Cut-elimination, β-reduction and quantitative properties of programs

Simona Ronchi Della Rocca
Dipartimento di Informatica - Università di Torino
joint work with Marco Gaboardi

April 14, 2008

- Computational properties of a logic L can be inherited by a programming language P through a transformation of L into a type assignment system for P, in the spirit of Curry-Howard isomorphism.
- The standard approach does not hold for the Light Logics, since the modality controlling the duplication produce a mismatch between the cut-elimination and the β-reduction, so loosing both the subject reduction and the complexity bound.
- In general such a mismatch is overcame by designing the type assignment system for P using the principles of L, arranged in **ad hoc** way.
- The consequences are that the properties of P are no-more inherited directly by L, but they need to be proved again.
- We will show another approach to the problem, taking the Soft Linear Logic (SLL) as case study.

- Computational properties of a logic L can be inherited by a programming language P through a transformation of L into a type assignment system for P, in the spirit of Curry-Howard isomorphism.
- The standard approach does not hold for the Light Logics, since the modality controlling the duplication produce a mismatch between the cut-elimination and the β-reduction, so loosing both the subject reduction and the complexity bound.
- In general such a mismatch is overcame by designing the type assignment system for P using the principles of L, arranged in **ad hoc** way.
- The consequences are that the properties of P are no-more inherited directly by L, but they need to be proved again.
- We will show another approach to the problem, taking the Soft Linear Logic (SLL) as case study.

Logics and Type Assignments

- Computational properties of a logic L can be inherited by a programming language P through a transformation of L into a type assignment system for P, in the spirit of Curry-Howard isomorphism.
- The standard approach does not hold for the Light Logics, since the modality controlling the duplication produce a mismatch between the cut-elimination and the β-reduction, so loosing both the subject reduction and the complexity bound.
- In general such a mismatch is overcame by designing the type assignment system for P using the principles of L, arranged in **ad hoc** way.
- The consequences are that the properties of P are no-more inherited directly by L, but they need to be proved again.
- We will show another approach to the problem, taking the Soft Linear Logic (SLL) as case study.

- Computational properties of a logic L can be inherited by a programming language P through a transformation of L into a type assignment system for P, in the spirit of Curry-Howard isomorphism.
- The standard approach does not hold for the Light Logics, since the modality controlling the duplication produce a mismatch between the cut-elimination and the β-reduction, so loosing both the subject reduction and the complexity bound.
- In general such a mismatch is overcame by designing the type assignment system for P using the principles of L, arranged in **ad hoc** way.
- The consequences are that the properties of P are no-more inherited directly by L, but they need to be proved again.
- We will show another approach to the problem, taking the Soft Linear Logic (SLL) as case study.

Logics and Type Assignments

- Computational properties of a logic L can be inherited by a programming language P through a transformation of L into a type assignment system for P, in the spirit of Curry-Howard isomorphism.
- The standard approach does not hold for the Light Logics, since the modality controlling the duplication produce a mismatch between the cut-elimination and the β-reduction, so loosing both the subject reduction and the complexity bound.
- In general such a mismatch is overcame by designing the type assignment system for P using the principles of L, arranged in **ad hoc** way.
- The consequences are that the properties of P are no-more inherited directly by L, but they need to be proved again.
- We will show another approach to the problem, taking the Soft Linear Logic (SLL) as case study.

Logics and Type Assignments

Curry-Howard

Logics $\Longleftrightarrow$ Type Assignment

Logics and Type Assignments

Logics	Curry-Howard $\Longleftrightarrow$	Type Assignment
$\Gamma \vdash \sigma$	proof decoration $\Longrightarrow$	$\Gamma^* \vdash M : \sigma$

Logics and Type Assignments

Curry-Howard

Logics $\Longleftrightarrow$ Type Assignment

proof decoration

$$\Gamma \vdash \sigma \quad \Longrightarrow \quad \Gamma^* \vdash M : \sigma$$

erasing terms

$$\Gamma \vdash \sigma \quad \Longleftarrow \quad \Gamma^* \vdash M : \sigma$$

Logics and Type Assignments

Curry-Howard

Logics $\iff$ Type Assignment

$$\Gamma \vdash \sigma \quad \overset{\text{proof decoration}}{\Longrightarrow} \quad \Gamma^* \vdash M : \sigma$$

$$\Gamma \vdash \sigma \quad \overset{\text{erasing terms}}{\Longleftarrow} \quad \Gamma^* \vdash M : \sigma$$

cut-elimination (normalization) $\iff$ reduction rules

A working example: the implicative fragment of LJ

$$\frac{\sigma \in \Gamma}{\Gamma \vdash \sigma}\ (A)$$

$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \rightarrow \tau}\ (\rightarrow I) \qquad \frac{\Gamma \vdash \sigma \rightarrow \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}\ (\rightarrow E)$$

A working example: the implicative fragment of LJ

$$\frac{\sigma \in \Gamma}{\Gamma \vdash \sigma}\ (A)$$

$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}\ (\to I) \qquad \frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}\ (\to E)$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ (A)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to I) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (\to E)$$

A working example: the implicative fragment of LJ

$$\frac{\sigma \in \Gamma}{\Gamma \vdash \sigma}\ (A)$$

$$\frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}\ (\to I) \qquad \frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}\ (\to E)$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ (A)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to I) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (\to E)$$

Proofs normalization in LJ implies termination for the typed terms.
LJ type assignment is the core of the programming language ML.

$$\frac{}{A \vdash A}\ (Id) \qquad \frac{\Gamma \vdash A \qquad \Delta, A \vdash B}{\Gamma, \Delta \vdash B}\ (cut)$$

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

$$\frac{\Gamma \vdash A \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash \forall\alpha.A}\ (\forall R) \qquad \frac{\Gamma, B[C/\alpha] \vdash A}{\Gamma, \forall\alpha.B \vdash A}\ (\forall L)$$

$$\frac{\Gamma, \overbrace{A \ldots, A}^{n \text{ times}} \vdash C}{\Gamma, !A \vdash C}\ (mpx) \qquad \frac{\Gamma \vdash A}{!\Gamma \vdash !A}\ (sp)$$

n is the rank of the rule (mpx).

PTIME Soundness

The cut elimination procedure applied on a proof Π of size n stops after a number of steps

$$\leq |\Pi| \times n^{2d}$$

where:
- $|\Pi|$ is the size of Π
- n is the maximum rank of a multiplexor in Π
- d is the maximum number of nested applications of rule (sp) in Π (depth of the proof).

PTIME Completeness

Every PTIME Turing Machine can be encoded by a SLL proof , in such a way that data are encoded by proofs with depth 0.

A standard decoration of SLL by λ-terms

$$\frac{}{x : A \vdash x : A}\ (Id) \qquad \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B \quad \Gamma \# \Delta}{\Gamma, \Delta \vdash N[M/x] : B}\ (cut)$$

$$\frac{\Gamma \vdash M : A \quad x : B, \Delta \vdash N : C \quad \Gamma \# \Delta \quad y \text{ fresh}}{\Gamma, y : A \multimap B, \Delta \vdash N[yM/x] : C}\ (\multimap L)$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \multimap B}\ (\multimap R)$$

$$\frac{\Gamma \vdash M : A}{!\Gamma \vdash M :!A}\ (sp) \qquad \frac{\Gamma, x_0 : A, ..., x_n : A \vdash M : B}{\Gamma, x :!A \vdash M[x/x_0, ..., x/x_n] : B}\ (mpx)$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall \alpha.A}\ (\forall R) \qquad \frac{\Gamma, x : A[B/\alpha] \vdash M : C}{\Gamma, x : \forall \alpha.A \vdash M : C}\ (\forall L)$$

Problems

The decorated system does not enjoy subject reduction.
Let $M \equiv y((\lambda z.sz)w)((\lambda z.sz)w)$ and
$A = B \multimap B \multimap D, E = D \multimap !B$.
Let Π be the derivation:

$$\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw : !B}{s : E, w : D \vdash (\lambda z.sz)w : !B}\,(cut) \quad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x : !B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)$$

M contains **two** identical redexes $(\lambda z.sz)w$. But every cut-elimination step would reduce both the redexes in the same time, so loosing the corresponding between cut-elimination and β-reduction.

The decorated system does not enjoy subject reduction.
Let $M \equiv y((\lambda z.sz)w)((\lambda z.sz)w)$ and
$A = B \multimap B \multimap D, E = D \multimap !B$.
Let Π be the derivation:

$$\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw :!B}{s : E, w : D \vdash (\lambda z.sz)w :!B}\,(cut) \quad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x :!B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)$$

M contains **two** identical redexes $(\lambda z.sz)w$. But every cut-elimination step would reduce both the redexes in the same time, so loosing the corresponding between cut-elimination and β-reduction.

Analyzing the problem

$$\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw :!B}{s : E, w : D \vdash (\lambda z.sz)w :!B}\,(cut) \quad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x :!B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)$$

The red subderivation has a modal conclusion, while a not modal context. So it cannot be duplicated.

There are **two** different subterms (syntactically equal) in the language, corresponding to the red subderivation. So β-reducing only one of them would not correspond to a correct logical proof.

Conclusion: a cut-elimination steps does not correspond to a β-reduction, so the polynomial bound on the logic is not inherited by the language.

Analyzing the problem

$$\dfrac{\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw :!B}{s : E, w : D \vdash (\lambda z.sz)w :!B}\,(cut) \qquad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x :!B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)}{}$$

The red subderivation has a modal conclusion, while a not modal context. So it cannot be duplicated.

There are **two** different subterms (syntactically equal) in the language, corresponding to the red subderivation. So β-reducing only one of them would not correspond to a correct logical proof.

Conclusion: a cut-elimination steps does not correspond to a β-reduction, so the polynomial bound on the logic is not inherited by the language.

Analyzing the problem

$$\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw :!B}{s : E, w : D \vdash (\lambda z.sz)w :!B}\,(cut) \quad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x :!B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)$$

The red subderivation has a modal conclusion, while a not modal context. So it cannot be duplicated.

There are **two** different subterms (syntactically equal) in the language, corresponding to the red subderivation. So β-reducing only one of them would not correspond to a correct logical proof.

Conclusion: a cut-elimination steps does not correspond to a β-reduction, so the polynomial bound on the logic is not inherited by the language.

Analyzing the problem

$$\dfrac{\dfrac{s : E \vdash \lambda z.sz : E \quad t : E, w : D \vdash tw :!B}{s : E, w : D \vdash (\lambda z.sz)w :!B}\,(cut) \quad \dfrac{y : A, r : B, l : B \vdash yrl : D}{y : A, x :!B \vdash yxx : D}\,(m)}{y : A, s : E, w : D \vdash y((\lambda z.sz)w)((\lambda z.sz)w) : D}\,(cut)$$

The red subderivation has a modal conclusion, while a not modal context. So it cannot be duplicated.

There are **two** different subterms (syntactically equal) in the language, corresponding to the red subderivation. So β-reducing only one of them would not correspond to a correct logical proof.

Conclusion: a cut-elimination steps does not correspond to a β-reduction, so the polynomial bound on the logic is not inherited by the language.

The (cut) rule can be split into three different rules, according to the shape of the formulae, of the contexts and of the derivation:

Linear cut

$$\frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B \quad A \text{ not modal}}{\Gamma, \Delta \vdash N[M/x] : B} \ (L\ cut)$$

It corresponds to a linear substitution. In case $N \equiv N[xQ]$ and $M \equiv \lambda x.P$, it generates a β-redex where the bound variable occurs exactly once, and the cut-elimination corresponds to a β-reduction.

The (cut) rule can be split into three different rules, according to the shape of the formulae, of the contexts and of the derivation:

Duplication cut

$$\frac{\Pi \triangleright !\Gamma \vdash M :!A \quad \Delta, x :!A \vdash N : B \quad \Pi \text{ duplicable (*)}}{!\Gamma, \Delta \vdash N[M/x] : B} \; (D\ cut)$$

It corresponds to a substitution where the proof Π is copied n times, if n is the degree of the multiplexor generating $x :!A$. In case $N \equiv N[xQ]$ and $M \equiv \lambda x.P$, it generates a β-redex (with n occurrences of the bound variable) and the cut-elimination corresponds to a β-reduction.

(*) duplicable denotes that in Π the ! has been introduced by a rule (sp).

The (cut) rule can be split into three different rules, according to the shape of the formulae, of the contexts and of the derivation:

Sharing cut

$$\frac{\Pi \triangleright \Gamma \vdash M :!A \quad \Delta, x :!A \vdash N : B \quad \Pi \text{ not duplicable}}{\Gamma, \Delta \vdash N[M/x] : B} \ (S\ cut)$$

It corresponds to a linear substitution. In case $N \equiv N[xQ]$ and $M \equiv \lambda x.P$, it generates a β-redex. But, x can occur in N more than once, so a single cut elimination can correspond to more than one β-reduction step.

cut elimination and β-reduction

- Let Π be a SLL derivation, and let Π^* its decoration by the λ-term M. If Π does not contain S-cuts, then the number of cut-elimination steps in the normalization of Π is $\leq$ of the number of β-reductions in the normalization of M. Since it is easy to prove that the size of M is less that the size of Π, then the polynomial bound for M follows.
- If Π contains S-cuts, the number of β-reductions in the normalization of M can be greater than the number of cut-elimination steps in the normalization of Π. So the typing does not induce any property on the complexity of M.

cut elimination and β-reduction

- Let Π be a SLL derivation, and let Π^* its decoration by the λ-term M. If Π does not contain S-cuts, then the number of cut-elimination steps in the normalization of Π is $\leq$ of the number of β-reductions in the normalization of M. Since it is easy to prove that the size of M is less that the size of Π, then the polynomial bound for M follows.
- If Π contains S-cuts, the number of β-reductions in the normalization of M can be greater than the number of cut-elimination steps in the normalization of Π. So the typing does not induce any property on the complexity of M.

We want to restrict SLL in such a way that:
- S-cuts are forbidden.
- The polynomial properties are preserved.

We want to restrict SLL in such a way that:
- S-cuts are forbidden.
- The polynomial properties are preserved.

Just erasing the rule $(S\text{-}cut)$ is not sufficient, since the cut elimination precedure could create new $(S\text{-}cut)$ rules. So we need to restrict both the rules and the formulae.

The formulae are restricted in the following way:

$A ::= \alpha \mid \sigma \multimap A \mid \forall\alpha.A$ (Linear Formulae)

$\sigma ::= A \mid !\sigma$ (Formulae)

The rules are restricted in the following way:

- axioms introduce linear formulae.
- weakening introduces linear formulae.
- the (cut) can be either an L-cut or a D-cut
- the other rules are arranged in such a way to preserve the correct syntax of formulae.

The formulae are restricted in the following way:

$A ::= \alpha \mid \sigma \multimap A \mid \forall\alpha.A$ (Linear Formulae)

$\sigma ::= A \mid !\sigma$ (Formulae)

The rules are restricted in the following way:

- axioms introduce linear formulae.
- weakening introduces linear formulae.
- the (cut) can be either an L-cut or a D-cut
- the other rules are arranged in such a way to preserve the correct syntax of formulae.

Essential Soft Linear Logic (ESLL)

$$\frac{}{A \vdash A}\ (Id) \qquad \frac{\Gamma \vdash \tau \quad A, \Delta \vdash \sigma}{\Gamma, \tau \multimap A, \Delta \vdash \sigma}\ (\multimap L)$$

$$\frac{\Gamma, \sigma \vdash A}{\Gamma \vdash \sigma \multimap A}\ (\multimap R) \qquad \frac{\Gamma \vdash^{!} \tau \quad \Delta, \tau \vdash \sigma}{\Gamma, \Delta \vdash \sigma}\ (cut)$$

$$\frac{\Gamma \vdash \sigma}{\Gamma, A \vdash \sigma}\ (w) \qquad \frac{\Gamma \vdash \sigma}{!\Gamma \vdash !\sigma}\ (sp) \qquad \frac{\Gamma, A[B/\alpha] \vdash \sigma}{\Gamma, \forall\alpha.A \vdash \sigma}\ (\forall L)$$

$$\frac{\Gamma, \overbrace{\tau, ..., \tau}^{n} \vdash \sigma}{\Gamma, !\tau \vdash: \sigma}\ (m) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \forall\alpha.A}\ (\forall R)$$

$\Gamma \vdash^{!} \tau$ means that, if τ is modal then all formulae in Γ are modal.
So the (cut) rule is either a L or a D cut.

Property
The set of ESLL proofs is a proper subset of the set of SLL proofs.

So from the polynomial soundness of SLL is follows as corollary:

ESLL **PTIME Soundness**

The cut elimination procedure applied on an ESLL-proof Π of size n stops after a number of steps

$$\leq |\Pi| \times n^{2d}$$

where:
- $|\Pi|$ is the size of Π
- n is the maximum rank of a multiplexor in Π
- d is the maximum number of nested applications of rule (sp) in Π (depth of the proof).

Property
The set of ESLL proofs is a proper subset of the set of SLL proofs.

So from the polynomial soundness of SLL is follows as corollary:

ESLL **PTIME Soundness**

The cut elimination procedure applied on an ESLL-proof Π of size n stops after a number of steps

$$\leq |\Pi| \times n^{2d}$$

where:
- $|\Pi|$ is the size of Π
- n is the maximum rank of a multiplexor in Π
- d is the maximum number of nested applications of rule (sp) in Π (depth of the proof).

PTIME Completeness

Every PTIME Turing Machine can be encoded by a ESLL proof , in such a way that data are encoded by proofs with depth 0.

The ESTA Type Assignment System

$$\frac{}{x : A \vdash x : A}\ (Id) \qquad \frac{\Gamma \vdash M : \tau \quad x : A, \Delta \vdash N : \sigma \quad \Gamma \# \Delta \quad y \text{ fresh}}{\Gamma, y : \tau \multimap A, \Delta \vdash N[yM/x] : \sigma}\ (\multimap L)$$

$$\frac{\Gamma, x : \sigma \vdash M : A}{\Gamma \vdash \lambda x.M : \sigma \multimap A}\ (\multimap R) \qquad \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : \sigma \quad \Gamma \# \Delta}{\Gamma, \Delta \vdash N[M/x] : \sigma}\ (cut)$$

$$\frac{\Gamma \vdash M : \sigma}{\Gamma, x : A \vdash M : \sigma}\ (w) \qquad \frac{\Gamma \vdash M : \sigma}{!\Gamma \vdash M :!\sigma}\ (sp) \qquad \frac{\Gamma, x : A[B/\alpha] \vdash M : \sigma}{\Gamma, x : \forall \alpha.A \vdash M : \sigma}\ (\forall L)$$

$$\frac{\Gamma, x_1 : \tau, ..., x_n : \tau \vdash M : \sigma}{\Gamma, x :!\tau \vdash M[x/x_1, ..., x/x_n] : \sigma}\ (m) \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall \alpha.A}\ (\forall R)$$

Properties of ESTA

Property Let M be such that $\Pi \triangleright \Gamma \vdash M : \sigma$, for some Π, Γ, σ.

- The size of M is less than the size of Π.
- The number of β-reductions necessary to normalize M is less or equal to the number of cut-elimination steps necessary to normalize Π.

Corollary [Polynomial soundness]
Let M be such that $\Pi \triangleright \Gamma \vdash M : \sigma$, for some Π, Γ, σ. Then M reduces to normal form in a number of β-reduction steps

$$\leq |M| \times n^{2d}$$

where:
- $|M|$ is the number of symbols of M
- n is the maximum rank of a multiplexor in Π,
- d is the depth of Π.

FPTIME Completeness

Let a function $\mathcal{F}$ be computed in *polynomial time* P, where $deg(P) = m$, and in *polynomial space* Q, where $deg(Q) = l$, by a Turing Machine $\mathcal{M}$. Then it is λ-definable by a term $\underline{M}$ typable in STA as $!^{max(l,m,1)+1}\mathbf{S} \vdash \underline{M} : \mathbf{S}_{2m+1}$.

A further step: ESLL in natural deduction style

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

A further step: ESLL in natural deduction style

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

A further step: ESLL in natural deduction style

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

A further step: ESLL in natural deduction style

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

A further step: ESLL in natural deduction style

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

- A logic in sequent calculus style can be decorated by λ-terms, but:
- The same λ-term decorates some proofs
- Terms are built through substitution
- It is not possible to curry out proofs by induction on the structure of terms
- In fact the Curry-howard isomorphism is stated for logics in natural deduction style.
- In order to preserve the complexity properties we need to design a transformation of ESLL in natural deduction style, in such a way that cut-free proofs are translated in normal proofs and every cut-elimination step is trasformed into a normalization step.

The translation from ESLL to NESLL: an example

Let Π^* be the naturale deduction version of Π.
The rule

$$\frac{\Pi_1 : \Gamma \vdash \sigma \quad \Pi_2 : A, \Delta \vdash \tau}{\Gamma, \Delta, \sigma \multimap A \vdash \tau} \ (\multimap L)$$

is translated by replacing the axiom $A \vdash A$ in Π_2^* by:

$$\frac{\dfrac{}{\sigma \multimap A \vdash \sigma \multimap A} \ (Ax) \quad \Pi_1^* : \Gamma \vdash \sigma}{\sigma \multimap A, \Gamma \vdash A} \ (\multimap E)$$

so obtaining a proof of $\Gamma, \Delta, \sigma \multimap A \vdash \tau$.

The translation from ESLL to NESLL: an example

The rule

$$\dfrac{\dfrac{\Pi_2 \rhd \Gamma_2, A \vdash \tau \quad \Pi_1 \rhd \Gamma_1, \vdash \sigma}{\Gamma_1, \Gamma_2, \sigma \multimap A \vdash \tau}\,(\multimap L) \quad \dfrac{\Pi_3 \rhd \Delta, \sigma \vdash A}{\Delta \vdash \sigma \multimap A}\,(\multimap R)}{\Gamma_1, \Gamma_2, \Delta \vdash \tau}\,(cut)$$

is traslated by replacing the axiom $A \vdash A$ in Π_2^* by:

$$\dfrac{\dfrac{\Pi_3^* \rhd \Delta, \sigma \vdash A}{\Delta \vdash \sigma \multimap A}\,(\multimap I) \quad \Pi_1^* \rhd \Gamma_1 \vdash \sigma}{\Gamma_1, \Delta \vdash \tau}\,(\multimap E)$$

so obtaining a proof of $\Gamma_1, \Gamma_2, \Delta \vdash \tau$.

The translation of a cut is a detour!

NESLL

$$\frac{}{A \vdash A}\ (Ax) \qquad \frac{\Gamma \vdash \sigma}{\Gamma, A \vdash \sigma}\ (w)$$

$$\frac{\Gamma, \sigma \vdash A}{\Gamma \vdash \sigma \multimap A}\ (\multimap I) \qquad \frac{\Gamma \vdash \sigma \multimap A \qquad \Delta \vdash A}{\Gamma, \Delta \vdash A}\ (\multimap E)$$

$$\frac{\Gamma, \overbrace{\sigma, \ldots, \sigma}^{n} \vdash A}{\Gamma, !\sigma \vdash A}\ (mpx) \qquad \frac{\Gamma \vdash \sigma}{!\Gamma \vdash !\sigma}\ (sp)$$

$$\frac{\Gamma \vdash A \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash \forall\alpha.A}\ (\forall I) \qquad \frac{\Gamma \vdash \forall\alpha.A}{\Gamma \vdash A[B/\alpha]}\ (\forall E)$$

$$\frac{}{x : A \vdash x : A}\ (Ax) \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma, x : A \vdash M : \sigma}\ (w)$$

$$\frac{\Gamma, x : \sigma \vdash M : A}{\Gamma \vdash \lambda x.M : \sigma \multimap A}\ (\multimap I) \qquad \frac{\Gamma \vdash M : \sigma \multimap A \qquad \Delta \vdash N : A \quad \Gamma \# \Delta}{\Gamma, \Delta \vdash MN : A}\ (\multimap E)$$

$$\frac{\Gamma, x_1 : \sigma, \ldots, x_n : \sigma \vdash M : A}{\Gamma, x :!\sigma \vdash M[x/x_1, ..., x/x_n] : A}\ (mpx) \qquad \frac{\Gamma \vdash \sigma}{!\Gamma \vdash !\sigma}\ (sp)$$

$$\frac{\Gamma \vdash A \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash M : \forall \alpha.A}\ (\forall I) \qquad \frac{\Gamma \vdash M : \forall \alpha.A}{\Gamma \vdash M : A[B/\alpha]}\ (\forall E)$$

NOTE. $\Gamma \# \Delta$ denotes that the two contexts have disjoint variables.

Property Let M be such that $\Pi \triangleright \Gamma \vdash M : \sigma$, for some Π, Γ, σ.

- The size of M is less than the size of Π.
- The number of β-reductions necessary to normalize M is less or equal to the number of cut-elimination steps necessary to normalize Π.

Corollary [Polynomial soundness]
Let M be such that $\Pi \triangleright \Gamma \vdash M : \sigma$, for some Π, Γ, σ. Then M reduces to normal form in a number of β-reduction steps

$$\leq |M| \times n^{2d}$$

where:
- $|M|$ is the number of symbols of M
- n is the maximum rank of a multiplexor in Π,
- d is the depth of Π.

Properties of STA

FPTIME Completeness

Let a function $\mathcal{F}$ be computed in *polynomial time* P, where $deg(P) = m$, and in *polynomial space* Q, where $deg(Q) = l$, by a Turing Machine $\mathcal{M}$. Then it is λ-definable by a term $\underline{M}$ typable in STA as $!^{max(l,m,1)+1}\mathbf{S} \vdash \underline{M} : \mathbf{S}_{2m+1}$.

Bibliography

- *Gaboardi M., Ronchi Della Rocca S., " A Soft type assignment system for λ-calculus", CSL '07.*
- *Gaboardi M., Marion J. Y., Ronchi Della Rocca S., " A logical account of $PSPACE$", POPL'08.*
- *Gaboardi M., Marion J. Y., Ronchi Della Rocca S., "Soft Linear Logic and Polynomial Complexity Classes", LSFA '08, ENTCS, to appear*
- *Baillot P., Mogbil V., "Soft lambda-calculus: a language for polynomial time computation", Proc. 7th International Conference on Foundations of Software Science and Computational Structures, 2004*
- *Baillot P., Terui K., "Light Types for polynomial time computation in λ-calculus", LICS 04.*