Soft Time and Soft Space

Soft Linear Logic and Polynomial-bound Complexity Classes

Simona Ronchi Della Rocca

Dipartimento di Informatica - Università di Torino

`ronchi@di.unito.it`

Introduction

- ICC: Implicit Computational Complexity
- The problem: to design programming languages with bounded computational complexity
- The proposed solution: a ML-like approach
  - λ-calculus as paradigmatic programming language
  - Types as semantic properties of terms
  - Type assignment for λ-calculus such that:
    - types garantee the correctness of terms, in particular their complexity bound
    - if the type inference is decidable, the desired properties can be checked statically at compilation time
  - The tecnical tool: the Light Logics (derived from the Linear Logic of Girard) where the cut-elimination procedure is bounded in time by the size of the proof, exploiting the isomorphism:

$$FORMULAE \text{ as } TYPES$$

Outline

- Soft Linear Logic (SLL)(Lafont, 1988)
- STA, a type assignment for λ-calculus derived from SLL
- Properties of STA:
  - Subject reduction
  - Correctness: a term typable in STA reduces to normal form in a number of steps polynomial in its size
  - Completeness : all polynomial functions can be programmed in STA
- $\mathrm{STA}_{\mathbf{B}}$, an extension of STA typing an extended λ-calculus
  - Subject reduction
  - Correctness : a term typable in $\mathrm{STA}_{\mathbf{B}}$ can be reduced to normal form using polynomial space in its size
  - Completeness : all polynomial space functions can be programmed in $\mathrm{STA}_{\mathbf{B}}$
- Future development

Intuitionistic Linear Logic ($\multimap, !, \forall$ fragment)

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

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

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\ (!R) \qquad \frac{\Gamma, B \vdash A}{\Gamma, !B \vdash A}\ (!L)$$

$$\frac{\Gamma \vdash A}{\Gamma, !B \vdash A}\ (W) \qquad \frac{\Gamma, !B, !B \vdash A}{\Gamma, !B \vdash A}\ (C)$$

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

An equivalent formulation of ILL

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

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

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

$$\frac{\Gamma, !!B \vdash A}{\Gamma, !B \vdash A}\ (digging)$$

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

NOTE. (W) is (mpx), with $n = 0$. (C) is (mpx)+$(digging)$.

From ILL to SLL

$$\text{SLL} = \text{ILL} - (digging)$$

which means that

$$!A \multimap !!A$$

does not hold anymore.

So the modality ! can effectively be used for counting the number of duplications of formulae performed in a proof.

Soft Linear Logic (SLL) ($\multimap$, !, $\forall$ fragment)

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

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

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

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

n is the rank of the rule (mpx).

Properties of SLL

The cut elimination procedure applied on a proof Π of size n takes a number of steps $\leq |\Pi| \times n^d$, 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).

So, considering:

$$\begin{array}{ccc} PROOFS & as & PROGRAM \\ CUT-ELIMINATION & as & COMPUTATION \end{array}$$

SLL is correct for polynomial time computations. Moreover, every polynomial time Turing Machine can be encoded by a SLL proof. Since data can be encoded by proofs with depth 0, SLL is also complete for polynomial time computations.

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.

$$x : A \multimap !B, y : A \vdash xy :!B$$

So $x : A \multimap !B, y : A \vdash (\lambda zw.wzz)(xy) :!B \multimap (!B \multimap !B \multimap A) \multimap A$, but
$x : A \multimap !B, y : A \not\vdash \lambda w.w(xy)(xy) :!B \multimap (!B \multimap !B \multimap A) \multimap A$

- The decorated system does not inherit the complexity properties of SLL : some terms can be typed, which reduce in exponential time in their size:

$$z :!A, y_1 :!A \multimap !A \multimap !A, ..., y_n :!A \multimap !A \multimap !A \vdash_L (\lambda x.y_1xx)(...((\lambda x.y_nxx)z))...) :!A$$

(Technical reason: a term with a modal type can be derived from a not modal context, so modality does not implies anymore that the term can be duplicated) .
Moreover:

- The sequent calculus presentation is not suitable for a programming language. : it does not allow proofs by induction on terms.

Solution

STA is a natural deduction style type assignment system inspired by SLL, but:

- Terms are built in a linear way, and (mpx) rule is used for controlling variable duplication.
  Technically this is realized by using as types a subset of the SLL formulae such that:
  - $\forall$ is not allowed on modal formulae
  - ! is not allowed on the right of $\multimap$
- weakening introduces not modal formulae

STA types are the following subset of SLL formulae:

$$
\begin{array}{lcll}
A & ::= & \alpha \mid \sigma \multimap A \mid \forall \alpha.A & \text{(linear types)} \\
\sigma & ::= & A \mid !\sigma &
\end{array}
$$

Rules of STA

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

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

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

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

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

Linearity Properties of STA

- $\Gamma \vdash M : \sigma$ and $x : A \in \Gamma$ imply x occurs at most once in M;
- $\Pi :!\Gamma \vdash M :!\sigma$ implies Π can be tranformed into a derivation Π':

$$\frac{\Gamma \vdash M : \sigma}{!\Gamma \vdash M :!\sigma} \ (sp)$$

So the modality ! is truly a witness of the possibility of duplication!

Properties of STA

Theorem 1 (Subject Reduction) $\Gamma \vdash M : \mu$ *and* $M \rightarrow_\beta M'$ *imply* $\Gamma \vdash M' : \mu$

Theorem 2 (Polynomial Time Soundness) *Let* M *be typable in* STA *and let* $\Pi : \Gamma \vdash M : \sigma$*, for some* Γ *and* σ*, and let* $d(\Pi)$ *be the maximal nesting of (*sp*) rule applications in* Π*. Then reduces to a normal form in a number of steps:*

$$\leq \mid M \mid^{d(\Pi)+1}$$

and this implies that it reduces in normal form on a Turing machine in time:

$$\leq \mid M \mid^{3 \times (d(\Pi)+1)}$$

This means that every typing for M gives an upper bound to its reduction time !

Toward the Polynomial Completeness

Definition 1 (λ-definability) *Let f be an n-ary total function from $I_1 \times ... \times I_n$ to O, and let elements in I_i and in O be encoded by λ-terms ($1 \leq i \leq n$). Let $\underline{d}$ be the term encoding the data d.*
f is λ-definable if, for some $\underline{f} \in \Lambda$: $\underline{f}\,\underline{i_1}...\underline{i_n} =_\beta \underline{f(i_1, ..., i_n)}$.

So we can code:

- iterators by Church numerals

$$\underline{n} = \lambda xy.\, \underbrace{x(...x(x}_{n}\, y))) : \forall\alpha.!^i(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$$

- natural numbers by strings of booleans

$$[b_0, b_1, \ldots, b_n] \stackrel{\text{def}}{=} \lambda cz.cb_0(\cdots(cb_n z)\cdots) : \forall\alpha.!^i(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)$$

where $\mathbf{B} \stackrel{\text{def}}{=} \forall\alpha.\alpha \multimap \alpha \multimap \alpha$

Polynomial Completeness

Theorem 3 (PTIME Completeness) *If a decision problem* $\mathfrak{P}$ *is decided 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 representable by a term* $\underline{M}$ *typable in STA with a derivation* Π *with conclusion*

$$s :!^{max(l,m,1)+1}\forall\alpha.(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \vdash \underline{M} : \mathbf{B}$$

Theorem 4 (FPTIME Completeness) *If a function* $\mathcal{F}$ *is 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 representable by a term* $\underline{M}$ *such that:*

$$s :!^{max(l,m,1)+1}\forall\alpha.(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \vdash \underline{M} : \forall\alpha.!^{2m+1}(\mathbf{B} \multimap \alpha \multimap \alpha) \multimap (\alpha \multimap \alpha)$$

From Polynomial Time to Polynomial Space

Polynomial Space Computations coincide with polynomial time alternating Turing Machine Computations (APTIME). In particular:

PSPACE = NPSPACE = APTIME

So we can start from STA, characterizing polynomial time computations, adding to it some features (both to types and to the λ-calculus) in order to catch PSPACE.

We need to represent a computation that repeatedly fork into subcomputations and whose result is obtained by a backward computation from all the subcomputations results.

Technically we need:

- an if constructor on the language
- a special type $\mathbf{B}$ for booleans

Terms and Types of $\mathrm{STA}_{\mathbf{B}}$

Terms of $\mathrm{STA}_{\mathbf{B}}$:

$$M ::= x \mid 0 \mid 1 \mid \lambda x.M \mid MM \mid \ \texttt{if}\ M\ \texttt{then}\ M\ \texttt{else}\ M$$

Reduction rules:

$$(\lambda x.M)N \to_\beta M[N/x]$$

$$\texttt{if}\ 0\ \texttt{then}\ M\ \texttt{else}\ N\ \to_\delta M \qquad \texttt{if}\ 1\ \texttt{then}\ M\ \texttt{else}\ N\ \to_\delta N$$

$\to^*_{\beta\delta}$ denotes the reflexive and transitive closure of $\to_{\beta\delta}$.

Types of $\mathrm{STA}_{\mathbf{B}}$:

$$A ::= \mathbf{B} \mid \alpha \mid \sigma \multimap A \mid \forall \alpha.A \quad \text{(Linear Types)}$$

$$\sigma ::= A \ \mid !\sigma$$

Rules of $\mathrm{STA_B}$

$$\frac{}{x : A \vdash x : A}\ (Ax) \qquad \frac{}{\vdash 0 : \mathbf{B}}\ (\mathbf{B}_0 I) \qquad \frac{}{\vdash 1 : \mathbf{B}}\ (\mathbf{B}_1 I) \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma, x : A \vdash M : \sigma}\ (w)$$

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

$$\frac{\Gamma, x_1 : \sigma, \ldots, x_n : \sigma \vdash M : \mu}{\Gamma, x :!\sigma \vdash M[x/x_1, \cdots, x/x_n] : \mu}\ (m) \qquad \frac{\Gamma \vdash M : \sigma}{!\Gamma \vdash M :!\sigma}\ (sp)$$

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

$$\frac{\Gamma \vdash M : \mathbf{B} \quad \Gamma \vdash N_0 : \sigma \quad \Gamma \vdash N_1 : \sigma}{\Gamma \vdash \ \mathtt{if}\ M\ \mathtt{then}\ N_0\ \mathtt{else}\ N_1\ : \sigma}\ (\mathbf{B}E)$$

Properties of $\mathrm{STA}_{\mathbf{B}}$

- **Theorem 5 (Subject Reduction)** *Let* $\Gamma \vdash M : \sigma$ *and* $M \to_{\beta\delta} N$*. Then* $\Gamma \vdash N : \sigma$*.*

Remark 1 *The new rule* $(\mathbf{B}E)$ *has an additive behaviour of contexts. As consequence,* $\mathrm{STA}_{\mathbf{B}}$ *is no more correct for polynomial time computations.*
In fact, let:

$$M_n = (\lambda yz.y^n z)(\lambda x.\ \mathtt{if}\ x\ \mathtt{then}\ x\ \mathtt{else}\ x\ )0$$

for all n:

$$\vdash M_n :!(\mathbf{B} \multimap \mathbf{B}) \multimap \mathbf{B} \multimap \mathbf{B}$$

but

$$M_n \to^*_{\beta\delta} 0$$

in a number of steps exponential in n!

Toward PSPACE characterization

Let $M_0 \rightarrow_{\beta\delta} M_1 \rightarrow_{\beta\delta} \ldots \rightarrow_{\beta\delta} M_n$, where M_n is a normal form. The space used by this reduction is the maximum size of M_i $(0 \leq i \leq n)$.

While for STA the complexity time properties hold for every reduction strategy (i.e., a term M typable in STA reduces to normal form in a polynomial number of steps, for every reduction strategy), the space characterization will hold only for the leftmost-outermost reduction strategy. In fact, let:

$$M = (\lambda yz.z)M_n = (\lambda yz.z)((\lambda yz.y^n z)(\lambda x.\ \mathtt{if}\ x\ \mathtt{then}\ x\ \mathtt{else}\ x\ )0) \rightarrow^*_{\beta\delta} \lambda z.z$$

Clearly the size of M is linear in n. Using the leftmost outermost reduction strategy, it takes space linear in M:

$$(\lambda yz.z)M_n \rightarrow_{\beta\delta} \lambda z.z$$

while, using the innermost strategy, it takes space exponential in n, since (posing $P = \lambda x.\ \mathtt{if}\ x\ \mathtt{then}\ x\ \mathtt{else}\ x\ )0$)

$$M \rightarrow^*_{\beta\delta} (\lambda yz.z)(P^n 0) \rightarrow^*_{\beta\delta} 0$$

A leftmost outermost reduction machine

The machine is a set of rules of the shape:

$$\mathcal{C}, \mathcal{A} \models N \Downarrow b$$

where:

- $\mathcal{A}$ is the store, and it allows to perform substitutions one occurrence at a time:

$$\mathcal{A} ::= \emptyset \mid \mathcal{A}@\{x := M\}$$

- $\mathcal{C}$ is a context remembering the computation path, and it allows to avoid backtracking:

$$\mathcal{C}[\circ] ::= \circ \mid (\ \mathtt{if}\ \mathcal{C}[\circ]\ \mathtt{then}\ L\ \mathtt{else}\ R\ )V_1 \cdots V_n$$

- N is program (a closed term of type $\mathbf{B}$)

The rules of the machine

$$\frac{}{\mathcal{C}, \mathcal{A} \models \mathtt{b} \Downarrow \mathtt{b}} \ (Ax)$$

$$\frac{\mathcal{C}, \mathcal{A} @ \{x' := N\} \models M[x'/x]V_1 \cdots V_m \Downarrow \mathtt{b}^*}{\mathcal{C}, \mathcal{A} \models (\lambda x.M)NV_1 \cdots V_m \Downarrow \mathtt{b}} \ (\beta)$$

$$\frac{\{x := N\} \in \mathcal{A} \quad \mathcal{C}, \mathcal{A} \models NV_1 \cdots V_m \Downarrow \mathtt{b}}{\mathcal{C}, \mathcal{A} \models xV_1 \cdots V_m \Downarrow \mathtt{b}} \ (h)$$

$$\frac{\mathcal{C}[(\ \mathtt{if}\ [\circ]\ \mathtt{then}\ N_0\ \mathtt{else}\ N_1\ )V_1 \cdots V_m], \mathcal{A} \models M \Downarrow \mathtt{0} \quad \mathcal{C}, \mathcal{A} \models N_0 V_1 \cdots V_m \Downarrow \mathtt{b}}{\mathcal{C}, \mathcal{A} \models (\ \mathtt{if}\ M\ \mathtt{then}\ N_0\ \mathtt{else}\ N_1\ )V_1 \cdots V_m \Downarrow \mathtt{b}} \ (\ \mathtt{if}\ \mathtt{0})$$

$$\frac{\mathcal{C}[(\ \mathtt{if}\ [\circ]\ \mathtt{then}\ N_0\ \mathtt{else}\ N_1\ )V_1 \cdots V_m], \mathcal{A} \models M \Downarrow \mathtt{1} \quad \mathcal{C}, \mathcal{A} \models N_1 V_1 \cdots V_m \Downarrow \mathtt{b}}{\mathcal{C}, \mathcal{A} \models (\ \mathtt{if}\ M\ \mathtt{then}\ N_0\ \mathtt{else}\ N_1\ )V_1 \cdots V_m \Downarrow \mathtt{b}} \ (\ \mathtt{if}\ \mathtt{1})$$

(*) x' is a fresh variable.

Properties of $\mathrm{STA_B}$

Let the abstract machine compute: $\mathcal{C}, \mathcal{A} \models M \Downarrow \mathtt{b}$. Then the space used by the machine during this computation is:

the maximal size of the store used during the computation

+

the maximal size of the context used during the computation

Theorem 6 (Polynomial Space Soundness) *Let M be a program (a closed term of type $\mathbf{B}$), and let Π be a derivation of $\vdash M : \mathbf{B}$, and let $d(\Pi)$ be the depth of Π (the maximal nesting of applications of (sp) rule in Π). Then M reduces to normal form using a space*

$$\leq 3 \times \mid M \mid^{3 \times d(\Pi)+4}$$

This means that every typing for M gives an upper bound to its reduction space !

Properties of $\mathrm{STA}_{\mathbf{B}}$

Lemma 1 *A decision problem $\mathcal{D} : \{0,1\}^* \to \{0,1\}$ decidable by an Alternating Turing Machine $\mathcal{M}$ in polynomial time and space is programmable in $\mathrm{STA}_{\mathbf{B}}$.*

The proof is given by a coding of Alternating Turing Machine, similar to the coding used for STA.

Theorem 7 (Polynomial Space Completeness) *Every decision problem $\mathcal{D} \in$ PSPACE is programmable in $\mathrm{STA}_{\mathbf{B}}$.*

Bibliography

- STA and $\mathrm{STA}_{\mathbf{B}}$ have been presented respectively in:

  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 $PSPACE$", submitted.

- Other characterization of polynomial computations though λ-calculus and type assignment system based on LAL (Light Affine Logic):

  Baillot P., Terui K., "Light Types for polynomial time computation in λ-calculus", LICS 04.

- A characterization of elementary computations though λ-calculus and type assignment system based on EAL (Elementary Affine Logic):

  Coppola P., Dal Lago U., Ronchi Della Rocca S.,"Elementary Affine Logic and and the call-by-value λ-calculus", TLCA 05.

- There are not other logical charactizations of PSPACE, beyond $\mathrm{STA}_{\mathbf{B}}$.

Future developments

- The STA type assignment system is undecidable. We are exploring decidable restrictions of STA, which preserve the complexity bounds.
  (with Marco Gaboardi and Luca Roversi)
- We would like to give a characterization by a type assignment system also for (F)NPTIME , the computations that can be curried out in polynomial time by a non deterministic Turing Machine. The idea is to extend the λ-calculus by a non determistic operator, and STA by a logical sum.
  (with Marco Gaboardi)