  - Use a syntax with explicit sharing mechanism

Difficulty of LLL/LAL

- Lambda terms typable in LLL/LAL do not normalize in Ptime by β-reduction.

$$\frac{f : A \multimap !B \quad a : A}{fa : !B} \qquad dup(fa) \longrightarrow (fa, fa)$$

- Terms of type $!B$ are sharable, but not all of them are duplicable.
- λ-calculus confuses them. So it leads to bad exponential blow-up by β-reduction.
- Two solutions:
  - Use a syntax with explicit sharing mechanism
  - Fobid $A \multimap !B$.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.
- Solution: Forbid $\S!A$.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.
- Solution: Forbid $\S!A$.
- Then, types look like $\S \cdots \S A$ or $!\S \cdots \S A$. Thus type inference boils down to solving (boolean and) integer-constraints.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.
- Solution: Forbid $\S!A$.
- Then, types look like $\S \cdots \S A$ or $!\S \cdots \S A$. Thus type inference boils down to solving (boolean and) integer-constraints.
- We also forbid $\forall \alpha.!A$.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.
- Solution: Forbid $\S!A$.
- Then, types look like $\S \cdots \S A$ or $!\S \cdots \S A$. Thus type inference boils down to solving (boolean and) integer-constraints.
- We also forbid $\forall\alpha.!A$.
- ! is only used as $!A \multimap B$.

Difficulty of LLL/LAL

- Typability in (propositional) LAL is decidable (Baillot 02), but is extremely complicated, because of types like

$$\S!\S\S!!!\S \cdots \S!A$$

- Type inference involves solving word-constraints over $\{!, \S\}^*$.
- Solution: Forbid $\S!A$.
- Then, types look like $\S \cdots \S A$ or $!\S \cdots \S A$. Thus type inference boils down to solving (boolean and) integer-constraints.
- We also forbid $\forall\alpha.!A$.
- ! is only used as $!A \multimap B$.
- Then why don't you come back to $A \Rightarrow B$? — **DLAL**.

Dual Light Affine Logic

- DLAL seen as a refined type system for system F terms.

Dual Light Affine Logic

- DLAL seen as a refined type system for system F terms.
- Types of DLAL:

$$A, B ::= \alpha \mid A \multimap B \mid A \Rightarrow B \mid \S A \mid \forall \alpha . A$$

Dual Light Affine Logic

- DLAL seen as a refined type system for system F terms.
- Types of DLAL:

$$A, B ::= \alpha \mid A \multimap B \mid A \Rightarrow B \mid \S A \mid \forall \alpha . A$$

- The erasure map $(.)^-$ to System F types:

$$(\S A)^- = A^-, \quad (A \multimap B)^- = (A \Rightarrow B)^- = A^- \to B^-.$$

Dual Light Affine Logic

- DLAL seen as a refined type system for system F terms.
- Types of DLAL:

$$A, B ::= \alpha \mid A \multimap B \mid A \Rightarrow B \mid \S A \mid \forall \alpha . A$$

- The erasure map $(.)^-$ to System F types:

$$(\S A)^- = A^-, \quad (A \multimap B)^- = (A \Rightarrow B)^- = A^- \to B^-.$$

- A is a decoration of a system F type T if $A^- = T$.

Dual Light Affine Logic

- DLAL seen as a refined type system for system F terms.
- Types of DLAL:

$$A, B ::= \alpha \mid A \multimap B \mid A \Rightarrow B \mid \S A \mid \forall \alpha . A$$

- The erasure map $(.)^-$ to System F types:

$$(\S A)^- = A^-, \quad (A \multimap B)^- = (A \Rightarrow B)^- = A^- \to B^-.$$

- A is a decoration of a system F type T if $A^- = T$.
- Judgements: of the form $\Gamma;\Delta \vdash M : A$, where M is a system F term, Γ contains non-linear variables, and Δ linear variables.

Dual Light Affine Logic

$$\frac{}{;\, x^{A^-} : A \vdash x^{A^-} : A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1, x : A \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1, x : A;\Delta_1 \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M : A}\ \text{(Weak)}$$

$$\frac{;\Gamma, x_1 : B_1, \ldots, x_n : B_n \vdash M : A}{\Gamma; x_1 : \S B_1, \ldots, x_n : \S B_n \vdash M : \S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1;\Delta_1 \vdash \Lambda\alpha.M : \forall\alpha.A}\ (\forall \text{ i) (*)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \multimap B \quad \Gamma_2;\Delta_2 \vdash N : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash (M\ N) : B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \Rightarrow B \quad ;z : C \vdash N : A}{\Gamma_1, z : C;\Delta_1 \vdash (M\ N) : B}\ (\Rightarrow \text{ e})$$

$$\frac{x_1 : A, x_2 : A, \Gamma_1;\Delta_1 \vdash M : B}{x : A, \Gamma_1;\Delta_1 \vdash M[x/x_1, x/x_2] : B}\ \text{(Cntr)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash N : \S A \quad \Gamma_2; x : \S A, \Delta_2 \vdash M : B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M[N/x] : B}\ (\S \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : \forall\alpha.A}{\Gamma_1;\Delta_1 \vdash MB^- : A[B/\alpha]}\ (\forall \text{ e})$$

Dual Light Affine Logic

$$\frac{}{;x^{A^-}:A \vdash x^{A^-}:A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1,x:A\vdash M:B}{\Gamma_1;\Delta_1\vdash \lambda x^{A^-}.M:A\multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A\multimap B \quad \Gamma_2;\Delta_2\vdash N:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash (M\ N):B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1,x:A;\Delta_1\vdash M:B}{\Gamma_1;\Delta_1\vdash \lambda x^{A^-}.M:A\Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A\Rightarrow B \quad ;z:C\vdash N:A}{\Gamma_1,z:C;\Delta_1\vdash (M\ N):B}\ (\Rightarrow \text{ e})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash M:A}\ \text{(Weak)}$$

$$\frac{x_1:A,x_2:A,\Gamma_1;\Delta_1\vdash M:B}{x:A,\Gamma_1;\Delta_1\vdash M[x/x_1,x/x_2]:B}\ \text{(Cntr)}$$

$$\frac{;\Gamma,x_1:B_1,\ldots,x_n:B_n\vdash M:A}{\Gamma;x_1:\S B_1,\ldots,x_n:\S B_n\vdash M:\S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash N:\S A \quad \Gamma_2;x:\S A,\Delta_2\vdash M:B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash M[N/x]:B}\ (\S \text{ e})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A}{\Gamma_1;\Delta_1\vdash \Lambda\alpha.M:\forall\alpha.A}\ (\forall \text{ i) (*)}$$

$$\frac{\Gamma_1;\Delta_1\vdash M:\forall\alpha.A}{\Gamma_1;\Delta_1\vdash MB^-:A[B/\alpha]}\ (\forall \text{ e})$$

Dual Light Affine Logic

$$\frac{}{;\, x^{A^-} : A \vdash x^{A^-} : A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1, x : A \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1, x : A;\Delta_1 \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M : A}\ \text{(Weak)}$$

$$\frac{;\Gamma, x_1 : B_1, \ldots, x_n : B_n \vdash M : A}{\Gamma; x_1 : \S B_1, \ldots, x_n : \S B_n \vdash M : \S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1;\Delta_1 \vdash \Lambda\alpha.M : \forall\alpha.A}\ (\forall \text{ i) (*)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \multimap B \quad \Gamma_2;\Delta_2 \vdash N : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash (M\ N) : B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \Rightarrow B \quad ;z : C \vdash N : A}{\Gamma_1, z : C;\Delta_1 \vdash (M\ N) : B}\ (\Rightarrow \text{ e})$$

$$\frac{x_1 : A, x_2 : A, \Gamma_1;\Delta_1 \vdash M : B}{x : A, \Gamma_1;\Delta_1 \vdash M[x/x_1, x/x_2] : B}\ \text{(Cntr)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash N : \S A \quad \Gamma_2; x : \S A, \Delta_2 \vdash M : B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M[N/x] : B}\ (\S \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : \forall\alpha.A}{\Gamma_1;\Delta_1 \vdash MB^- : A[B/\alpha]}\ (\forall \text{ e})$$

$(\multimap i)$ (resp. $(\Rightarrow i)$) corresponds to abstraction on a linear (resp. non-linear) variable,

Dual Light Affine Logic

$$\frac{}{;x^{A^-}:A \vdash x^{A^-}:A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1,x:A \vdash M:B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M:A \multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M:A \multimap B \quad \Gamma_2;\Delta_2 \vdash N:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash (M\ N):B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1,x:A;\Delta_1 \vdash M:B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M:A \Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M:A \Rightarrow B \quad ;z:C \vdash N:A}{\Gamma_1,z:C;\Delta_1 \vdash (M\ N):B}\ (\Rightarrow \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M:A}\ \text{(Weak)}$$

$$\frac{x_1:A,x_2:A,\Gamma_1;\Delta_1 \vdash M:B}{x:A,\Gamma_1;\Delta_1 \vdash M[x/x_1,x/x_2]:B}\ \text{(Cntr)}$$

$$\frac{;\Gamma,x_1:B_1,\ldots,x_n:B_n \vdash M:A}{\Gamma;x_1:\S B_1,\ldots,x_n:\S B_n \vdash M:\S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash N:\S A \quad \Gamma_2;x:\S A,\Delta_2 \vdash M:B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M[N/x]:B}\ (\S \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M:A}{\Gamma_1;\Delta_1 \vdash \Lambda\alpha.M:\forall\alpha.A}\ (\forall \text{ i) (*)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash M:\forall\alpha.A}{\Gamma_1;\Delta_1 \vdash MB^-:A[B/\alpha]}\ (\forall \text{ e})$$

an argument N of a term M of type $A \Rightarrow B$ must have at most one occurrence z of free variable, which is linear.

Dual Light Affine Logic

$$\frac{}{;x^{A^-}:A\vdash x^{A^-}:A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1,x:A\vdash M:B}{\Gamma_1;\Delta_1\vdash\lambda x^{A^-}.M:A\multimap B}\ (\multimap\text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A\multimap B\quad\Gamma_2;\Delta_2\vdash N:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash(M\ N):B}\ (\multimap\text{ e})$$

$$\frac{\Gamma_1,x:A;\Delta_1\vdash M:B}{\Gamma_1;\Delta_1\vdash\lambda x^{A^-}.M:A\Rightarrow B}\ (\Rightarrow\text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A\Rightarrow B\quad;z:C\vdash N:A}{\Gamma_1,z:C;\Delta_1\vdash(M\ N):B}\ (\Rightarrow\text{ e})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash M:A}\ \text{(Weak)}$$

$$\frac{x_1:A,x_2:A,\Gamma_1;\Delta_1\vdash M:B}{x:A,\Gamma_1;\Delta_1\vdash M[x/x_1,x/x_2]:B}\ \text{(Cntr)}$$

$$\frac{;\Gamma,x_1:B_1,\ldots,x_n:B_n\vdash M:A}{\Gamma;x_1:\S B_1,\ldots,x_n:\S B_n\vdash M:\S A}\ (\S\text{ i})$$

$$\frac{\Gamma_1;\Delta_1\vdash N:\S A\quad\Gamma_2;x:\S A,\Delta_2\vdash M:B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2\vdash M[N/x]:B}\ (\S\text{ e})$$

$$\frac{\Gamma_1;\Delta_1\vdash M:A}{\Gamma_1;\Delta_1\vdash\Lambda\alpha.M:\forall\alpha.A}\ (\forall\text{ i})\ (*)$$

$$\frac{\Gamma_1;\Delta_1\vdash M:\forall\alpha.A}{\Gamma_1;\Delta_1\vdash MB^-:A[B/\alpha]}\ (\forall\text{ e})$$

the rule (§ i) allows to turn linear variables (in Γ) into non-linear ones.

Dual Light Affine Logic

$$\frac{}{;\, x^{A^-} : A \vdash x^{A^-} : A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1, x : A \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1, x : A;\Delta_1 \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M : A}\ \text{(Weak)}$$

$$\frac{;\Gamma, x_1 : B_1, \ldots, x_n : B_n \vdash M : A}{\Gamma; x_1 : \S B_1, \ldots, x_n : \S B_n \vdash M : \S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1;\Delta_1 \vdash \Lambda\alpha.M : \forall\alpha.A}\ (\forall \text{ i) (*)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \multimap B \quad \Gamma_2;\Delta_2 \vdash N : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash (M\ N) : B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \Rightarrow B \quad ;z : C \vdash N : A}{\Gamma_1, z : C;\Delta_1 \vdash (M\ N) : B}\ (\Rightarrow$$

$$\frac{x_1 : A, x_2 : A, \Gamma_1;\Delta_1 \vdash M : B}{x : A, \Gamma_1;\Delta_1 \vdash M[x/x_1, x/x_2] : B}\ \text{(Cntr)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash N : \S A \quad \Gamma_2; x : \S A, \Delta_2 \vdash M : B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M[N/x] : B}\ (\S$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : \forall\alpha.A}{\Gamma_1;\Delta_1 \vdash MB^- : A[B/\alpha]}\ (\forall \text{ e})$$

Dual Light Affine Logic

$$\frac{}{;\, x^{A^-} : A \vdash x^{A^-} : A}\ \text{(Id)}$$

$$\frac{\Gamma_1;\Delta_1, x : A \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \multimap B}\ (\multimap \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \multimap B \quad \Gamma_2;\Delta_2 \vdash N : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash (M\ N) : B}\ (\multimap \text{ e})$$

$$\frac{\Gamma_1, x : A;\Delta_1 \vdash M : B}{\Gamma_1;\Delta_1 \vdash \lambda x^{A^-}.M : A \Rightarrow B}\ (\Rightarrow \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A \Rightarrow B \quad ;z : C \vdash N : A}{\Gamma_1, z : C;\Delta_1 \vdash (M\ N) : B}\ (\Rightarrow \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M : A}\ \text{(Weak)}$$

$$\frac{x_1 : A, x_2 : A, \Gamma_1;\Delta_1 \vdash M : B}{x : A, \Gamma_1;\Delta_1 \vdash M[x/x_1, x/x_2] : B}\ \text{(Cntr)}$$

$$\frac{;\Gamma, x_1 : B_1, \ldots, x_n : B_n \vdash M : A}{\Gamma; x_1 : \S B_1, \ldots, x_n : \S B_n \vdash M : \S A}\ (\S \text{ i})$$

$$\frac{\Gamma_1;\Delta_1 \vdash N : \S A \quad \Gamma_2; x : \S A, \Delta_2 \vdash M : B}{\Gamma_1,\Gamma_2;\Delta_1,\Delta_2 \vdash M[N/x] : B}\ (\S \text{ e})$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : A}{\Gamma_1;\Delta_1 \vdash \Lambda\alpha.M : \forall\alpha.A}\ (\forall \text{ i}) \text{ (*)}$$

$$\frac{\Gamma_1;\Delta_1 \vdash M : \forall\alpha.A}{\Gamma_1;\Delta_1 \vdash MB^- : A[B/\alpha]}\ (\forall \text{ e})$$

Depth d of a derivation $\mathcal{D}$: maximal number of (§ i) and r.h.s. premises of ($\Rightarrow$ e) in a branch of $\mathcal{D}$.

Data types in DLAL

- Data types in DLAL:

$$
\begin{aligned}
N &= \forall\alpha.(\alpha \multimap \alpha) \Rightarrow \S(\alpha \multimap \alpha) \\
W &= \forall\alpha.(\alpha \multimap \alpha) \Rightarrow (\alpha \multimap \alpha) \Rightarrow \S(\alpha \multimap \alpha) \\
L(A) &= \forall\alpha.(A \multimap \alpha \multimap \alpha) \Rightarrow \S(\alpha \multimap \alpha)
\end{aligned}
$$

Inhabitants:

$$
\begin{aligned}
\underline{3} &= \Lambda\alpha.\lambda f^{\alpha\to\alpha}.\lambda x^{\alpha}.f(f(fx)) && : N \\
\underline{010} &= \Lambda\alpha\lambda f_0^{\alpha\to\alpha}.\lambda f_1^{\alpha\to\alpha}.\lambda x^{\alpha}.f_0(f_1(f_0x)) && : W
\end{aligned}
$$

- These are decorations of system F data types.

Examples of terms

$$N = \forall\alpha.(\alpha \multimap \alpha) \Rightarrow \S(\alpha \multimap \alpha)$$

$$add = \lambda n.\lambda m.\Lambda\alpha.\lambda f.\lambda x.(n\alpha\ f\ (m\alpha\ f\ x)) \quad : \quad N \multimap N \multimap N$$

$$mult = \lambda n.\lambda m.(m(N \to N)\ (add\ n))\ \underline{0} \quad : \quad N \Rightarrow N \multimap \S N$$

$$square \quad : \quad N \multimap \S^2 N$$

Example 1: reverse

a *reverse* function on binary lists

$$\lambda l^{W}.\Lambda\beta.\lambda so^{\beta\to\beta}.\lambda si^{\beta\to\beta}.(l\ (\beta\to\beta))$$
$$\lambda a^{\beta\to\beta}.\lambda x^{\beta}.(a)(so)x$$
$$\lambda a^{\beta\to\beta}.\lambda x^{\beta}.(a)(si)x\ (\Lambda\alpha.\lambda z^{\alpha}.z)\beta \quad : W\to W$$

This term is well-typed in F and typable in DLAL with type: $W^{DLAL} \multimap W^{DLAL}$.

Example 2: insertion sort

assume given

$comp : A \otimes A \multimap A \otimes A$, with $(comp\ a_1\ a_2) \longrightarrow a_1 \otimes a_2$ if $a_1 \leq a_2$

$a_2 \otimes a_1$ if $a_2 \leq a_1$

- **insertion function**:
  we define it by iteration on type $B = A \multimap (\alpha \multimap \alpha)$:
  idea:

$$insert\ (a' :: l, a) = \text{let}\ comp\ a\ a'\ \text{be}\ a_1 \otimes a_2\ \text{in}\ a_1 :: insert(l, a_2)$$

$$insert(nil, a) = a :: nil$$

Example 2: insertion sort

assume given

$comp : A \otimes A \multimap A \otimes A$, with $(comp\ a_1\ a_2) \longrightarrow a_1 \otimes a_2$ if $a_1 \leq a_2$

$a_2 \otimes a_1$ if $a_2 \leq a_1$

- **insertion function**:
  we define it by iteration on type $B = A \multimap (\alpha \multimap \alpha)$:
  step $t : A \multimap B \multimap B$ and base $u : B$ given by

$$t \quad = \quad \lambda a^A f^B a'^A.$$

$$\text{let } comp\ a\ a' \text{ be } a_1 \otimes a_2 \text{ in}$$

$$\lambda x^\alpha.(s^B\ a_1{}^A\ (f\ a_2\ x)^\alpha)^\alpha \quad : A \multimap B \multimap B$$

$$u \quad = \quad \lambda a^A.(s^B\ a) \quad : B$$

  observe that both have 1 occurrence of free variable: s^B.
  (so t can be used as argument of non-linear application)
  then:

$$insert \quad = \quad \lambda l^{L(A)}\ a_0^{\S A}.$$

$$\Lambda\alpha.\lambda s^B.((l\ t\ u)\ a_0) \ : \ L(A) \multimap \S A \multimap L(A)$$

Example 2: insertion sort (continued)

- **sorting**:

  $insert : L(A) \multimap \S A \multimap L(A)$ has been defined.

  $insert' : \S A \multimap L(A) \multimap L(A)$

  then

$$sort = \lambda l^{L(\S A)}.(l\ insert'\ nil)\ \ : L(\S A) \multimap \S L(A).$$

Relationship with LAL

- Clearly, $DLAL \subseteq LAL$. Do we lose anything?

Relationship with LAL

- Clearly, $DLAL \subseteq LAL$. Do we lose anything?
- Not at all. LAL is encodable in DLAL by:

$$(!A)^\bullet = \forall\alpha.(A^\bullet \Rightarrow \alpha) \multimap \alpha$$

As a consequence,

Relationship with LAL

- Clearly, $DLAL \subseteq LAL$. Do we lose anything?
- Not at all. LAL is encodable in DLAL by:

$$(!A)^\bullet = \forall\alpha.(A^\bullet \Rightarrow \alpha) \multimap \alpha$$

As a consequence,

- Theorem (Extensional Ptime Completeness): If a function $f : \{0,1\}^\star \to \{0,1\}^\star$ is computable in polynomial time, then there exists a system F term M and an integer d such that $\vdash_{DLAL} M : W \multimap \S^d W$ and M computes f.

Ptime Strong Normalization

- Theorem (Ptime strong normalization): Let M be a system F term which has a typing derivation $\mathcal{D}$ of depth d in DLAL. Then M normalizes in:
  - at most $|M|^{2^d}$ β-reduction steps
  - and in time $O(|M|^{2^{d+2}})$ on a Turing machine.

  This result holds for any reduction strategy.

Ptime Strong Normalization

- Theorem (Ptime strong normalization): Let M be a system F term which has a typing derivation $\mathcal{D}$ of depth d in DLAL. Then M normalizes in:
  - at most $|M|^{2^d}$ β-reduction steps
  - and in time $O(|M|^{2^{d+2}})$ on a Turing machine.

  This result holds for any reduction strategy.
- Entails that every program of type $W \multimap \S^d W$ works in Ptime.

Main Result

- $DLAL$ typing problem: Given a closed term M^T of system F, determine if there exists a decoration A of T such that $\vdash_{DLAL} M : A$.

Main Result

- $DLAL$ typing problem: Given a closed term M^T of system F, determine if there exists a decoration A of T such that $\vdash_{DLAL} M : A$.
- Theorem: It can be solved in polynomial time.

Main Result

- $DLAL$ typing problem: Given a closed term M^T of system F, determine if there exists a decoration A of T such that $\vdash_{DLAL} M : A$.
- Theorem: It can be solved in polynomial time.
  - Negative aspect: big gap between Ptime and Σ_2. DLAL captures only those programs which are very easily seen to be Ptime.

Main Result

- $DLAL$ typing problem: Given a closed term M^T of system F, determine if there exists a decoration A of T such that $\vdash_{DLAL} M : A$.
- Theorem: It can be solved in polynomial time.
  - Negative aspect: big gap between Ptime and Σ_2. DLAL captures only those programs which are very easily seen to be Ptime.
  - Positive aspect: Still useful as a first quick check.

Methodology for DLAL typing

- Typing in DLAL = Decorating system F terms with ! and § boxes

Methodology for DLAL typing

- Typing in DLAL = Decorating system F terms with ! and § boxes
- Naive tactics
  1. Put ! where sharing takes place.
  2. Place ! and § boxes
     (with opening doors § and closing doors $\overline{\S}$)
     whenever neccesary.
  3. Update types. Go to 2.

Methodology for DLAL typing

- Typing in DLAL = Decorating system F terms with ! and § boxes
- Naive tactics
  1. Put ! where sharing takes place.
  2. Place ! and § boxes
     (with opening doors § and closing doors $\overline{\S}$)
     whenever neccesary.
  3. Update types. Go to 2.
- 2 is very difficult. Infinitely many ways to place boxes.

Methodology for DLAL typing

- Typing in DLAL = Decorating system F terms with ! and § boxes
- Naive tactics
  1. Put ! where sharing takes place.
  2. Place ! and § boxes
     (with opening doors § and closing doors $\overline{\S}$)
     whenever neccesary.
  3. Update types. Go to 2.
- 2 is very difficult. Infinitely many ways to place boxes.
- Instead, we place only opening doors § and closing doors $\overline{\S}$.

Methodology for DLAL typing

- Typing in DLAL = Decorating system F terms with ! and § boxes
- Naive tactics
  1. Put ! where sharing takes place.
  2. Place ! and § boxes
     (with opening doors § and closing doors $\overline{\S}$)
     whenever neccesary.
  3. Update types. Go to 2.
- 2 is very difficult. Infinitely many ways to place boxes.
- Instead, we place only opening doors § and closing doors $\overline{\S}$.
- Fundamental observation: if on every path from λx to x the doors are well-bracketed (and ...), then boxes can be rebuilt around the doors.

Methodology for DLAL typing

- Main Lemmas:

  1. Term M is typable in DLAL $\Longleftrightarrow$ one can insert doors into M in such a way that it is locally well-typed and doors are well-bracketed (and ...)

Methodology for DLAL typing

- Main Lemmas:
  1. Term M is typable in DLAL $\Longleftrightarrow$ one can insert doors into M in such a way that it is locally well-typed and doors are well-bracketed (and ...)
  2. Local well-typedness and well-bracketedness can be expressed by a set of boolean and integer constraints.

Methodology for DLAL typing

- Main Lemmas:
  1. Term M is typable in DLAL $\Longleftrightarrow$ one can insert doors into M in such a way that it is locally well-typed and doors are well-bracketed (and ...)
  2. Local well-typedness and well-bracketedness can be expressed by a set of boolean and integer constraints.
- For 2, replace sequences $\S \cdots \S$ of actual doors with formal integer parameters $\mathbf{n}$.
  - $\mathbf{n} > 0$ stands for $\S \ldots \S$ (n times)
  - $\mathbf{n} < 0$ stands for $\overline{\S} \ldots \overline{\S}$ (n times)

Methodology for DLAL typing

- Main Lemmas:
  1. Term M is typable in DLAL $\Longleftrightarrow$ one can insert doors into M in such a way that it is locally well-typed and doors are well-bracketed (and ...)
  2. Local well-typedness and well-bracketedness can be expressed by a set of boolean and integer constraints.
- For 2, replace sequences $\S \cdots \S$ of actual doors with formal integer parameters $\mathbf{n}$.
  - $\mathbf{n} > 0$ stands for $\S \ldots \S$ (n times)
  - $\mathbf{n} < 0$ stands for $\overline{\S} \ldots \overline{\S}$ (n times)
- We also use formal boolean parameters to distinguish ! from $\S$.

Example

$$M = (\lambda g^{\alpha\to\alpha}.(g\ (g\ x^{\alpha})))\lambda y^{\alpha}.z^{\alpha} \quad : \alpha$$

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

Example

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

boxing conditions:

$$\begin{cases}
\mathbf{n_1} \geq 0 & \\
\mathbf{n_1}+\mathbf{n_2} \geq 0 & \\
\mathbf{n_1}+\mathbf{n_2}+\mathbf{n_3}+\mathbf{n_6}+\mathbf{n_7} = 0 & \text{bracketing } (x) \\
\mathbf{n_1}+\mathbf{n_8} \geq 0 & \\
\mathbf{n_1}+\mathbf{n_8}+\mathbf{n_9} = 0 & \text{bracketing } (z) \\
\mathbf{n_3} \geq 0 & \\
\mathbf{n_3}+\mathbf{n_4} = 0 & \lambda\text{-scope } (g^1) \\
\mathbf{n_3}+\mathbf{n_6} \geq 0 & \\
\mathbf{n_3}+\mathbf{n_6}+\mathbf{n_5} = 0 & \lambda\text{-scope } (g^2)
\end{cases}$$

Example

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

p-types:

$$g : \S^{\mathbf{b_1},\mathbf{m_1}}(\S^{\mathbf{b_2},\mathbf{m_2}}\alpha \multimap \S^{\mathbf{m_3}}\alpha) \qquad x : \S^{\mathbf{b_4},\mathbf{m_4}}\alpha$$

$$y : \S^{\mathbf{b_5},\mathbf{m_5}}\alpha \qquad z : \S^{\mathbf{b_6},\mathbf{m_6}}\alpha$$

Local typing conditions:

$$\begin{cases} \mathbf{n_9}z^{\alpha} & : \quad \mathbf{n_9} + \mathbf{m_6} \geq 0 \\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha}) & : \quad \mathbf{n_8} \geq 0 \\ [(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})] & : \quad \mathbf{n_2} = 0, \mathbf{m_1} = \mathbf{n_8}, \mathbf{m_2} = \mathbf{m_5}, \\ & \qquad \mathbf{m_3} = \mathbf{n_9} + \mathbf{m_6}, \mathbf{b_2} = \mathbf{b_5} \\ \ldots & \\ (g \text{ has 2 occurrences}) & \qquad \mathbf{b_1} = 1 \end{cases}$$

Example

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

p-types:

$$g : \S^{\mathbf{b_1},\mathbf{m_1}}(\S^{\mathbf{b_2},\mathbf{m_2}}\alpha \multimap \S^{\mathbf{m_3}}\alpha) \qquad x : \S^{\mathbf{b_4},\mathbf{m_4}}\alpha$$
$$y : \S^{\mathbf{b_5},\mathbf{m_5}}\alpha \qquad z : \S^{\mathbf{b_6},\mathbf{m_6}}\alpha$$

Bang conditions:

$$\begin{cases} \mathbf{b_2} = 1 \Rightarrow \mathbf{b_4} = 1 \\ \mathbf{b_2} = 0 \\ \mathbf{b_1} = 1 \Rightarrow \mathbf{b_6} = 1 \\ \mathbf{b_1} = 1 \Rightarrow \mathbf{n_8} \geq 1 \end{cases}$$

Example

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

p-types:

$$g : \S^{\mathbf{b_1},\mathbf{m_1}}(\S^{\mathbf{b_2},\mathbf{m_2}}\alpha \multimap \S^{\mathbf{m_3}}\alpha) \quad x : \S^{\mathbf{b_4},\mathbf{m_4}}\alpha$$
$$y : \S^{\mathbf{b_5},\mathbf{m_5}}\alpha \qquad\qquad z : \S^{\mathbf{b_6},\mathbf{m_6}}\alpha$$

Boolean constraints:

$$\begin{aligned}\mathsf{Const}^b(\overline{M}) \quad = \quad & \{\mathbf{b_1} = 1,\ (\mathbf{b_2} = 1 \Rightarrow \mathbf{b_4} = 1), \mathbf{b_1} = 0,\\ & (\mathbf{b_1} = 1 \Rightarrow \mathbf{b_6} = 1),\ \mathbf{b_2} = \mathbf{b_5},\ \mathbf{b_2} = \mathbf{b_4}\}\end{aligned}$$

Minimal solution ψ^b:

$$\mathbf{b_1} = \mathbf{b_6} = 1, \quad \mathbf{b_2} = \mathbf{b_4} = \mathbf{b_5} = 0.$$

Example

$$\overline{M} = \mathbf{n_1}[(\mathbf{n_2}\lambda g^{\alpha\to\alpha}.\mathbf{n_3}(\mathbf{n_4}g\ \mathbf{n_6}(\mathbf{n_5}g\ \mathbf{n_7}x^{\alpha})))\ \mathbf{n_8}(\lambda y^{\alpha}.\mathbf{n_9}z^{\alpha})]$$

p-types:

$$g : \S^{\mathbf{b_1},\mathbf{m_1}}(\S^{\mathbf{b_2},\mathbf{m_2}}\alpha \multimap \S^{\mathbf{m_3}}\alpha) \quad x : \S^{\mathbf{b_4},\mathbf{m_4}}\alpha$$
$$y : \S^{\mathbf{b_5},\mathbf{m_5}}\alpha \qquad\qquad z : \S^{\mathbf{b_6},\mathbf{m_6}}\alpha$$

The linear system has solutions. One of them gives:

$$g : !(\alpha \multimap \alpha) \quad x : \S\alpha$$
$$y : \alpha \qquad\qquad z : !\alpha$$

$$\overline{M} = (\lambda g.\S\ (\ \bar{\S}g\ (\bar{\S}g\ \bar{\S}x)))\ \S(\lambda y.\bar{\S}z) \quad : \ \S\alpha$$

Implementation

- Written in (functional) OCAML; uses an external LP solver (GLPSOL).
- Input: F typed lambda-term.
- Successive phases:
  1. parsing
  2. constraints generation
  3. boolean constraints resolution
  4. linear constraints printing, and passing to the solver.
- Current version downloadable from

  `http://www-lipn.univ-paris13.fr/~atassi/`

Conclusion

- DLAL is a variant of LLL suitable for λ-calculus typing.
- An efficient type decoration algorithm for DLAL.
- An implementation in CAML.
- W.r.t other ICC systems (like TRS): modest intensional expressivity (fewer algorithms), but efficient checking procedure.
- Related project: NO-CoST project (*New Tools for Complexity: Semantics and Types*): 2005-2008 (ANR). http://www-lipn.univ-paris13.fr/nocost/