Structural Proof-Theory and Computational Complexity

One year of work for PROTOCOLLO

Luca Roversi

Dipartimento di Informatica – Università di Torino

Rough organization of this summary

- Overview of our goals and methodology
- What we wanted to do
- What we have done
- Where we'd like to go

Goals: overview

- Structural proof-theory for characterizing complexity classes under the analogy:
  - **derivations-as-programs**
  - **normalization-as-evaluation**
- Cut-elimination constrained:
  - only some sub-derivations can be duplicated
  - duplicated sub-derivations may interact with care

Bottom-up methodology

- sub-recursive sub-systems of **LL**:

ILL	Elementary	Polynomial
$!(A_1 \otimes \cdots \otimes A_n) \multimap !A$	OK	$!A \multimap !B$
$!A \otimes !B \multimap !(A \otimes B)$	OK	Forbidden
$!A \multimap !A \otimes !A$	OK	OK
$!A \multimap !!A$	Forbidden	Forbidden
$!A \multimap A$	$!A \multimap \S A$	$!A \multimap \S A$
$!A \multimap \mathbb{I}$	OK	OK

- *soundness* and *completeness*: **ELL** w.r.t. elementary time; **LLL, LAL, SLL** w.r.t. polynomial time

Detailing out goals

1. On the **bottom-up** approach:
   - **delineating the meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: “light” computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**

Detailing out goals

1. On the bottom-up approach:
   - **delineating the meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: “light” computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**

Delineating the meaning

Bottom-up characterizations of a **complexity class** $\mathcal{C}$:

- $\mathbf{S} \subset$**LL** characterizes $\mathcal{C}$ if, **fixed** any $f \in \mathcal{C}$:
  - f can be represented by $\hat{f} \in \mathbf{S}$
  - normalizing $\hat{f}(\hat{n})$ costs $O(|\hat{f}(\hat{n})|^{e(d(\hat{f}(\hat{n})))})$:
    - $d(\hat{f}(\hat{n}))$ is the **depth** of $\hat{f}(\hat{n})$
    - e is some function;
  - $|\hat{n}|$ is linear in the value of n
  - $|\hat{f}|$ and $d(\hat{f}(\hat{n}))$ are **constants** w.r.t. the value of n
- **ELL**, **EAL**: e is a tower of exponentials
- **LLL**, **LAL**, **SLL**: e is a constant

Delineating the meaning

Bottom-up characterizations strongly depend on the chosen representation

- **LAL**:
  - is **DTIME**$[n^k]$ sound and complete, using $\forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$ as arguments
  - is **DTIME**$[2^{2^n}]$ sound and complete with $\S^{\mathbf{4n}}!(\forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha))$ as **non constant depth** argument data-types [MairsonMoller-02]

- **MLL**:
  - has a linear cut-elimination
  - is **DTIME**$[n^k]$ sound and complete, representing boolean circuits [Mairson03],[MairsonTerui-03]

Delineating the meaning

By all that, it seems to mean ...

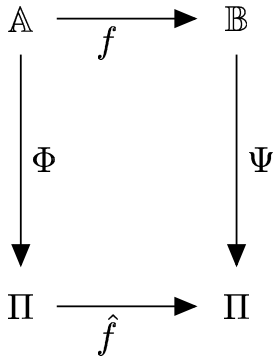
- sub-recursive sub-systems of **LL** can be very expressive and flexible:
  - they can supply argument data-types with "connected" dimension and depth
  - they can represent radically different computational models
- **bottom-up characterizations** of some $\mathcal{C}$ must be **up to structural constraints**

[DalLago-03]
Canonical representations of DTIME[n^k] in LAL

Delineating the meaning

[DalLago-03] Uniform representation in LAL

- Let $\mathcal{C}$ be **DTIME**$[n^k]$
- $\mathbb{A} \xrightarrow{f} \mathbb{B} \in \mathcal{C}$ is **uniformly representable** in **LAL** if



  - **commutes**
  - Π – set of arguments – has only cut-free derivations
  - $\Phi, \Phi^-, \Psi, \Psi^-$ are **SPACE**$[\ln]$ transducers

Delineating the meaning

[DalLago-03] Canonical representation in LAL

- $\mathbb{A} \xrightarrow{f} \mathbb{B} \in \mathcal{C}$ is **canonically representable** in **LAL** if
  - $\mathbb{A} \xrightarrow{f} \mathbb{B}$ is **uniformly representable**
  - Π – set of cut-free arguments – without $\forall$-left rules
  - $\forall$-left may hide depth and dimension interplays in argument data-types
- In **LAL**
  - **bottom-up characterizations** are canonical
  - $\exists f \in$ **DTIME**$[2^n]$ which is uniform, but not canonical
  - $\S^{\mathbf{4n}}!(\forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha))$ gives non uniform representation of **DTIME**$[2^{2^n}]$

Delineating the meaning

Research directions

- **natural** and **general** structural constraints, common to sub-recursive sub-systems of **LL**, to answer:

What is the meaning of bottom-up characterizations of complexity classes?

Detailing out goals

1. On the bottom-up approach:
   - **delineating the meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: “light” computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**:

Sub-recursive sub-systems of LL

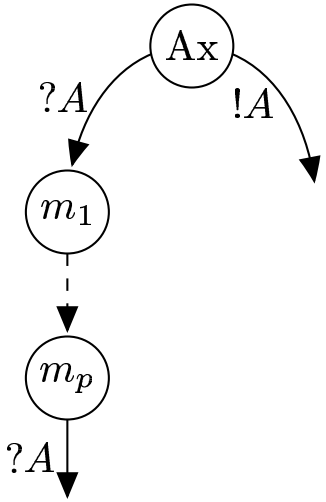
[Mazza-02]

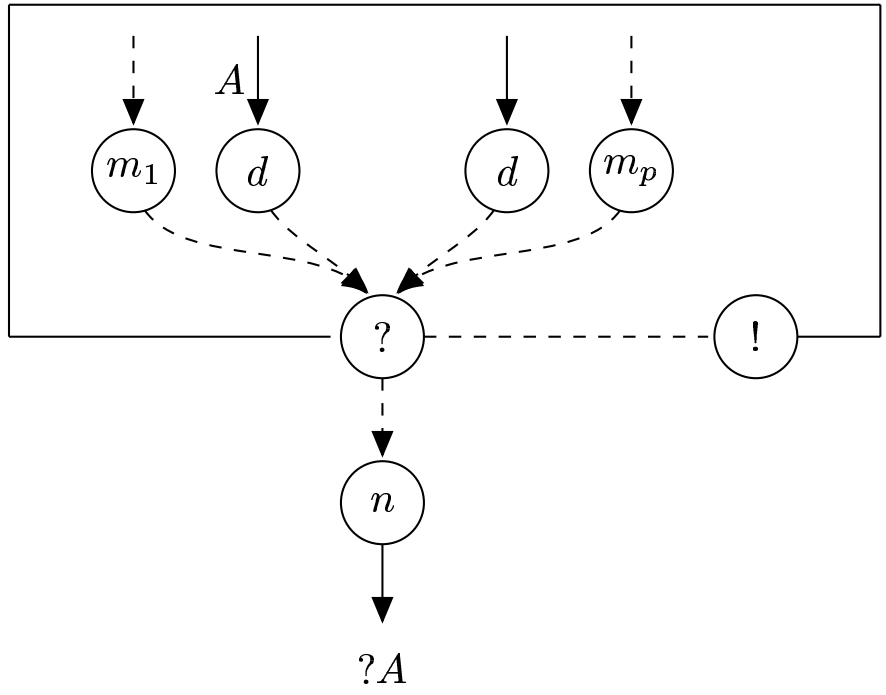
$$\overline{\textbf{LLL}} \;=\; \textbf{ELL}\text{-DJ} + (\S)\,\frac{\vdash ?\Gamma, \Delta}{\vdash ?\Gamma, \S\Delta} + \text{affine !-boxes} + \text{conditions on maximal exponential paths}$$

- $\overline{\textbf{LLL}}$ is sound and complete w.r.t. **DTIME**$[n^k]$

Sub-recursive sub-systems of LL

ELL-DJ = LL + global stratification conditions

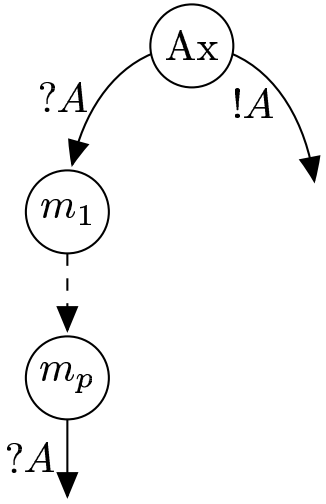


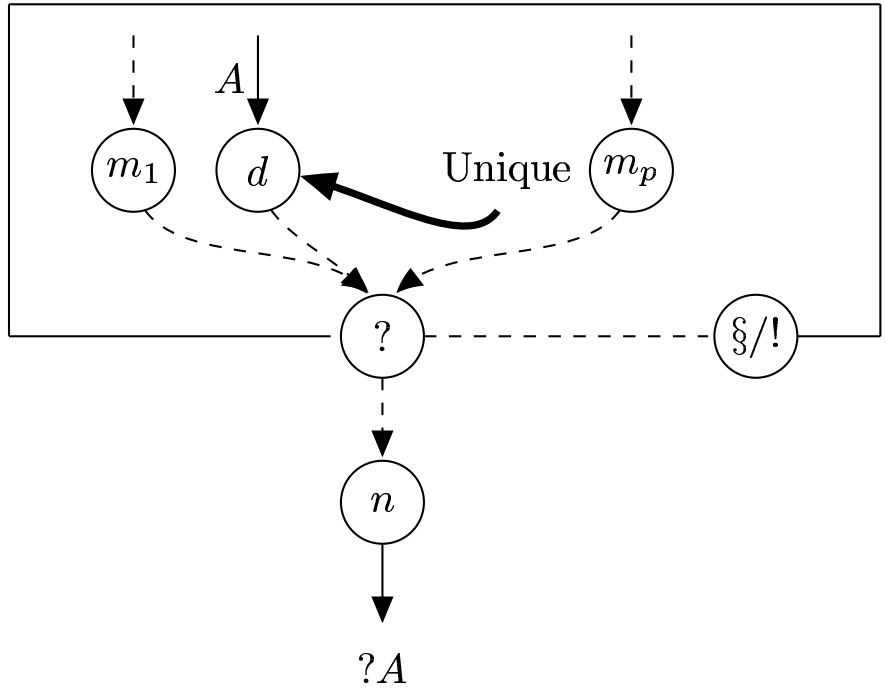


- $m_1, \ldots, m_p$ cannot be !-box premises
- one !-box crossed from **every** d to n

Sub-recursive sub-systems of LL

[Mazza-02] upward conditions of $\overline{\text{LLL}}$





- $m_1, \ldots, m_p$ cannot be !-box premises
- one !/§-box crossed from **every** d to m

Sub-recursive sub-systems of LL

[Mazza-03]

LLL as a sub-system LLL* of LL + $(\S)\ \dfrac{\vdash ?\Gamma, \Delta}{\vdash ?\Gamma, \S\Delta}$

- **LLL*** recovers the overlapping meaning of formulae inside additive blocks of **LLL**:

$$\dfrac{\dfrac{\dfrac{}{\vdash A^{\perp}; A}\,A}{\vdash A^{\perp}, B^{\perp}; A}\,W \qquad \dfrac{\dfrac{}{\vdash B^{\perp}; B}\,A}{\vdash A^{\perp}, B^{\perp}; B}\,W}{\vdash A^{\perp}, B^{\perp}; A\&B}\,\&$$

- **LLL*** uses weights on formulae and rules of **LL** + (§), to model "overlapping colors"

Sub-recursive sub-systems of LL

[Mazza-03]

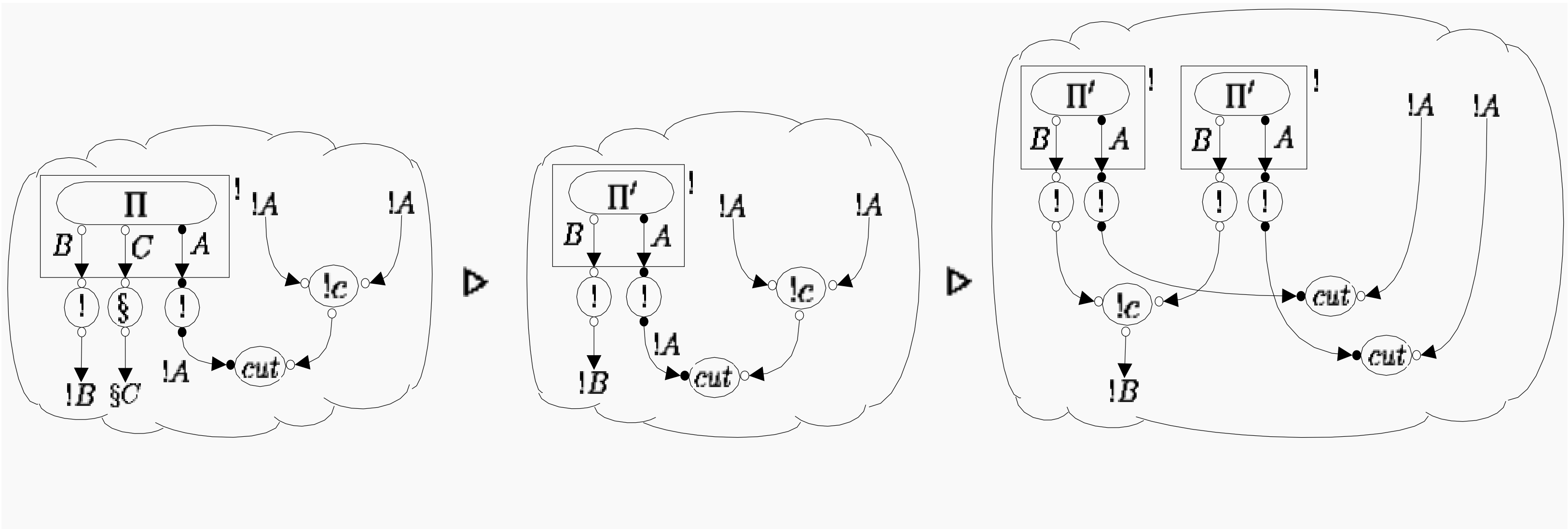
LLL as a sub-system LLL* of LL + $(\S)\ \dfrac{\vdash ?\Gamma, \Delta}{\vdash ?\Gamma, \S\Delta}$

- **LLL*** and **LLL** are equivalent w.r.t. provability;
- **LLL*** is **DTIME**$[n^k]$ complete and (**last minute news**) sound

Sub-recursive sub-systems of LL

LAL $\subset$ LALSup largest super-system for DTIME[n^k]?

- **LALSup** = **ELL** + §+ !-boxes become affine under cut-elimination



- Is **LALSup** more flexible than **LAL** w.r.t. programming?

Detailing out goals

1. On the bottom-up approach:
   - **delineating meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: “light” computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**

Looking for limits

[Pedicini-03]
Polynomial computations on rings in **LAL**

- [BlumShubSmile-end80] **BSS**:
  - hierarchy of complexity classes on rings ($\mathbb{R}, \mathbb{C}, \ldots$)
  - based on Turing machines

Detailing out goals

1. On the bottom-up approach:
   - **delineating meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: "light" computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**

Sub-recursive sub-systems

[DalLagoMartiniRoversi-03]

HOLRR $=$ linear affine λ-calculus $+$
higher-order ramified recursion

- sound and complete w.r.t. **DTIME**$[n^k]$
- typeable terms with $A ::= \mathbf{W} \mid A \otimes A \mid A \multimap A$
- recursion: **binder of higher-order variables**
- polynomial cost: depends only on the term structure
- normalization specified on terms
- **ramified recurrence** [Leivant-Marion] **embeds into HOLRR**

Sub-recursive sub-systems

HOLRR and Structural proof-theory?

- **HOLRR** originates from some speculations about **LALSup**
- it's relation with **LL** has to be made explicit:
  - does recursion use additively higher-order variables it binds?
  - do nested recursions inside **HOLRR** correspond to nested boxes of some "light logic" derivations?
  - are algebra constructors necessary to represent words in **HOLRR**? ($(\mathbb{I} \otimes \mathbf{Bool}) \simeq \mathbb{I}$+ δ-rule)

Detailing out goals

1. On the bottom-up approach:
   - **delineating meaning** of characterizing complexity classes by means of structural proof-theory
   - **identifying sub-recursive systems** as strict sub-systems of **LL**
   - **looking for limits**: “light” computations on interesting algebraic structures
2. **Sub-recursive sub-systems** à la **LL** vs. systems built under other principles
3. **Programming languages and light logics**

Programming languages and light logics

[CoppolaRonchi-03]
Principal typing schemata for Λ^{EA}

- $\Lambda^{EA} \subset \Lambda^{\rightarrow}$ typeable by **LJ**$^{\rightarrow}$
- Λ^{EA} contains **simple terms**: the corresponding proof-nets share, by contractions, only variables
- **Principal typing schemata**:
  - $M \in \Lambda^{EA}$ has finite set of principal types: all others are generated by substitution
  - types are formulae of **EAL**
- Lamping's optimal interpreter evaluates $M \in \Lambda^{EA}$ without overhead, typical on $\Lambda^{\rightarrow}$

Programming languages and light logics

Principal typing schemata for ...

- Λ^{EA} + recursion à la **HOLRR**?
- Λ^{EA} + constants?

Summing up our work ...

- P. Coppola, S. Ronchi della Rocca. *Principal Typing in Elementary Affine Logic*. In Proceedings of 6th International Conference, TLCA2003, LNCS 2701, 10-12 June 2003.
- Ugo Dal Lago. *On the expressive power of light affine logic*. In Proceeding of 8th Italian Conference on Theoretical Computer Science, ICTCS03, LNCS 2841, 13-15 October 2003.

... Summing up our work

- U. Dal Lago, S. Martini, L. Roversi. *Higer-order linear ramified recurrence*. Submitted 2003.
- D. Mazza. *Logica Lineare e Complessità Computazionale*. Tesi di laurea, Università degli Studi Roma Tre, 2002. **Submission under consideration**
- M. Pedicini. ***LAL** and polynomial computations on rings*. Work in progress, 2003
- Informal workshop ...

An informal workshop post-ICTCS03

- Participants:
  1. Paolo Coppola (Udine)
  2. Ugo Dal Lago (Bologna)
  3. Martin Hoffman (Munich)
  4. Harry Mairson (Boston - MA)
  5. Simone Martini (Bologna)
  6. Damiano Mazza (Marseille)
  7. Simona Ronchi Della Rocca (Torino)
  8. Luca Roversi (Torino)
  9. Katsushige Terui (Tokio)
  10. Lorenzo Tortora de Falco (Roma)

Thanks

Diagrams with Dal Lago-Puppi's MPost macros