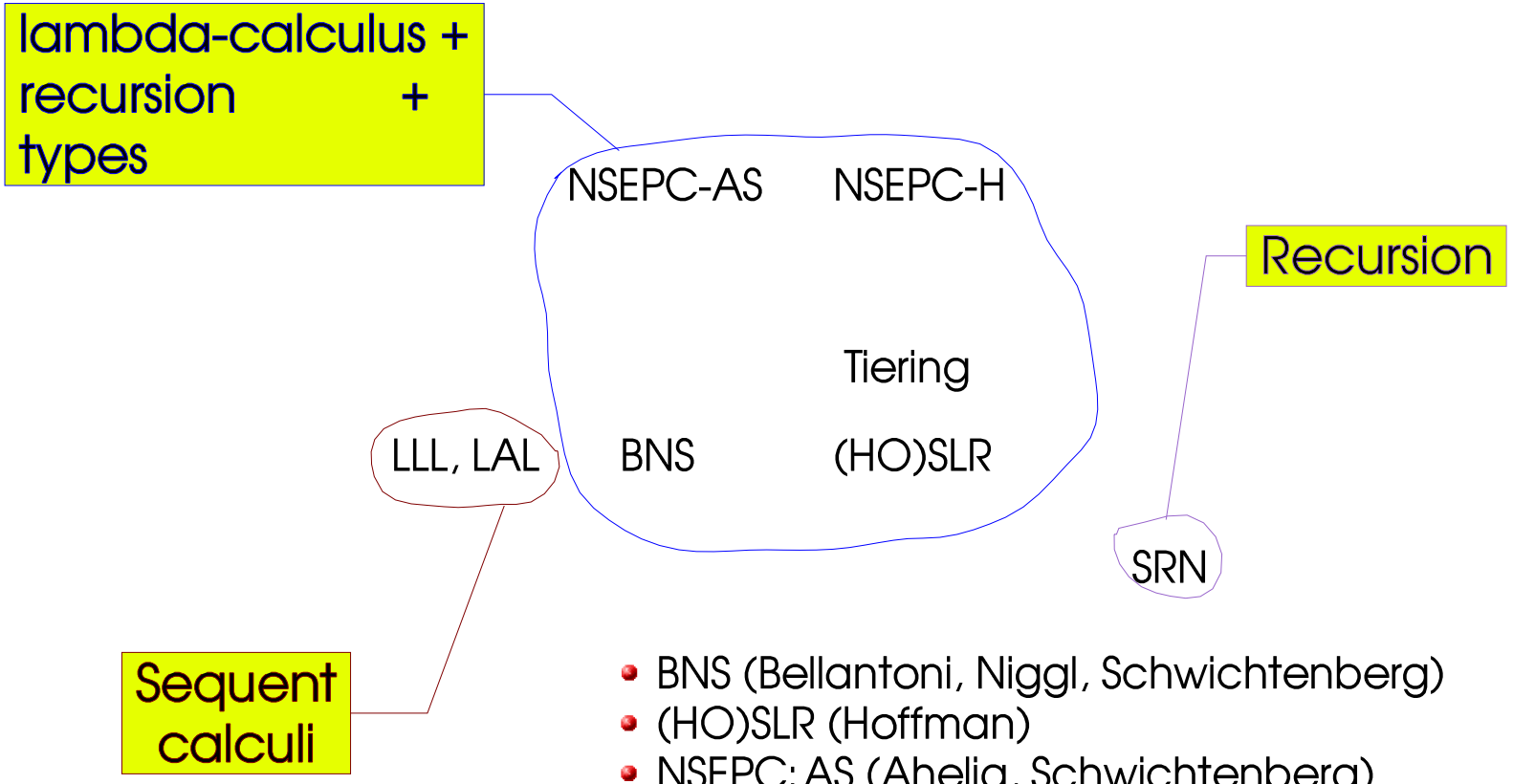
Light Linear Logic and Programming Languages

Luca Roversi

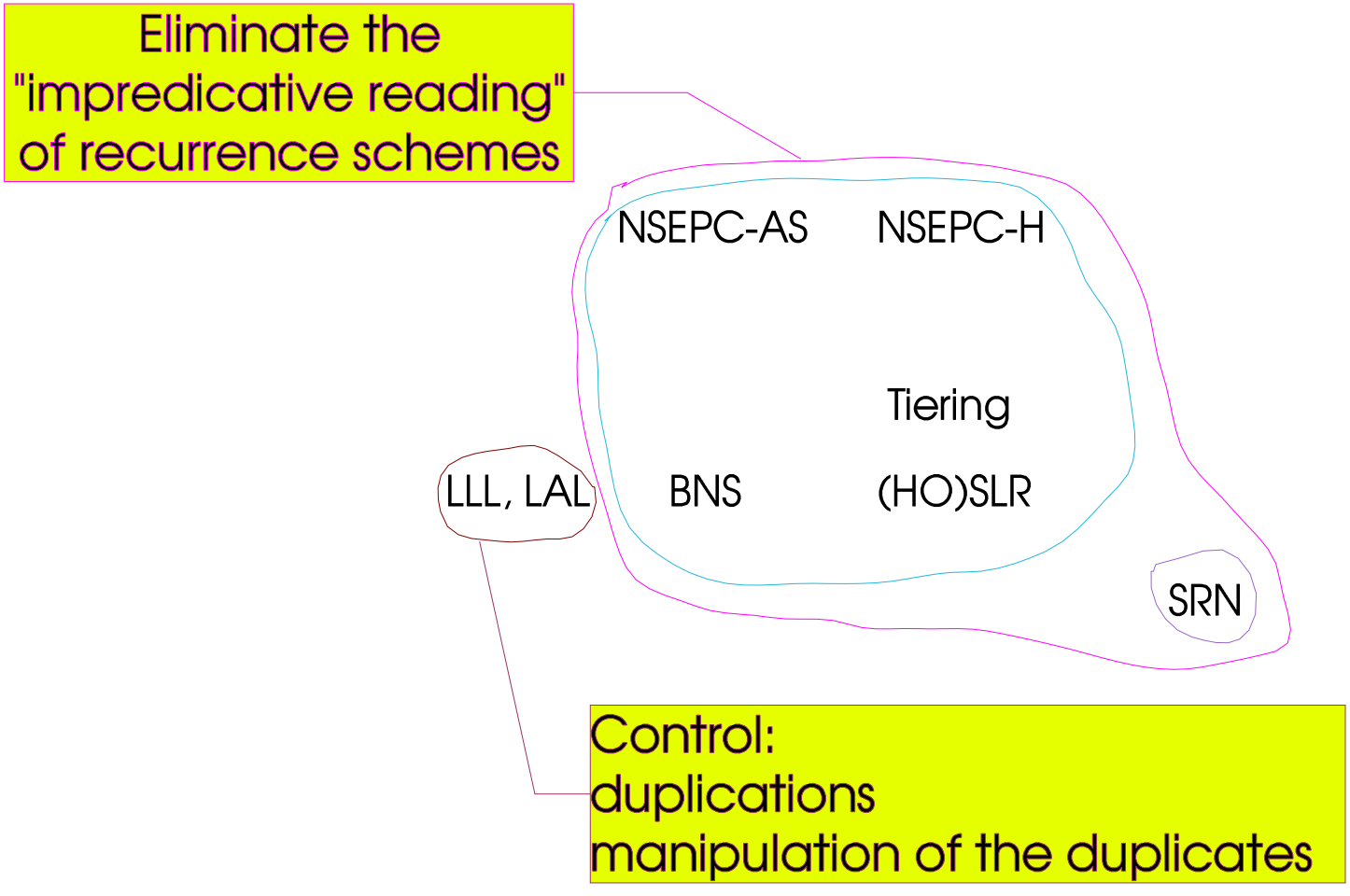
Dipartimento di Informatica – Università di Torino

“State of the art”

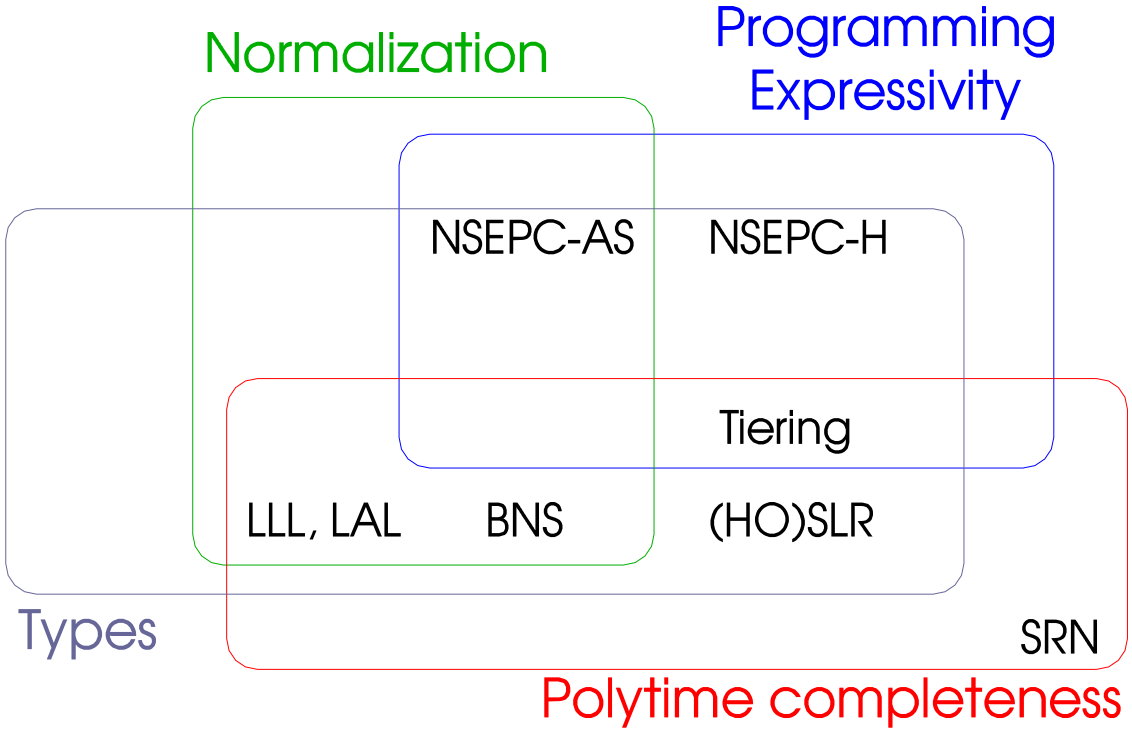


- BNS (Bellantoni, Niggl, Schwichtenberg)
- (HO)SLR (Hoffman)
- NSEPC: AS (Ahelig, Schwichtenberg)
- H (Hoffman)
- Tiering (Leivant, Marion)
- SRN (Bellantoni, Cook)

Comparing the methodologies



Comparing the programming languages



Goal

Intuitions about **LAL** as a programming language

- Sequent calculus + "λ-terms" + reduction
- Programming pattern:
  - organize a term by layers
  - "linearize" as much as possible every layer
  - sometimes replications are required: use their result inside a deeper layer
- Programming examples

Light Linear Logic (LLL) [Girard‘98]

- Characterizes **DTIME**$[n^k]$ under the paradigm “derivations-as-programs and normalization-as-evaluation”
- Limits two dynamical aspects:
  - the duplication of terms;
  - the freedom of arbitrary and independent manipulation of the duplicated terms

LLL and DTIME[n^k]

- **LLL** is *sound* with respect to **DTIME**[n^k]:
  - Any $\Pi \in$ **LLL** can be normalized, in polynomial time with respect to $|\Pi|$
- **LLL** is *complete* with respect to **DTIME**[n^k]:
  - There exists $\widehat{(\ )}$:**DTIME**[n^k]$\rightarrow$**LLL** such that, for every function $f \in$ **DTIME**[n^k], if $f(a) = b$, then $\widehat{f}(\widehat{a})$ normalizes to $\widehat{b}$

ILAL: “armless” rules ...

$$(ax)\ \frac{}{x : B \vdash x : B}$$

$$(cut)\ \frac{\Gamma \vdash M : A \quad \Delta, x : A \vdash N : B}{\Gamma, \Delta \vdash N\{^{M}/_{x}\} : B}$$

$$(\multimap_l)\ \frac{\Gamma \vdash M : A \quad \Delta, y : B \vdash N : C}{\Gamma, \Delta, x : A \multimap B \vdash N\{^{xM}/_{y}\} : C}$$

$$(\multimap_r)\ \frac{\Gamma, x_1 \otimes \ldots \otimes x_n : B_1 \otimes \ldots \otimes B_n \vdash M : B}{\Gamma \vdash \lambda x_1 \otimes \ldots \otimes x_n . M : B_1 \otimes \ldots \otimes B_n \multimap B}$$

- x in $(\multimap_l)$ is a new variable
- $\mathrm{Dom}(\Gamma) \cap \mathrm{Dom}(\Delta) = \emptyset$
- λ-terms have pattern matching

... “armless” rules

$$(\otimes_l)\ \frac{\Gamma, x_1 : B_1, x_2 : B_2 \vdash M : B}{\Gamma, x_1 \otimes x_2 : B_1 \otimes B_2 \vdash M : B}$$

$$(\otimes_r)\ \frac{\Gamma \vdash M : B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash M \otimes N : B \otimes A}$$

$$(\forall_l)\ \frac{\Gamma, x : \{^B/_\alpha\}A \vdash M : B}{\Gamma, x : \forall\alpha.A \vdash M : B}$$

$$(\forall_r)\ \frac{\Gamma \vdash M : A \quad \alpha \notin \mathtt{FV}(\Gamma)}{\Gamma \vdash M : \forall\alpha.A}$$

ILAL: “dangerous” rules

$$(w)\ \frac{\Gamma \vdash M : B}{\Gamma, x : A \vdash M : B}$$

$$(c)\ \frac{\Gamma, x :!A, y :!A \vdash M : B}{\Gamma, z :!A \vdash M\{{}^{z}/_{x}\ {}^{z}/_{y}\} : B}$$

$$(!)\ \frac{\ldots x_i : A_i \ldots \vdash M : B \qquad 0 \leq i \leq n \leq 1}{\ldots x_i :!A_i \ldots \vdash\ !M\{\ldots {}^{\overline{!}x_i}/_{x_i} \ldots\} :!B}$$

$$(\S)\ \frac{\ldots x_i : B_i \ldots x'_j : A_j \ldots \vdash M : B \qquad 0 \leq i \leq m \quad 0 \leq j \leq n}{\ldots x_i :!B_i \ldots x'_j : \S A_j \ldots \vdash \S M\{\ldots {}^{\overline{!}x_i}/_{x_i} \ldots {}^{\overline{\S}x'_j}/_{x'_j} \ldots\} : \S B}$$

Dynamics on the λ-terms

$$(\lambda x_1 \otimes \ldots \otimes x_m.M)(M_1 \otimes \ldots \otimes M_m) \quad \rhd_\beta \quad M\{^{M_1}/_{x_1} \ldots ^{M_m}/_{x_m}\}$$

$$\bar{?}!M \quad \rhd_! \quad M$$

$$\bar{\S}\S M \quad \rhd_\S \quad M$$

Integers

$$\begin{aligned}
\mathbf{Int} &= \forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)\\
\overline{0} &= \lambda x.\S(\lambda y.y) : \mathbf{Int}\\
\overline{n} &= \lambda x.\S(\lambda y.\underbrace{\bar{!}x(\ldots(\bar{!}x}_{n}\ y)\ldots)) : \mathbf{Int}
\end{aligned}$$

$$(\forall_r)\ \dfrac{(\multimap_r)\ \dfrac{(c)\ \dfrac{(\S)\ \dfrac{(\multimap_r)\ \dfrac{(\multimap_l)\ \dfrac{\alpha \vdash \alpha \quad \alpha, \alpha\cdots\alpha \multimap \alpha \vdash \alpha}{\alpha, \alpha \multimap \alpha\cdots\alpha \multimap \alpha \vdash \alpha}}{\alpha \multimap \alpha\cdots\alpha \multimap \alpha \vdash \alpha \multimap \alpha}}{!(\alpha \multimap \alpha)\cdots!(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{!(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{\vdash !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}}{\vdash \forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}$$

Successor on integers

$$\mathsf{succ} \quad = \quad \lambda z x.\S(\lambda y.\bar{!}x(\ \bar{\S}(z\ x)\ y)) : \mathbf{Int} \multimap \mathbf{Int}$$

$$\begin{aligned}
\mathsf{succ}\ \overline{n} \quad &\rightsquigarrow \quad \lambda x.\S(\lambda y.\bar{!}x(\ \bar{\S}(\overline{n}\ x)\ y)) \\
&\rightsquigarrow \quad \lambda x.\S(\lambda y.\bar{!}x(\ \bar{\S}\S(\lambda w.\overbrace{\bar{!}x(\ldots(\bar{!}x}^{n}\ w)\ldots))\ y)) \\
&\rightsquigarrow \quad \lambda x.\S(\lambda y.\bar{!}x((\lambda w.\bar{!}x(\ldots(\bar{!}x\ w)\ldots))\ y)) \\
&\rightsquigarrow \quad \lambda x.\S(\lambda y.\underbrace{\bar{!}x(\bar{!}x(\ldots(\bar{!}x}_{n+1}\ y)\ldots))) \\
&= \quad \overline{n+1}
\end{aligned}$$

Duplicating integers

$$\begin{array}{lcl}
\mathsf{dupl} & \equiv & \lambda n.\S(\,\overline{\S}(n\ !\mathsf{next})\ (\overline{0}\otimes\overline{0})) : \mathbf{Int} \multimap \S(\mathbf{Int}\otimes\mathbf{Int}) \\
\mathsf{next} & \equiv & \lambda n_l\otimes n_r.(\mathsf{succ}\ n_l)\otimes(\mathsf{succ}\ n_r) : \\
 & & (\mathbf{Int}\otimes\mathbf{Int}) \multimap (\mathbf{Int}\otimes\mathbf{Int})
\end{array}$$

$$\begin{array}{lcl}
\mathsf{dupl}\ \overline{n} & \rightsquigarrow & \S(\,\overline{\S}(\overline{n}\ !\mathsf{next})) \\
 & \rightsquigarrow^* & \S(\,\overline{\S}\S((\lambda y.\overbrace{\mathsf{next}(\mathsf{next}(\ldots(\mathsf{next}}^{n}\ y)\ldots)))(\overline{0}\otimes\overline{0})) \\
 & \rightsquigarrow^* & \S(\mathsf{next}(\mathsf{next}(\ldots(\mathsf{next}\ (\overline{0}\otimes\overline{0}))\ldots))) \\
 & \rightsquigarrow^* & \S(\mathsf{next}(\mathsf{next}(\ldots(\overline{1}\otimes\overline{1})\ldots))) \\
 & \rightsquigarrow^* & \S(\overline{n}\otimes\overline{n})
\end{array}$$

A first standard predecessor ...

(standard λ-calculus + pattern matching)

$$\begin{aligned} \mathsf{pred}_1 &\equiv \lambda n.\mathsf{fst}(n\ \mathsf{next}_1\ \mathsf{base}_1) \\ \mathsf{base}_1 &\equiv \overline{0} \otimes \overline{0} \\ \mathsf{next}_1 &\equiv \lambda l \otimes r.\, r \otimes (\mathsf{succ}\ r) \end{aligned}$$

- 1st replication layer: n replicates next_1
- 2nd replication layer: next_1 duplicates r
- 3rd replication layer: succ reconstructs its argument from scratch

... cannot be typed

$$n : \forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$$

$$\mathsf{dupl}\ \overline{n} \rightsquigarrow \S(\overline{n} \otimes \overline{n}) : \S(\mathbf{Int} \otimes \mathbf{Int})$$

$$\mathsf{next}_1 \equiv \lambda l \otimes r.\S((\lambda r_1 \otimes r_2.r_1 \otimes (\mathsf{succ}\ r_2))\ \bar{\S}(\mathsf{dupl}\ r)) : (\mathbf{Int} \otimes \mathbf{Int}) \multimap \S(\mathbf{Int} \otimes \mathbf{Int})$$

A second standard predecessor ...

(standard λ-calculus + pattern matching)

$$\begin{aligned} \mathsf{pred}_2 &\equiv \lambda nxy.\mathsf{fst}(n\ \mathsf{next}_2\ \mathsf{base}_2) \\ \mathsf{base}_2 &\equiv y \otimes y \\ \mathsf{next}_2 &\equiv (\lambda l \otimes r.r \otimes (x\ r)) \end{aligned}$$

... cannot be typed either

$$(\otimes_r)\ \frac{y : \mathbf{Int} \vdash y : \mathbf{Int} \quad y : \mathbf{Int} \vdash y : \mathbf{Int}}{y : \mathbf{Int}, y : \mathbf{Int} \vdash y \otimes y : \mathbf{Int} \otimes \mathbf{Int}}$$

is an incorrect derivation

Linear predecessor: intuition

$$\mathcal{T}_f(g,h) \quad = \quad (f, gh)$$

$$\overbrace{\mathcal{T}_f(\ldots\mathcal{T}_f}^{n}(g,h)\ldots) \quad = \quad (f, \overbrace{f(\ldots f}^{n-1}(gh)\ldots))$$

Linear predecessor: the term

$$T = \lambda f.\lambda g \otimes h.(f \otimes (gh)) : (\alpha \multimap \alpha) \multimap ((\alpha \multimap \alpha) \otimes \alpha) \multimap ((\alpha \multimap \alpha) \otimes \alpha)$$

$$\mathsf{base} = \lambda y.T\ I\ (I \otimes y) : \alpha \multimap ((\alpha \multimap \alpha) \otimes \alpha)$$

$$\mathsf{next} = \lambda x.T\ x : (\alpha \multimap \alpha) \multimap ((\alpha \multimap \alpha) \otimes \alpha) \multimap ((\alpha \multimap \alpha) \otimes \alpha)$$

$$\mathsf{pred} = \lambda n x.\S(\lambda y.\pi_2(\ \overline{\S}(n\ !(\mathsf{next}\ \overline{!}x))(\mathsf{base}\ y))) : \mathbf{Int} \multimap \mathbf{Int}$$

$$n\ !(\mathsf{next}\ \overline{!}x) : \S(((\alpha \multimap \alpha) \otimes \alpha) \multimap ((\alpha \multimap \alpha) \otimes \alpha))$$

Linear predecessor: iteration scheme

$$\dfrac{\dfrac{!\Gamma \vdash \mathcal{T}_f : !(A \multimap A) \qquad \Delta, y : \S(A \multimap A) \vdash M : B}{!\Gamma, \Delta, n : (!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha))\{^{A}/_{\alpha}\} \vdash M\{^{\bar{\S}(n\ \mathcal{T}_f)}/_{y}\} : B}}{!\Gamma, \Delta, \mathbf{Int} \vdash M\{^{\bar{\S}(n\ \mathcal{T}_f)}/_{y}\} : B}$$

What have we learned (hopefully)?

- “Programming by layers” is not quite friendly
- “Programming by layers” suggests new reasoning patterns to write programs