

Safe Recursion on Notation into a Light Logic by Levels

Luca Roversi and Luca Vercelli

Università di Torino

March 27-28th, 2010, Cyprus

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - Objections

ICC and FPTIME

- **Implicit computational complexity** tries to capture complexity classes avoiding to mention Turing machines
- For us: avoid to explicitly mention its bound
- Here, **FPTIME** = the class of functions computable in polynomial time

FPTIME, SRN and ILAL

We shall consider two different traditional ways to characterize **FPTIME**:

- **SRN**, or **BC**, an algebra of functions.
All the **FPTIME** functions can be represented by elements in the algebra.
- **ILAL**, a light logic.
All the **FPTIME** functions can be represented as proofs, or proof nets, in the logic.

FPTIME, SRN and ILAL

We shall consider two different traditional ways to characterize **FPTIME**:

- **SRN**, or **BC**, an algebra of functions.
All the **FPTIME** functions can be represented by elements in the algebra.
- **ILAL**, a light logic.
All the **FPTIME** functions can be represented as proofs, or proof nets, in the logic.

FPTIME, SRN and ILAL

We shall consider two different traditional ways to characterize **FPTIME**:

- **SRN**, or **BC**, an algebra of functions.
All the **FPTIME** functions can be represented by elements in the algebra.
- **ILAL**, a light logic.
All the **FPTIME** functions can be represented as proofs, or proof nets, in the logic.

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - Objections

Safe Recursion on Notation

Base cases

$$s_i(; x) = 2x + i$$

$$i \in \{0, 1\}$$

$$\text{pred}(; 2x + i) = x$$

$$\pi_k^{n,s}(\vec{x}^n ; \vec{y}^s) = \begin{cases} x_k & \text{if } x \leq n \\ y_k & \text{if } x > n \end{cases}$$

$$\text{cond}(; a, b, c) = \begin{cases} b & \text{if } a \text{ odd} \\ c & \text{if } a \text{ even} \end{cases}$$

Safe Recursion on Notation

Inductive cases

$$f'(\vec{x}^n; \vec{y}^s) = f(g_1(\vec{x}^n; \cdot), \dots, g_{n'}(\vec{x}^n; \cdot); \\ h_1(\vec{x}^n; \vec{y}^s), \dots, h_{s'}(\vec{x}^n; \vec{y}^s))$$

$$\begin{cases} f(0, \vec{x}^n; \vec{y}^s) = g(\vec{x}^n; \vec{y}^s) \\ f(s_i(w), \vec{x}^n; \vec{y}^s) = h_i(w, \vec{x}^n; \vec{y}^s, f(w, \vec{x}^n; \vec{y}^s)) \end{cases} \quad i \in \{0, 1\}$$

Outline

- 1 The problem
 - SRN
 - **ILAL**
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - Objections

Intuitionistic Light Affine Logic (ILAL)

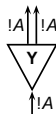
- A logic derived from Linear Logic
- Two modalities: $!$, \S
- Contraction restricted to $!$ -formulae:
- Free weakening:

Intuitionistic Light Affine Logic (ILAL)

- A logic derived from Linear Logic
- Two modalities: $!$, \S
- Contraction restricted to $!$ -formulae:
- Free weakening:

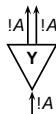
Intuitionistic Light Affine Logic (ILAL)

- A logic derived from Linear Logic
- Two modalities: $!$, \wp
- **Contraction** restricted to $!$ -formulae:
- Free weakening:



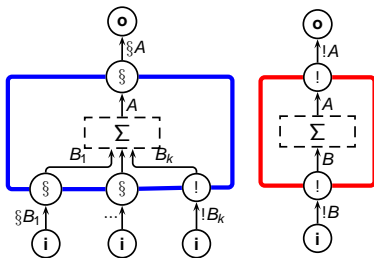
Intuitionistic Light Affine Logic (ILAL)

- A logic derived from Linear Logic
- Two modalities: $!$, ξ
- Contraction restricted to $!$ -formulae:
- Free weakening:



Intuitionistic Light Affine Logic (ILAL)

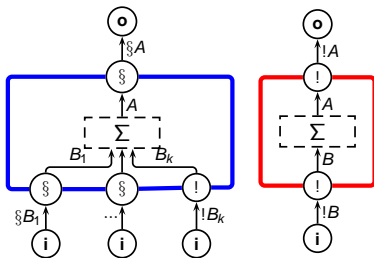
- Two promotion rules:



- No dereliction, no digging \rightarrow stratification

Intuitionistic Light Affine Logic (ILAL)

- Two promotion rules:



- No dereliction, no digging \rightarrow stratification

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - Objections

Similarities between **SRN** and **ILAL**

- In **SRN**, there are two distinct **sorts** of variables. An integer cannot be used to iterate itself, but only integers of “lower” sort.
- In **ILAL**, proofs are organized in **levels**. A proof (possibly representing an integer) cannot be used to iterate itself, but only proofs at “inner” levels.

sorts \leftrightarrow levels

From BC^- into ILAL

- BC^- is a fragment of **SRN**, in which safe arguments are never duplicated
- For example, given $g(; x, y)$, in BC^- we cannot define $g(; x, x)$
- It is possible to inject BC^- into **ILAL**:

Teorema (Murawski-Ong 2004)

*There exists a **compositional map** associating each term t of BC^- with a proof net $\lceil t \rceil$ of **ILAL**, such that $\lceil t \rceil$ behaves as expected.*

From BC^- into ILAL

- $\Rightarrow BC^-$ is the largest fragment of **SRN** that can be injected into **ILAL**, under reasonable assumptions.
- We shall get a better result, so we'll be obliged to reject some (but not all) of these reasonable assumptions.

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - Objections

Light Affine Logic by Levels (**LALL**)

LALL extends **ILAL** in two different directions:

- Recursive types
- Levels

Light Affine Logic by Levels (LALL)

First extension: recursive types

Recursive types: Formulæ are quotiented under

$$\mathcal{S} \equiv \forall \alpha. \alpha \multimap (\mathcal{S} \multimap \alpha) \multimap \alpha$$

We use a property of light logics that is seldom used in literature:

recursive types do not affect the complexity of **ILAL**

Why recursive types?

- \mathcal{S} is the type of **Scott numerals**.
- Simple successor, predecessor, conditional, **duplication**
- No simple sum

Why recursive types?

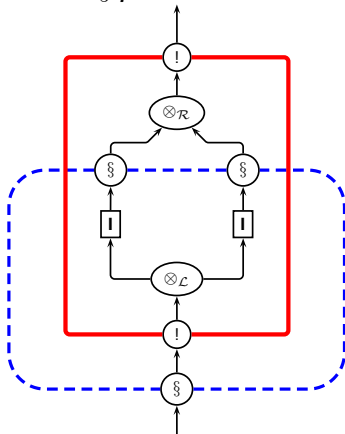
- \mathcal{S} is the type of **Scott numerals**.
- Simple successor, predecessor, conditional, **duplication**
- No simple sum

Well, to be precise, for every $M \in \mathbb{N}$ there exists a proof net $\nabla_{S_M} : \mathcal{S} \multimap \mathcal{S} \otimes \mathcal{S}$ that duplicates Scott numerals *smaller than M* .

Light Affine Logic by Levels (LALL)

Second extension: levels

Levels: the \S -promotion rules are more flexible than in **ILAL**

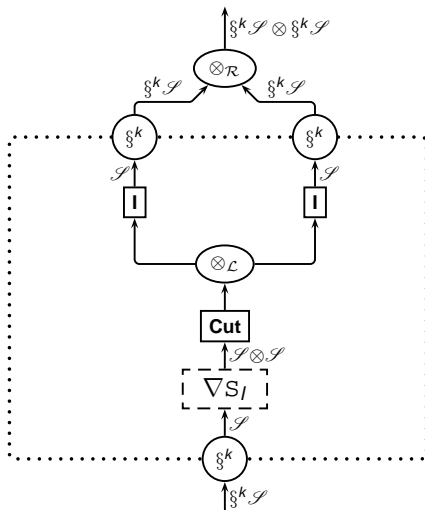


$$\frac{\frac{\overline{A \vdash A}}{A \vdash \S A} \text{id} \quad \S}{\frac{\overline{B \vdash B}}{B \vdash \S B} \text{id} \quad \S} \otimes R$$

$$\frac{A, B \vdash \S A \otimes \S B}{A \otimes B \vdash \S A \otimes \S B} \otimes L$$

$$\frac{\frac{!(A \otimes B) \vdash !(\S A \otimes \S B)}{\S!(A \otimes B) \vdash !(\S A \otimes \S B)} ! \quad \S}{\S!(A \otimes B) \vdash !(\S A \otimes \S B)} \S$$

Why levels?



A proof net duplicating a Scott integer exists with type

$$\S^k \mathcal{I} \multimap \S^k \mathcal{I} \otimes \S^k \mathcal{I}$$

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - **Main result**
 - Insertion sort
 - Objections

From SRN into ILAL

Theorem

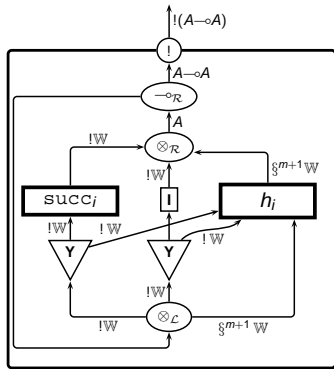
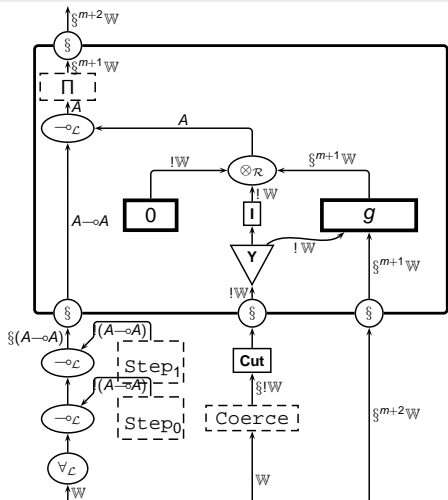
There exists a *compositional map*:

$$\begin{array}{ccc} \mathbf{SRN} \times \mathbb{N} & \rightarrow & \mathbf{LALL} \text{ proof nets} \\ (t, l) & \mapsto & \lceil t \rceil^l \end{array}$$

such that $\lceil t \rceil^l$ represents t for arguments smaller than l .

From SRN into ILAL

Remember this?



Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - **Insertion sort**
 - Objections

Just as an example, InsertionSort

$$\begin{aligned}
 \text{gt}^{\mathbb{C} \rightarrow \mathcal{I} \rightarrow \mathbb{B}} &= \lambda x^{\mathbb{C}} . \lambda y^{\mathcal{I}} . \pi_2 (x \{ \mathcal{I} \otimes \mathbb{B} \} (\text{step})(y \otimes \mathbb{F})) \\
 \text{step}^{\mathcal{I} \otimes \mathbb{B} \rightarrow \mathcal{I} \otimes \mathbb{B}} &= \lambda s^{\mathcal{I}} \otimes b^{\mathbb{B}} . b \{ \mathcal{I} \otimes \mathbb{B} \} (0 \otimes \mathbb{T}) \\
 &\quad \left(s(0 \otimes \mathbb{T})(\lambda x^{\mathbb{B}} . \lambda y^{\mathcal{I}} . y \otimes \mathbb{F}) \right)
 \end{aligned}$$

True iff $x > y$.

Just as an example, InsertionSort

$$\text{ord}_I^{\mathcal{I} \otimes \mathcal{I} \rightarrow \mathcal{I} \otimes \mathcal{I}} = \lambda x^{\mathcal{I}} \otimes y^{\mathcal{I}} . (\lambda x_1^{\mathcal{I}} \otimes x_2^{\mathcal{I}} . \lambda y_1^{\mathcal{I}} \otimes y_2^{\mathcal{I}} . \\ \text{B2B}(\text{gt}(S2W_I[x_1]) y_1)(y_2 \otimes x_2))(\nabla_I x)(\nabla_I y)$$

Orders a couple (x, y) of words **at most / bits long**

$S2W_I$ transforms a Scott word into a Church word

Just as an example, InsertionSort

$$\mathbb{B} \rightarrow \mathbb{B} = \lambda x^{\mathbb{B}}. x\{\mathbf{B}\}(\lambda \gamma. \lambda w^{\gamma} \otimes z^{\gamma}. w \otimes z) \\ (\lambda \gamma. \lambda w^{\gamma} \otimes z^{\gamma}. z \otimes w)$$

Transforms booleans from type

$$\mathbb{B} = \forall \gamma. \gamma \rightarrow \gamma \rightarrow \gamma$$

into

$$\mathbf{B} = \forall \gamma. (\gamma \otimes \gamma) \rightarrow (\gamma \otimes \gamma)$$

Just as an example, InsertionSort

$$\text{putTop}[c]_I^{\mathcal{I} \rightarrow \mathcal{I} \otimes \alpha \rightarrow \mathcal{I} \otimes \alpha} = \lambda a^{\mathcal{I}} . \lambda b^{\mathcal{I}} \otimes t^{\alpha} .$$

$$(\lambda i^{\mathcal{I}} \otimes \lambda j^{\mathcal{I}} . i \otimes c j t)(\text{ord}_I a \otimes b)$$

Puts an element a at the top of a list, or at the second position.
 Integers are **at most / bits long**.
 (Think α is a list)

Just as an example, InsertionSort

$$\begin{aligned} \text{sort}_l^{L(\mathcal{S}) \multimap \circ \S L(\mathcal{S})} &= \lambda t^{L(\mathcal{S})}. t\{L(\mathcal{S})\}(\text{insert}_l) \emptyset^{L(\mathcal{S})} \\ \text{insert}_l^{\mathcal{S} \multimap \circ L(\mathcal{S}) \multimap \circ L(\mathcal{S})} &= \lambda n^{\mathcal{S}}. \lambda t^{L(\mathcal{S})}. \lambda c^{!(\mathcal{S} \multimap \circ \alpha \multimap \circ \alpha)}. \lambda z^{\alpha}. \\ &\quad (\lambda x^{\mathcal{S}} \otimes y^{\alpha}. c x y)((t \text{putTop}[c]_l) \\ &\quad (\text{Coers}_l^1[n] \otimes z)) \end{aligned}$$

And finally, InsertionSort algorithm.

Outline

- 1 The problem
 - SRN
 - ILAL
 - Relationship between SRN and ILAL
- 2 Contribution
 - LALL
 - Main result
 - Insertion sort
 - **Objections**

Possible (and Frequent) Objections

- Q Encoding finite fragments is not so standard, and not so interesting
- A/1 Real-world machines only compute finite fragments of programs
- A/2 *Circuits* behaves exactly that way

Possible (and Frequent) Objections

- Q You could just encode **SRN** into circuits; it is well-known circuits encode into **ILAL**
- A This is exactly what we do *not* call a simple encoding from **SRN** to **ILAL**

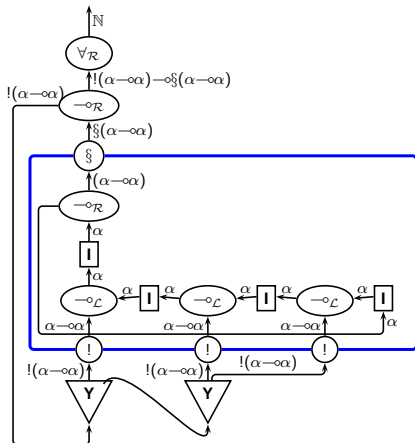
Summing up

- We find a polytime logic **LALL** extending **ILAL**
- We can embed in **LALL** all the **finite fragments** of **SRN** programs
- Two crucial features: **recursive types, levels**

Thank you.

Intuitionistic Light Affine Logic (ILAL)

Example: the number 3



$$\mathbb{N} = \forall \alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$$

ILAL

Example: the successor of 3...

