Semantic evaluation in Elementary Linear Logic

NGUYỄN Lê Thành Dũng (LIPN, Université Paris 13)
nltd@nguyentito.eu
joint work with Thomas SEILLER
Séminaire LDP, Marseille, 2018–09–13
Introduction
Two calculi of elementary complexity (1)

A “well-known” implicit characterization of ELEMENTARY: Elementary Linear Logic (ELL).

- *Exponential depth* controls complexity
- Soundness: normalization of terms of type !^k_{\text{bool}} is in $f(k)$-EXPTIME\(^1\) for some $f : \mathbb{N} \to \mathbb{N}$
- Completeness: any language in ELEMENTARY can be computed by an ELL function !^{\text{str}} \to !^k_{\text{bool}}

\(^1\)Tower of exponentials of height $f(k)$.
In fact there is another “calculus of elementary complexity” meeting these criteria:
Two calculi of elementary complexity (2)

In fact there is another “calculus of elementary complexity” meeting these criteria: the simply-typed λ-calculus (STλ).

- Functionality order controls complexity
  \( \text{ord}(\alpha \to \beta) = \max(\text{ord}(\alpha) + 1, \text{ord}(\beta)) \)

- Soundness: normalization in
  \( f(\max \text{ order in subterm}) \)-EXPTIME

- Completeness: non-obvious, discussed later
  - Corollary: normalization for STλ is non-elementary
    (Statman 1979)

Side remark: Linear Logic by Levels (L³) generalizes both

- ELL is a subset of L³ such that depth = level
- STλ embeds into L³, sending order to level
Church encodings of inputs in STλ

Church (or Böhm–Berarducci) encodings:

- For $\omega \in \{0, 1\}^*$, $\omega : \text{Str}[A]$ for any simple type $A$ (meta-$\forall$)
  - $\text{Str}[A] = (A \to A) \to (A \to A) \to (A \to A)$
  - $\overline{\omega} = \lambda f_0. \lambda f_1. \lambda x. f_{\omega[0]} (\ldots (f_{\omega[n-1]} x) \ldots)$

- $\text{Bool} = o \to o \to o$ ($o$ base type)

For $t : \text{Str}[A] \to \text{Bool}$, $\mathcal{L}(t) = \{\omega \in \{0, 1\}^* \mid t \overline{\omega} \rightarrow^*_\beta \text{true}\}$. Do we get extensional completeness for ELEMENTARY?
Church encodings of inputs in \( \text{ST}\lambda \)

Church (or Böhm–Berarducci) encodings:

- For \( w \in \{0, 1\}^* \), \( w : \text{Str}[A] \) for any simple type \( A \) (meta-\( \forall \))
  - \( \text{Str}[A] = (A \rightarrow A) \rightarrow (A \rightarrow A) \rightarrow (A \rightarrow A) \)
  - \( \overline{w} = \lambda f_0. \lambda f_1. \lambda x. f_w[0] (\ldots (f_w[n-1] x) \ldots) \)
- \( \text{Bool} = o \rightarrow o \rightarrow o \) (\( o \) base type)

For \( t : \text{Str}[A] \rightarrow \text{Bool} \), \( \mathcal{L}(t) = \{w \in \{0, 1\}^* \mid t \overline{w} \rightarrow_{\beta}^* \text{true}\} \).

Do we get extensional completeness for \text{ELEMENTARY}?

No, we get a much much smaller class!

**Theorem (Hillebrand & Kanellakis, LICS’96)**

The languages decided by \( \text{ST}\lambda\text{-terms of type} \text{Str}[A] \rightarrow \text{Bool} \) are exactly the regular languages.
Theorem (Hillebrand & Kanellakis, LICS’96)

For any type $A$ and any ST\(\lambda\)-term $t : \text{Str}[A] \rightarrow \text{Bool}$, the language $\mathcal{L}(t) = \{ w \in \{0, 1\}^* \mid t \overline{w} \rightarrow^*_\beta \text{true} \}$ is regular.

Proof: by constructing a deterministic finite automaton.

We use the semantics $[-] : \text{ST}\lambda \rightarrow \text{FinSet}$, and build a DFA with states $Q = [\text{Str}[A]]$, acceptation as $[t] (-) = [\text{true}]$.

$$
\begin{array}{c}
[\overline{\epsilon}] \\
\rightarrow \\
0 \\
\rightarrow \\
[0] \\
\rightarrow \\
1 \\
\rightarrow \\
[01] \\
\rightarrow \\
1 \\
\rightarrow \\
[011] \\
\rightarrow \\
\ldots
\end{array}
$$

$w$ accepted $\iff [t \overline{w}] = [t] ([\overline{w}]) = [\text{true}] \iff t \overline{w} \rightarrow^*_\beta \text{true}$

(when $[\text{true}] \neq [\text{false}]$, or equivalently $|\overline{0}| \geq 2$)
Semantic evaluation in the simply-typed $\lambda$-calculus

Theorem (Hillebrand & Kanellakis, LICS’96)

For any type $A$ and any $\text{ST}\lambda$-term $t : \text{Str}[A] \rightarrow \text{Bool}$, the language $L(t) = \{ w \in \{0, 1\}^* \mid t \overline{w} \rightarrow^*_\beta \text{true} \}$ is regular.

Proof: by constructing a deterministic finite automaton.

We use the semantics $[-] : \text{ST}\lambda \rightarrow \text{FinSet}$, and build a DFA with states $Q = [[\text{Str}[A]]]$, acceptation as $[t] (-) = [\text{true}]$.

($|Q| < \infty$, e.g. $2^{2^{33}}$ when $A = \text{Bool}$)

\[\begin{array}{c}
\overset{0}{\text{[0]}} \quad 1 \quad \overset{1}{\text{[01]}} \quad \overset{1}{\text{[011]}} \quad \ldots
\end{array}\]

$w$ accepted $\Leftrightarrow [t \overline{w}] = [t] ([\overline{w}]) = [\text{true}] \Leftrightarrow t \overline{w} \rightarrow^*_\beta \text{true}$

(when $[\text{true}] \neq [\text{false}]$, or equivalently $|[0]| \geq 2$)
Moral of the story

Finite denotational semantics have complexity consequences.

• Analogous results for tree automata, and for propositional linear logic (using your favorite finite model)
• Another application to ST\(\lambda\) at fixed order:

**Theorem (Terui, RTA’12)**

Normalizing an ST\(\lambda\)-term of type \(\text{Bool}\) w/ order \(\leq r\) subterms is

• \(k\)-EXPTIME-complete for \(r = 2k + 2\)
• \(k\)-EXPSPACE-complete for \(r = 2k + 3\)

**Proof of membership in \(k\)-EXPTIME / \(k\)-EXPSPACE.**

\(\beta\)-reduce to halve order, then evaluate in LL Scott model. \(\square\)
Regular languages in ELL
Application to second-order ELL at fixed depth

Now, $\text{Str} = \forall X. !(X \rightarrow X) \rightarrow !(X \rightarrow X) \rightarrow !(X \rightarrow X)$

**Theorem (Baillot, APLAS’11)**

The proofs of $\vdash \text{Str} \rightarrow \dual((1 \oplus 1))$ in 2\textsuperscript{nd} order elementary affine logic with recursive types decide exactly the languages in $\mathsf{P}$. 

<table>
<thead>
<tr>
<th>Theorem (Baillot, APLAS’11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The proofs of $\vdash \text{Str} \rightarrow \dual((1 \oplus 1))$ in 2\textsuperscript{nd} order elementary affine logic with recursive types decide exactly the languages in $\mathsf{P}$.</td>
</tr>
</tbody>
</table>
Application to second-order ELL at fixed depth

Now, \( \text{Str} = \forall X. ! (X \rightarrow X) \rightarrow !(X \rightarrow X) \rightarrow !(X \rightarrow X) \)

**Theorem (Baillot, APLAS’11)**

The proofs of \( ! \text{Str} \rightarrow !! (1 \oplus 1) \) in 2\textsuperscript{nd} order elementary affine logic with recursive types decide exactly the languages in \( \mathbb{P} \).

Recursive types are crucial for the above, as we show:

**Theorem**

The proofs of \( ! \text{Str} \rightarrow !! (1 \oplus 1) \) in 2\textsuperscript{nd} order ELL decide exactly the regular languages.

(First characterization of regular languages in a type system with impredicative quantification?)
For semantic evaluation in ELL, we need a non-trivial finite semantics for 2\textsuperscript{nd} order MALL.

**Theorem**

Such a semantics exists.

**Proof.**

Obtained by observational quotient; will be the topic of the last part of the talk. (Let’s assume it for now.)

New result of independent interest, with no mention of complexity.
Proof ideas (1)

- Let $\pi : !\text{Str} \to !!(1 \oplus 1)$; $\pi$ has the shape (modulo commutations)

\[
\begin{align*}
\vdash & \hat{\pi} \\
\vdash & \text{Str}[A_1] \perp, \ldots, \text{Str}[A_n] \perp, !(1 \oplus 1) \\
\vdash & \text{Str} \perp, \ldots, \text{Str} \perp, !(1 \oplus 1) \\
\vdash & ?\text{Str} \perp, \ldots, ?\text{Str} \perp, !!(1 \oplus 1) \\
\vdash & ?\text{Str} \perp, !!(1 \oplus 1) \\
\vdash & !\text{Str} \to !!(1 \oplus 1)
\end{align*}
\]

- For $w \in \{0, 1\}^*$, $\overline{w} : \text{Str}$, $\pi(\overline{w}) = \hat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$
- W.l.o.g. depth$(\pi) = 2$, thanks to stratification
  - “Erasing” all depth $> 2$ exponentials doesn’t change $\pi(\overline{w})$
- In other words w.l.o.g. $A_1, \ldots, A_n \in \text{MALL}$
Proof ideas (2)

• We want to know if $\pi(\overline{w}) = \hat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$ is !!true
• $\overline{w}[A] : !((A \to A) \to !((A \to A) \to !(A \to A)), A \in \text{MALL}$
• Using finite MALL semantics $[\_ \_ ]$, $\overline{w}$ induces map

$$\|w\|_A : \left[ A \to \right] \times \left[ A \to \right] \to \left[ A \to \right]$$

Lemma

$\hat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$ is determined by $(\|w\|_{A_1}, \ldots, \|w\|_{A_n})$.

• By Church encoding definition and $\left[ A \to \right] \simeq \text{End}(\left[ A \right])$,

$$\|w\|_A : (f_1, f_2) \in \text{End}(\left[ A \right])^2 \mapsto f_{w_1} \circ \ldots \circ f_{w_n}$$

• Our language is a preimage for a morphism to the finite monoid $\prod_{i=1}^{n} \text{End}(\left[ A_i \right])^{\text{End}(\left[ A_i \right])^2}$, therefore regular
What about higher depths?

Theorem (Baillot, APLAS’11)

The proofs of $!\text{Str} \rightarrow !^{k+2}(1 \oplus 1)$ in $2^{nd}$ order elementary affine logic with recursive types decide exactly the languages in $k$-EXPTIME.

- For ELL without recursive types, we get a class between $(k - 1)$-EXPTIME and $k$-EXPTIME... but which one exactly?
- Semantics probably has a role to play in the answer
Plan

Introduction

Regular languages in ELL

More implicit computational complexity

A finite denotational semantics for second-order MALL

Summary and conclusion
More implicit computational complexity
A bit of descriptive complexity

Data represented as (totally ordered) finite first-order structures (a.k.a. finite models), over a signature of relation symbols.

**Example**

Signature for binary strings: \( \langle \leq, S \rangle \).

Finite models are \((D, \leq^D, S^D), |D| < \infty. S^D(d) = \text{“}d^{\text{th}}\text{ bit is 1”}\).

*Descriptive complexity*: characterize a complexity class \( \mathcal{C} \) as set of queries written in some logic \( L_\mathcal{C} \), i.e. “is this \( L_\mathcal{C} \) formula true in this finite model?”. For instance:

**Theorem (Fagin 1974)**

*Queries in existential second-order logic = NP.*
Finite models in $\text{ST}\lambda$ and extensional completeness

With type $d$ of elements (equipped with $\text{Eq}: d \rightarrow d \rightarrow \text{Bool}$),

- Represent $k$-ary relations as lists of $k$-tuples
  \[ \text{Rel}_k[A] = (d^k \rightarrow A \rightarrow A) \rightarrow A \rightarrow A \]
- Provide a list of all elements in domain ($\text{List}[A] = \text{Rel}_1[A]$)

Theorem (Hillebrand, Kanellakis & Mairson, LICS’93)

Terms of type $\text{List}[A] \rightarrow \text{Rel}_{k_1}[A_1] \rightarrow \ldots \rightarrow \text{Rel}_{k_m}[A_m] \rightarrow \text{Bool}$ in $\text{ST}\lambda$ compute exactly ELEMENTARY queries over finite models.

To feed input, instantiate $d = o^n \rightarrow o$ ($n =$ domain size).
Program has “$\forall d$”, input has “$\exists d$”. Size of semantics depends on input, breaking earlier expressivity upper bound.

\[ ^2 \text{In the spirit of database theory: relation} = \text{set of records.} \]
We transpose this idea to second-order ELL:

- We use $\text{Rel}_k = D \otimes^k \rightarrow 1 \oplus 1$ and
  $\text{List} = \forall X. !(D \rightarrow X \rightarrow X) \rightarrow !(X \rightarrow X)$
- We allow non-linear use of $D$: $\text{cont} : D \rightarrow D \otimes D$, $\text{wk} : D^\perp$
- With stratification constraints, input type is thus

\[
\text{Inp} = \exists D. !\text{List} \otimes \bigotimes_{i=1}^n !\text{Rel}_{k_i} \otimes !!(D \rightarrow D \otimes D) \otimes !D^\perp
\]

- For size $n$ domain, witness $D = 1 \oplus \ldots (n \text{ times}) \ldots \oplus 1$
  (positive, therefore duplicable)
Towards logarithmic space in ELL?

**Theorem (Immerman 1983)**

Queries in first-order logic with deterministic transitive closure = logarithmic space (L) queries.

**Proposition**

All L queries on finite models for a given signature can be computed by an ELL proof of \( \text{Inp} \odot !!(1 \oplus 1) \).

Proof idea: compute transitive closure of a relation \( R \subseteq D^k \times D^k \) by iterating \( \varphi_R : \mathcal{P}(D^k \times D^k) \rightarrow \mathcal{P}(D^k \times D^k) \).

Determinism of \( R \) ensures linearity: \( \varphi_R : \text{Rel}_{2k} \rightarrow \text{Rel}_{2k} \) in ELL.

This is remarkable enough to hope for:

**Conjecture**

Conversely, proofs of \( \text{Inp} \odot !!(1 \oplus 1) \) only decide L queries.
A finite denotational semantics for second-order MALL
Theorem

Second-order MALL (MALL2) has a non-trivial finite semantics.

• Remark: in propositional MALL, each formula has finitely many cut-free proofs, i.e. the syntactic model is finite
• 2nd order case: arbitrarily large \(\exists\) witnesses
Handling the existentials

**Theorem**

*Second-order MALL (MALL2) has a non-trivial finite semantics.*

- Remark: in propositional MALL, each formula has finitely many cut-free proofs, i.e. the syntactic model is finite
- 2\textsuperscript{nd} order case: arbitrarily large $\exists$ witnesses
- Solution: *observational quotient* of the syntax
  - Choose observations which “cannot inspect witnesses”
  - Intuition from programming languages: $\exists = $ abstract types, dual to $\forall = $ generic programs
Equivalence for propositional observations

**Definition**

Let $A$ be a MALL2 formula and $\pi, \pi' : A$. Define $\pi \sim_A \pi'$ as:
for any propositional MALL formula $B$, for any proof $\rho$ of $A \vdash B$, $\text{cut}(\pi, \rho)$ and $\text{cut}(\pi', \rho)$ have the same normal form.

- $\sim$ is a congruence: the quotient is a model of MALL2
- $A$ existential-free $\Rightarrow \sim_A$ trivial
- Example: the proofs of $\exists X. X$ cannot be distinguished
  - Impredicative encodings of units work, e.g. $\top \equiv \exists X. X$.

**Theorem**

For any MALL2 formula $A$, there are finitely many classes for $\sim_A$.

Next: proof in the MLL case, using unit-free proof nets.
What a propositional MLL proof net looks like

[Diagram showing a proof net with nodes labeled X, Y⊥, Y, X⊥, and connectives ⊗, ⋈.

Atoms of A

Tree of connectives of A

(ax links)
What a **MLL2** proof net looks like

<table>
<thead>
<tr>
<th>X</th>
<th>Y⊥</th>
<th>Y</th>
<th>X⊥</th>
</tr>
</thead>
</table>

ax links + existential witnesses

Atoms of \( A \)

Tree of connectives of \( A \):

\( \otimes, \otimes, \forall, \exists \)
Finiteness theorem (1): proof/observation interaction

A: MLL2 formula; B: propositional MLL formula
Finiteness theorem (1): proof/observation interaction

A: MLL2 formula; B: propositional MLL formula
Finiteness theorem (2): eliminating two cuts
Finiteness theorem (2): eliminating two cuts
Finiteness theorem (2): eliminating two cuts
Finiteness theorem (2): eliminating two cuts
Finiteness theorem (2): eliminating two cuts
Finiteness theorem (3): big-step reduction

\[ \text{ax} \]

\[ \text{Remove } \gamma - \gamma' \quad \text{ax} \]

\[ \text{Perform cut-elim inside} \]
Finiteness theorem (4): normalization

New redexes may appear during reduction.
New redexes may appear during reduction.
Finiteness theorem (4): normalization

New redexes may appear during reduction.
New redexes may appear during reduction.

\[ \text{cut} \]
Finiteness theorem (4): normalization

New redexes may appear during reduction.

Lemma (Progress)

As long as there remains a cut-link, there is a redex for $\Rightarrow$. 
Finiteness theorem (5): proving the progress lemma

Lemma (Progress)
As long as there remains a cut-link, there is a redex for $\Rightarrow$.

Proof idea: make use of the “privacy”\(^3\) of $\exists$ witnesses.

Lemma
No cut-link can appear between two non-ax links introduced by witnesses in different proofs.

Corollary
In reducts for $\Rightarrow$, all cut-links have at least one ax premise.

Conclude with combinatorial reasoning.

\(^3\)As in the C++ private keyword.
Lemma (big-step normalization)

\[ \Rightarrow^* \text{ reaches the cut-free normal form in } \#(\text{atoms in } A)/2 \text{ steps.} \]

Theorem

\[ \sim_A \text{ has finitely many equivalence classes.} \]

Proof.

Let \( \pi : A \). We use \( \Rightarrow \) to compute the normal form of \( \text{cut}(\pi, \rho) \) for any observation \( \rho \). The map \( \rho \mapsto \text{NF}(\text{cut}(\pi, \rho)) \) is determined by a “strategy” for \( \pi \) of depth \( \#(\text{atoms in } A)/2 \). Therefore, for a fixed \( A \):

- all \( \pi \)'s with the same strategy are equivalent for \( \sim_A \)
- there are finitely many strategies
Summary and conclusion
ELL and ST\(\lambda\) both capture ELEMENTARY; by fixing a parameter (depth/order) one can characterize lower complexity classes.

We brought techniques from the ST\(\lambda\) tradition to 2\(^{nd}\) order ELL, showing that similar phenomena occur in both:

- **Church encodings** of inputs restrict expressivity
- **Semantic evaluation** can prove this (and lots of other stuff)
- To overcome this, one can represent inputs as *finite models*

**Theorem**

*Proofs of \(!\text{Str} \rightarrow !!(1 \oplus 1)\) in ELL decide regular languages.*

Future work: higher depths, logspace conjecture.

Moral: *geometry* (e.g. stratification) and *typing* jointly control complexity; semantics reflects the latter.
Conclusion on semantics

Our complexity results required a finite denotational semantics for MALL2. We built one by *quotienting* the syntax.

**Definition**

\[ \pi \sim_A \pi' \text{ iff for any propositional } B, \text{ for any proof } \rho \text{ of } A \vdash B, \]

\[ \text{cut}(\pi, \rho) \text{ and cut}(\pi', \rho) \text{ have the same normal form}. \]

Future work: Weakening? For variable \( A \), is \( \sim_A \) decidable?
(For fixed \( A \), yes, as corollary of this work.)

P. Pistone and L. Tortora de Falco are investigating a 2\textsuperscript{nd} order extension of Rel characterizing this quotient.