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 $\mathsf{ELEMENTARY}$: Elementary Linear Logic (ELL).

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

[1] Tower of exponentials of height $f(k)$.

Two calculi of elementary complexity (2)

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
  $\mathrm{ord}(\alpha \to \beta) = \max(\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta))$
- Soundness: normalization in
  f(max order in subterm)-EXPTIME
- Completeness: non-obvious, discussed later
  - Corollary: normalization for STλ is non-elementary (Statman 1979)

Side remark: Linear Logic by Levels (L^3) generalizes both

- ELL is a subset of L^3 such that depth = level
- STλ embeds into L^3, sending order to level

Church encodings of inputs in $\mathrm{ST}\lambda$

Church (or Böhm–Berarducci) encodings:

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

For $t : \mathsf{Str}[A] \to \mathsf{Bool}$, $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \mathtt{true}\}$.
Do we get extensional completeness for $\mathsf{ELEMENTARY}$?

Church encodings of inputs in STλ

Church (or Böhm–Berarducci) encodings:

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

For $t : \mathsf{Str}[A] \to \mathsf{Bool}$, $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \mathtt{true}\}$.
Do we get extensional completeness for $\mathsf{ELEMENTARY}$?

No, we get a much much smaller class!

Theorem (Hillebrand & Kanellakis, LICS'96)

The languages decided by STλ*-terms of type* $\mathsf{Str}[A] \to \mathsf{Bool}$ *are exactly the* regular *languages.*

Semantic evaluation in the simply-typed λ-calculus

Theorem (Hillebrand & Kanellakis, LICS'96)

For any type A and any STλ*-term $t : \mathsf{Str}[A] \to \mathsf{Bool}$, the language $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \texttt{true}\}$ is* regular.

Proof: by constructing a deterministic finite automaton.

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

$$\longrightarrow [\![\overline{\varepsilon}]\!] \xrightarrow{0} [\![\overline{0}]\!] \xrightarrow{1} [\![\overline{01}]\!] \xrightarrow{1} [\![\overline{011}]\!] \longrightarrow \cdots$$

$$w \text{ accepted} \Leftrightarrow [\![t\,\overline{w}]\!] = [\![t]\!]\,([\![\overline{w}]\!]) = [\![\texttt{true}]\!] \Leftrightarrow t\,\overline{w} \to_\beta^* \texttt{true}$$

(when $[\![\texttt{true}]\!] \neq [\![\texttt{false}]\!]$, or equivalently $|\,[\![o]\!]\,| \geq 2$) $\square$

Semantic evaluation in the simply-typed λ-calculus

Theorem (Hillebrand & Kanellakis, LICS'96)

For any type A and any STλ*-term $t : \mathsf{Str}[A] \to \mathsf{Bool}$, the language $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\,\overline{w} \to_\beta^* \texttt{true}\}$ is* regular.

Proof: by constructing a deterministic finite automaton.

We use the semantics $[\![-]\!] : \mathrm{ST}\lambda \to \mathsf{FinSet}$, and build a DFA with states $Q = [\![\mathsf{Str}[A]]\!]$, acceptation as $[\![t]\!]\,(-) = [\![\texttt{true}]\!]$.
($|Q| < \infty$, e.g. $2^{2^{33}}$ when $A = \mathsf{Bool}$)

$$\longrightarrow [\![\varepsilon]\!] \xrightarrow{0} [\![0]\!] \xrightarrow{1} [\![01]\!] \xrightarrow{1} [\![011]\!] \longrightarrow \cdots$$

$$w \text{ accepted} \Leftrightarrow [\![t\,\overline{w}]\!] = [\![t]\!]\,([\![\overline{w}]\!]) = [\![\texttt{true}]\!] \Leftrightarrow t\,\overline{w} \to_\beta^* \texttt{true}$$

(when $[\![\texttt{true}]\!] \neq [\![\texttt{false}]\!]$, or equivalently $|\,[\![o]\!]\,| \geq 2$) $\square$

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 STλ *at fixed order*:

Theorem (Terui, RTA'12)

Normalizing an STλ*-term of type* 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.

β-reduce to halve order, then evaluate in LL Scott model. $\square$

Regular languages in ELL

Application to second-order ELL at fixed depth

Now, $\mathsf{Str} = \forall X.\,!(X \multimap X) \multimap !(X \multimap X) \multimap !(X \multimap X)$

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap !!(1 \oplus 1)$ *in 2*nd *order elementary affine logic* with recursive types *decide exactly the languages in* P.

Application to second-order ELL at fixed depth

Now, $\mathsf{Str} = \forall X.\, !(X \multimap X) \multimap\, !(X \multimap X) \multimap\, !(X \multimap X)$

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap\, !!(1 \oplus 1)$ *in* 2^{nd} *order elementary affine logic* with recursive types *decide exactly the languages in* P.

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

Theorem

The proofs of $!\mathsf{Str} \multimap\, !!(1 \oplus 1)$ *in* 2^{nd} *order* ELL *decide exactly the* regular *languages.*

(First characterization of regular languages in a type system with impredicative quantification?)

MALL semantics $\Rightarrow$ ELL complexity

For semantic evaluation in ELL, we need a non-trivial *finite* semantics for *2^{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.) $\square$

New result of independent interest, with no mention of complexity.

Proof ideas (1)

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

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\widehat{\pi}}{\vdash \mathsf{Str}[A_1]^\perp, \ldots, \mathsf{Str}[A_n]^\perp, !(1 \oplus 1)}}{\vdash \mathsf{Str}^\perp, \ldots, \mathsf{Str}^\perp, !(1 \oplus 1)}}{\vdash ?\mathsf{Str}^\perp, \ldots, ?\mathsf{Str}^\perp, !!(1 \oplus 1)}}{\vdash ?\mathsf{Str}^\perp, !!(1 \oplus 1)}}{\vdash !\mathsf{Str} \multimap !!(1 \oplus 1)}
$$

- For $w \in \{0,1\}^*$, $\overline{w} : \mathsf{Str}$, $\pi(!\overline{w}) = !\widehat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$
- W.l.o.g. $\mathrm{depth}(\pi) = 2$, thanks to *stratification*
  - "Erasing" all depth > 2 exponentials doesn't change $\pi(!\overline{w})$
- In other words w.l.o.g. $A_1, \ldots, A_n \in \mathrm{MALL}$

Proof ideas (2)

- We want to know if $\pi(!\overline{w}) = !\widehat{\pi}(\overline{w}[A_1], \ldots, \overline{w}[A_n])$ is $!!\texttt{true}$
- $\overline{w}[A] : !(A \multimap A) \multimap !(A \multimap A) \multimap !(A \multimap A)$, $A \in \mathrm{MALL}$
- Using finite MALL semantics $[\![-]\!]$, $\overline{w}$ induces map

$$\|w\|_A : [\![A \multimap A]\!] \times [\![A \multimap A]\!] \to [\![A \multimap A]\!]$$

Lemma

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

- By Church encoding definition and $[\![A \multimap A]\!] \simeq \mathrm{End}([\![A]\!])$,

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

- Our language is a preimage for a morphism to the finite monoid $\prod_{i=1}^n \mathrm{End}([\![A_i]\!])^{\mathrm{End}([\![A_i]\!])^2}$, therefore regular $\square$

What about higher depths?

Theorem (Baillot, APLAS'11)

The proofs of $!\mathsf{Str} \multimap !^{k+2}(1 \oplus 1)$ *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 $(D, \leq^D, S^D)$, $|D| < \infty$. $S^D(d)$ = "d^{th} 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 STλ and extensional completeness

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

- Represent k-ary relations as lists of k-tuples[2]
  $\mathsf{Rel}_k[A] = (d^k \to A \to A) \to A \to A$
- Provide a list of all elements in domain ($\mathsf{List}[A] = \mathsf{Rel}_1[A]$)

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

Terms of type $\mathsf{List}[A] \to \mathsf{Rel}_{k_1}[A_1] \to \ldots \to \mathsf{Rel}_{k_m}[A_m] \to \mathsf{Bool}$ *in* STλ *compute exactly* ELEMENTARY *queries over finite models.*

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

[2] In the spirit of database theory: relation = set of records.

Finite models in ELL

We transpose this idea to second-order ELL:

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

$$\mathsf{Inp} = \exists D.\, !\mathsf{List} \otimes \bigotimes_{i=1}^{n} !!\mathsf{Rel}_{k_i} \otimes !!(D \multimap D \otimes D) \otimes !!D^{\perp}$$

- 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* $\mathsf{Inp} \multimap !!(1 \oplus 1)$.

Proof idea: compute transitive closure of a relation $\mathcal{R} \subseteq D^k \times D^k$ by iterating $\varphi_{\mathcal{R}} : \mathcal{P}(D^k \times D^k) \to \mathcal{P}(D^k \times D^k)$.
Determinism of $\mathcal{R}$ ensures *linearity*: $\varphi_{\mathcal{R}} : \mathsf{Rel}_{2k} \multimap \mathsf{Rel}_{2k}$ in ELL.
This is remarkable enough to hope for:

Conjecture

Conversely, proofs of $\mathsf{Inp} \multimap !!(1 \oplus 1)$ *only decide* L *queries.*

A finite denotational semantics for second-order MALL

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^{nd} 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^{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 ρ of $A \vdash B$, $\mathbf{cut}(\pi, \rho)$ and $\mathbf{cut}(\pi', \rho)$ have the same normal form.

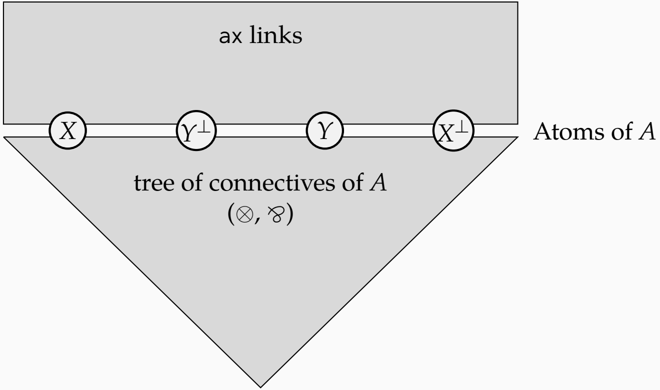
- $\sim$ is a congruence: the quotient is a model of MALL2
- A existential-free $\Rightarrow$ $\sim_A$ trivial
- Example: the proofs of $\exists X.\, X$ cannot be distinguished
  - Impredicative encodings of units work, e.g. $\top \equiv \exists X.\, X$.

Theorem

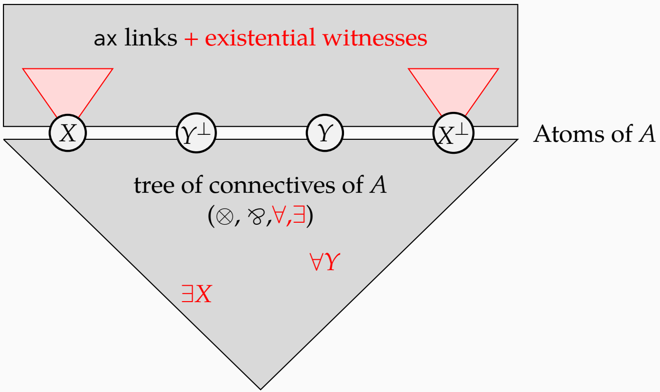
For any MALL2 *formula A, there are* finitely many *classes for $\sim_A$.*

Next: proof in the MLL case, using unit-free proof nets.

What a propositional MLL proof net looks like

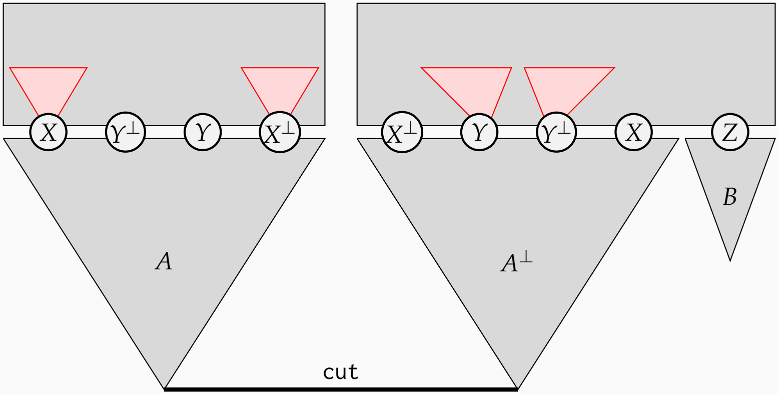


What a MLL2 proof net looks like

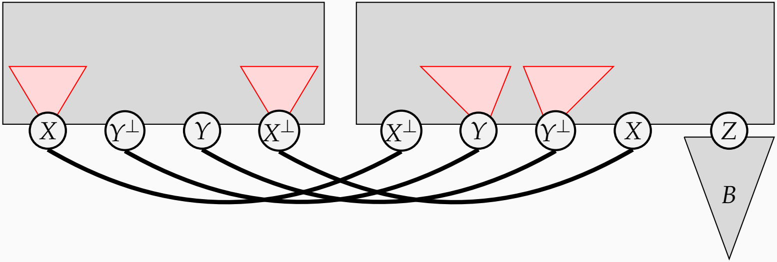


Finiteness theorem (1): proof/observation interaction

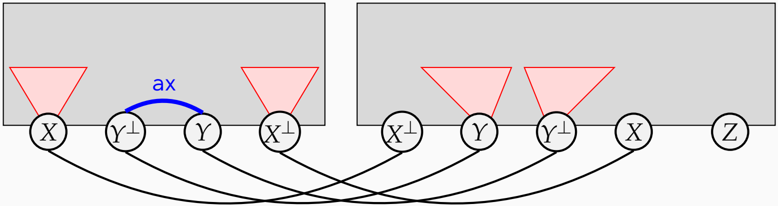
A: MLL2 formula; B: propositional MLL formula



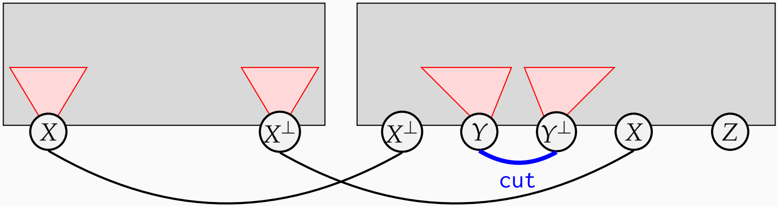
A: MLL2 formula; B: propositional MLL formula



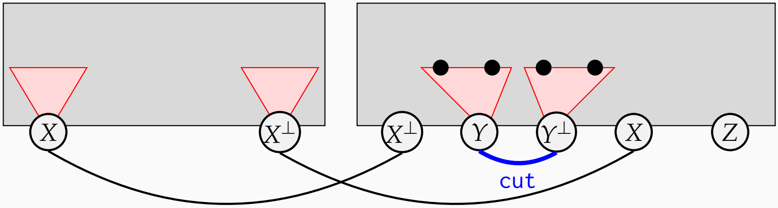
Finiteness theorem (2): eliminating two cuts



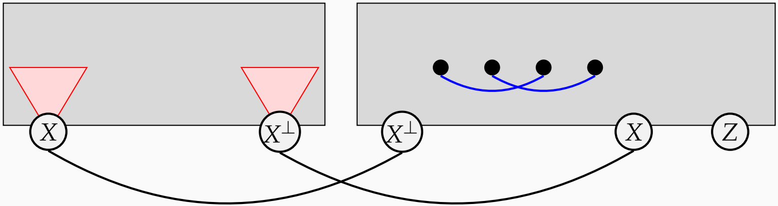
Finiteness theorem (2): eliminating two cuts



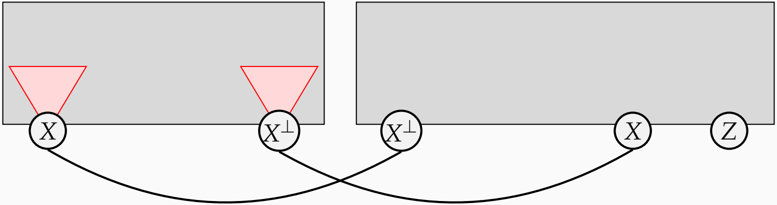
Finiteness theorem (2): eliminating two cuts



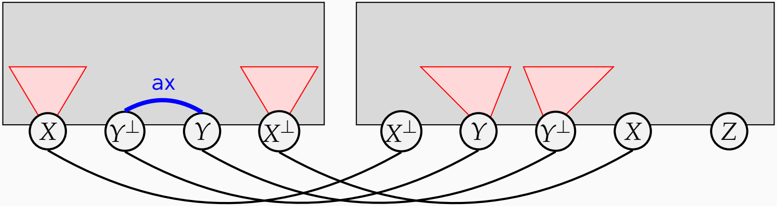
Finiteness theorem (2): eliminating two cuts



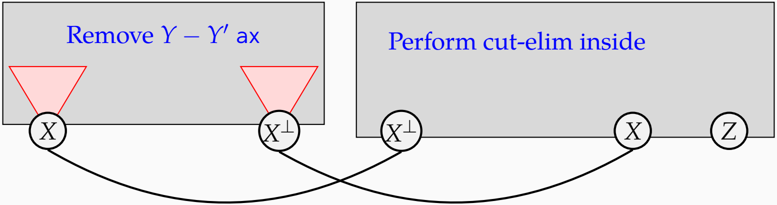
Finiteness theorem (2): eliminating two cuts



Finiteness theorem (3): big-step reduction

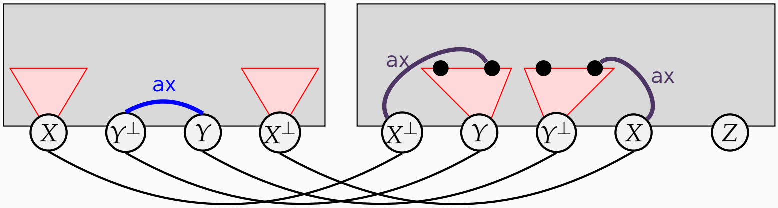


$\Downarrow$ big-step reduction



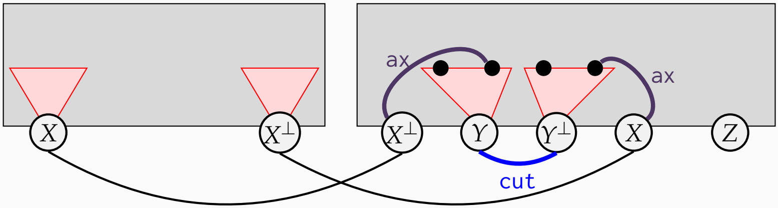
Finiteness theorem (4): normalization

New redexes may appear during reduction.



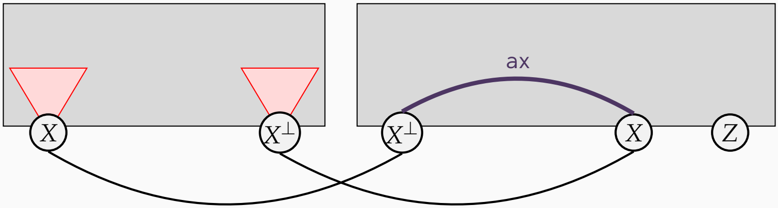
Finiteness theorem (4): normalization

New redexes may appear during reduction.



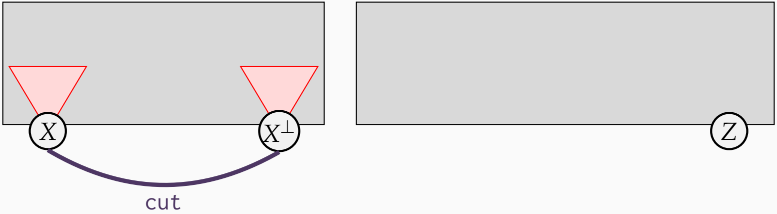
Finiteness theorem (4): normalization

New redexes may appear during reduction.



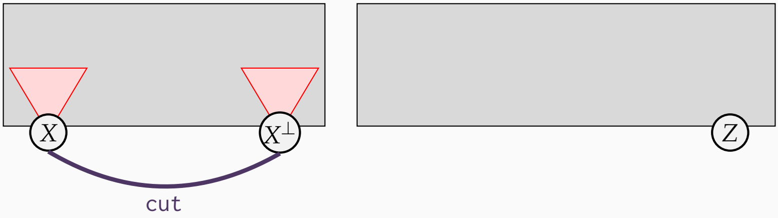
Finiteness theorem (4): normalization

New redexes may appear during reduction.



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* $\Rightarrow$.

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.

Finiteness theorem (6): concluding the proof

Lemma (big-step normalization)

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

Theorem

$\sim_A$ has finitely many equivalence classes.

Proof.

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

- all π's with the same strategy are equivalent for $\sim_A$
- there are finitely many strategies $\square$

Summary and conclusion

Conclusion on complexity

ELL and STλ both capture $\mathsf{ELEMENTARY}$; by fixing a parameter (depth/order) one can characterize lower complexity classes.

We brought techniques from the STλ tradition to 2nd 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 $!\mathsf{Str} \multimap !!(1 \oplus 1)$ *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

$\pi \sim_A \pi'$ iff for any *propositional* B, for any proof ρ of $A \vdash B$, $\mathbf{cut}(\pi, \rho)$ and $\mathbf{cut}(\pi', \rho)$ 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[nd] order extension of Rel characterizing this quotient.