A semantic conjecture on second-order MLL and its complexity consequences

Nguyễn Lê Thành Dũng[1,2] Thomas Seiller[2]

[1]École normale supérieure de Paris

[2]Laboratoire d'informatique de Paris Nord, CNRS / Université Paris 13

Linearity/TLLA 2018 (FLoC workshop)
Oxford, July 8th, 2018

Languages decided by the simply-typed λ-calculus

- For a simple type A, $\mathsf{Str}[A] = (A \to A) \to (A \to A) \to (A \to A)$
- Church encoding: $\overline{w} = \lambda f_0.\, \lambda f_1.\, \lambda x.\, f_{w[0]}\, (\ldots (f_{w[n-1]}\, x) \ldots) : \mathsf{Str}[A]$
  - Holds for all A (meta-$\forall$)
- $\mathsf{Bool} = o \to o \to o$ (o base type)
- For $t : \mathsf{Str}[A] \to \mathsf{Bool}$, $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\, \overline{w} \to_\beta^* \texttt{true}\}$

Languages decided by the simply-typed λ-calculus

- For a simple type A, $\mathsf{Str}[A] = (A \to A) \to (A \to A) \to (A \to A)$
- Church encoding: $\overline{w} = \lambda f_0.\, \lambda f_1.\, \lambda x.\, f_{w[0]}\, (\ldots (f_{w[n-1]}\, x) \ldots) : \mathsf{Str}[A]$
  - Holds for all A (meta-$\forall$)
- $\mathsf{Bool} = o \to o \to o$ (o base type)
- For $t : \mathsf{Str}[A] \to \mathsf{Bool}$, $\mathcal{L}(t) = \{w \in \{0,1\}^* \mid t\, \overline{w} \to_\beta^* \texttt{true}\}$

Theorem (Hillebrand & Kanellakis 1996)

For any type A and any simply-typed λ-term $t : \mathsf{Str}[A] \to \mathsf{Bool}$, $\mathcal{L}(t)$ is regular.

- Conversely, all regular languages can be obtained as $\mathcal{L}(t)$ for some A and some t

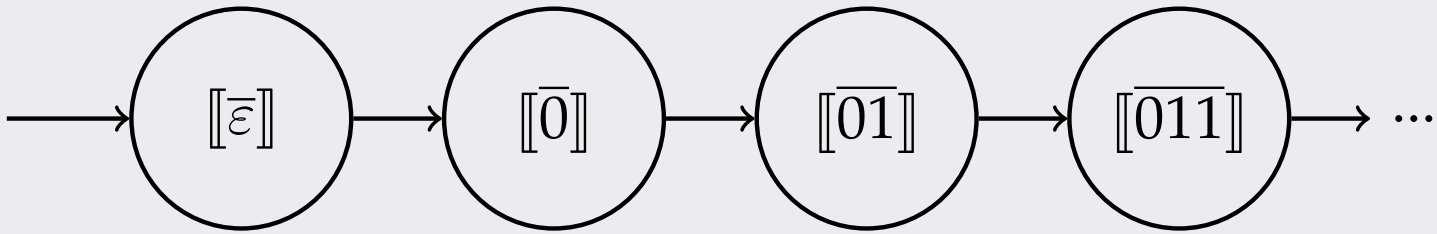
Semantic evaluation in the simply-typed λ-calculus

Theorem (Hillebrand & Kanellakis 1996)

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

Proof: by constructing a deterministic finite automaton.

We use the denotational semantics $[\![-]\!] : \mathrm{ST}\lambda\mathrm{C} \to \mathsf{FinSet}$, and build a DFA with states $Q = [\![\mathsf{Str}[A]]\!]$.



Define accepting states s.t.

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

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

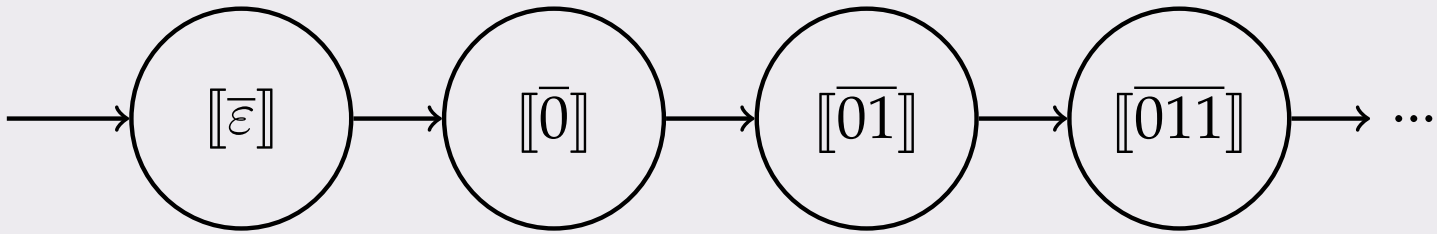
Semantic evaluation in the simply-typed λ-calculus

Theorem (Hillebrand & Kanellakis 1996)

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

Proof: by constructing a deterministic finite automaton.

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



Define accepting states s.t.

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

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

Semantic evaluation in propositional linear logic

- Main ingredient:
  a model in which each type has *finitely many* values
- Also exists in propositional LL, e.g. finite coherence spaces with $[\![!A]\!] = \mathcal{P}_{\text{fin}}([\![A]\!])$
- Using $\mathsf{Str}_{\mathrm{LL}}[A] = !(A \multimap A) \multimap !(A \multimap A) \multimap (A \multimap A)$:

Theorem

For any propositional LL formula A and any proof t of $!\mathsf{Str}_{\mathrm{LL}}[A] \multimap 1 \oplus 1$, $\mathcal{L}_{\mathrm{LL}}(t)$ is regular.

Semantic evaluation in propositional linear logic

- Main ingredient:
  a model in which each type has *finitely many* values
- Also exists in propositional LL, e.g. finite coherence spaces with $[\![!A]\!] = \mathcal{P}_{\text{fin}}([\![A]\!])$
- Using $\mathsf{Str}_{\mathrm{LL}}[A] = !(A \multimap A) \multimap !(A \multimap A) \multimap (A \multimap A)$:

Theorem

For any propositional LL formula A and any proof t of $!\mathsf{Str}_{\mathrm{LL}}[A] \multimap 1 \oplus 1$, $\mathcal{L}_{\mathrm{LL}}(t)$ is regular.

- What about *polymorphic* typed calculi?

Semantic evaluation in elementary affine logic

- EAL (resp. MAL): elementary (resp. multiplicative) affine logic

Conjecture

2^{nd} order MAL *has a non-trivial finite semantics.*

Corollary

The proofs of $!\mathsf{Str}_{\mathrm{EAL}} \multimap !!\mathsf{Bool}_{\mathrm{EAL}}$ *in* 2^{nd} *order* EAL *decide exactly the regular languages.*

Semantic evaluation in elementary affine logic

- EAL (resp. MAL): elementary (resp. multiplicative) affine logic

Conjecture

2^{nd} order MAL *has a non-trivial finite semantics.*

Corollary

The proofs of $!\mathsf{Str}_{\mathrm{EAL}} \multimap !!\mathsf{Bool}_{\mathrm{EAL}}$ *in* 2^{nd} *order* EAL *decide exactly the regular languages.*

- Compare with:

Theorem (Baillot 2011)

The proofs of $!\mathsf{Str}_{\mathrm{EAL}} \multimap !!\mathsf{Bool}_{\mathrm{EAL}}$ *in* 2^{nd} *order* EAL with recursive types *decide exactly the* polynomial-time computable *languages.*

A semantic theorem on second-order MLL and its complexity consequences

Nguyễn Lê Thành Dũng[1,2] Thomas Seiller[2]

[1]École normale supérieure de Paris

[2]Laboratoire d'informatique de Paris Nord, CNRS / Université Paris 13

Linearity/TLLA 2018 (FLoC workshop)
Oxford, July 8th, 2018

A finite semantics for second-order MLL

- MLL2 = second-order MLL

Theorem

MLL2 *has a non-trivial finite semantics.*

- Remark: in propositional MLL, each formula has finitely many cut-free proofs, i.e. the syntactic model is finite
- 2^{nd} order case: arbitrarily large $\exists$ witnesses

A finite semantics for second-order MLL

- MLL2 = second-order MLL

Theorem

MLL2 *has a non-trivial finite semantics.*

- Remark: in propositional MLL, 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

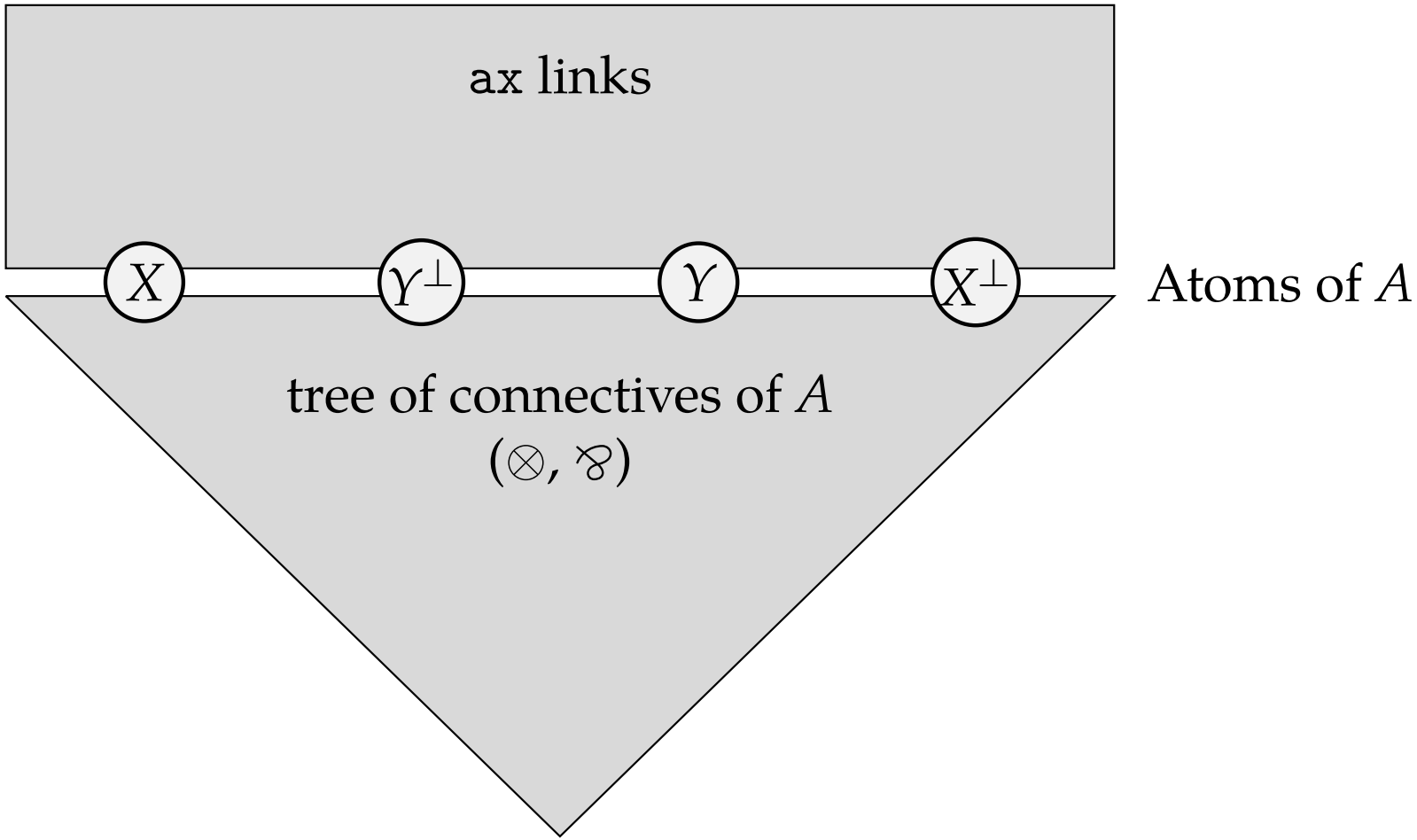
Let A be a MLL2 formula and π, π' be proofs of A. Define $\pi \sim_A \pi'$ as: for any *propositional* MLL formula B, for any proof ρ of $A \vdash B$, $\mathbf{cut}(\pi, \rho)$ and $\mathbf{cut}(\pi', \rho)$ have the same normal form.

- $\sim$ is a congruence: the quotient is a model of second-order MLL
- A existential-free $\Rightarrow \sim_A$ trivial
- Example: the proofs of $\exists X.\, X$ cannot be distinguished
  - thus $\exists X.\, X \equiv \top$ in this model
- Remark by P. Pistone: dinatural equivalence $\subsetneq \sim$

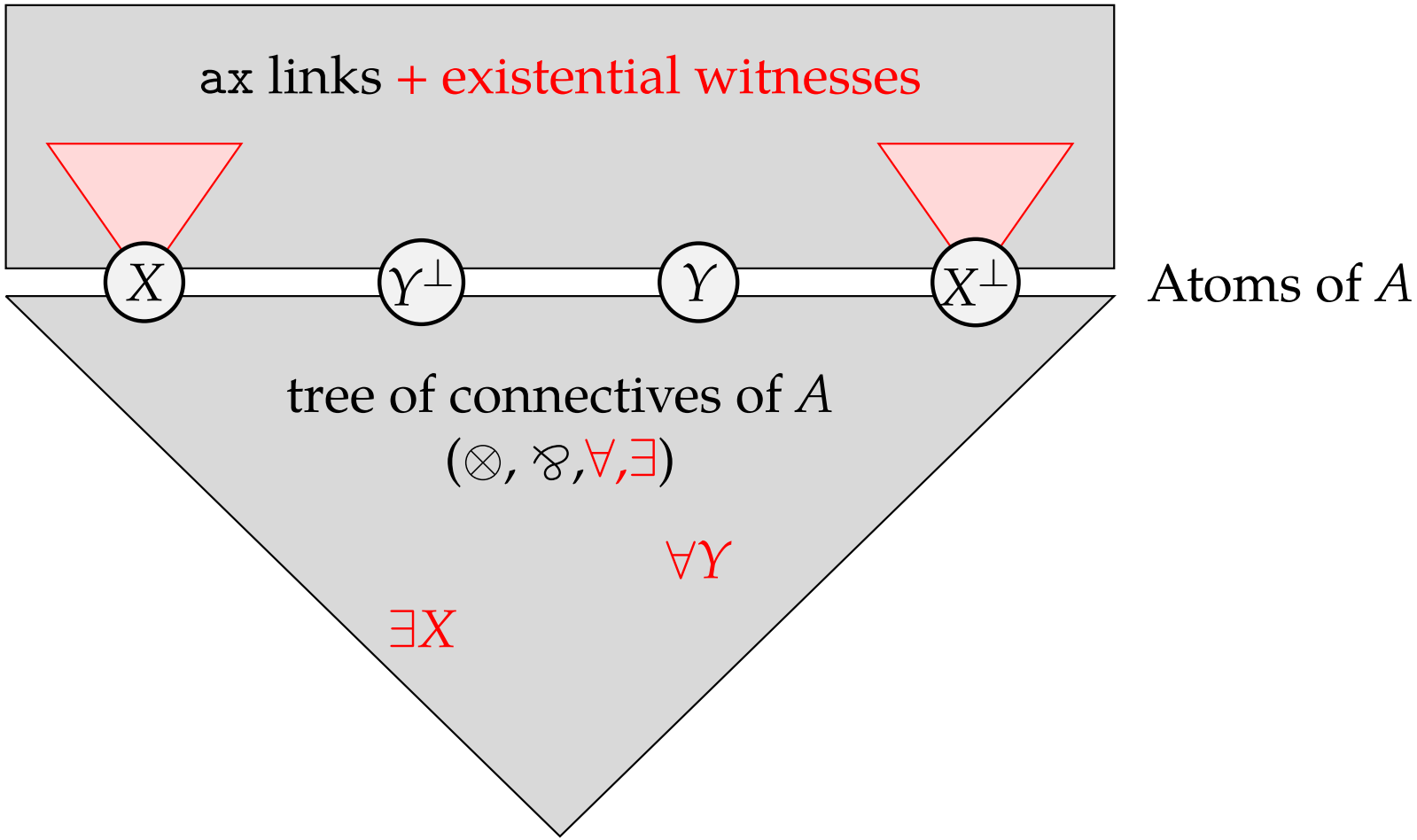
Theorem

For any MLL2 *formula A, there are* finitely many *equivalence classes for $\sim_A$.*

What a propositional MLL proof net looks like

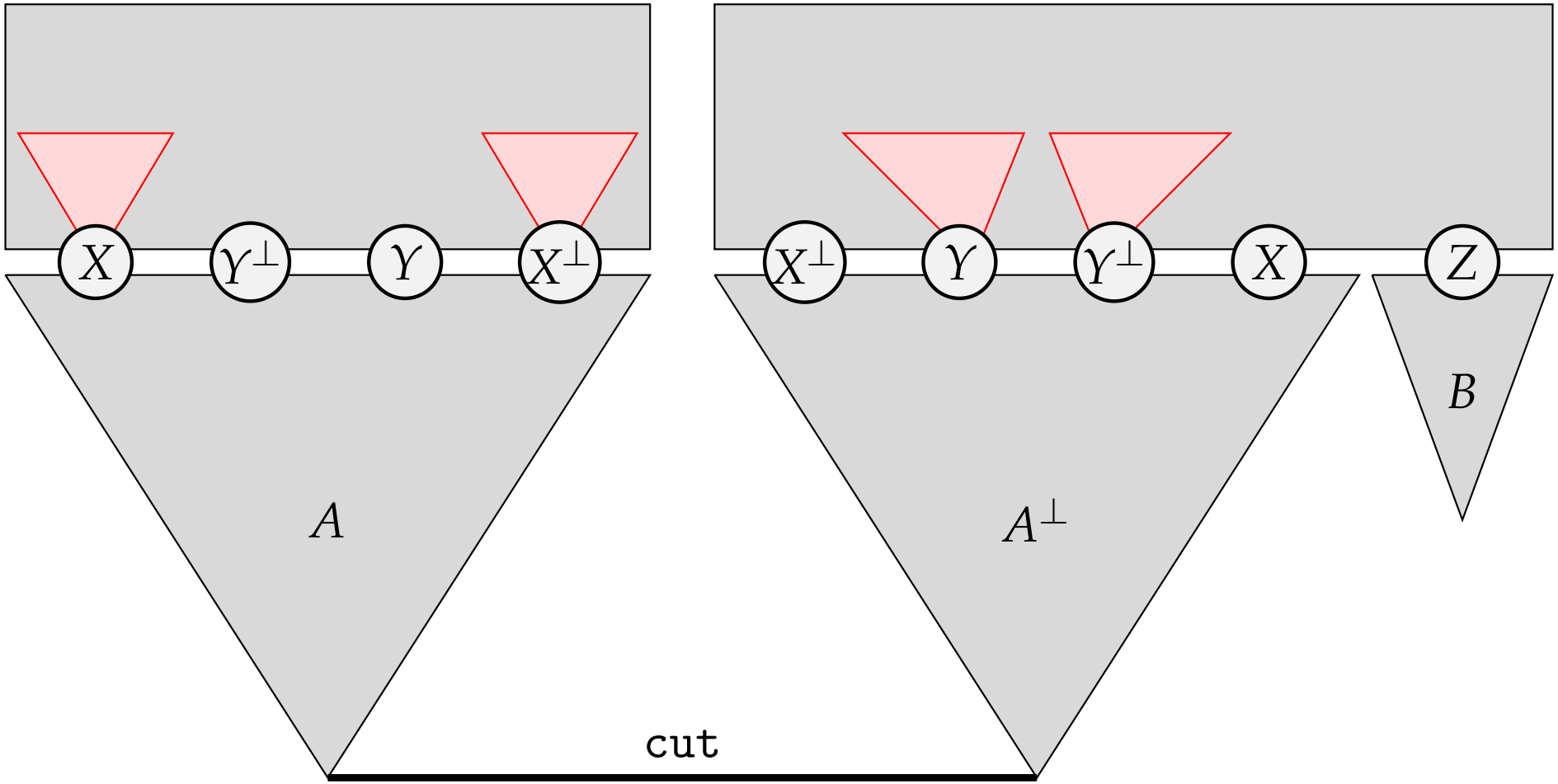


What a MLL2 proof net looks like



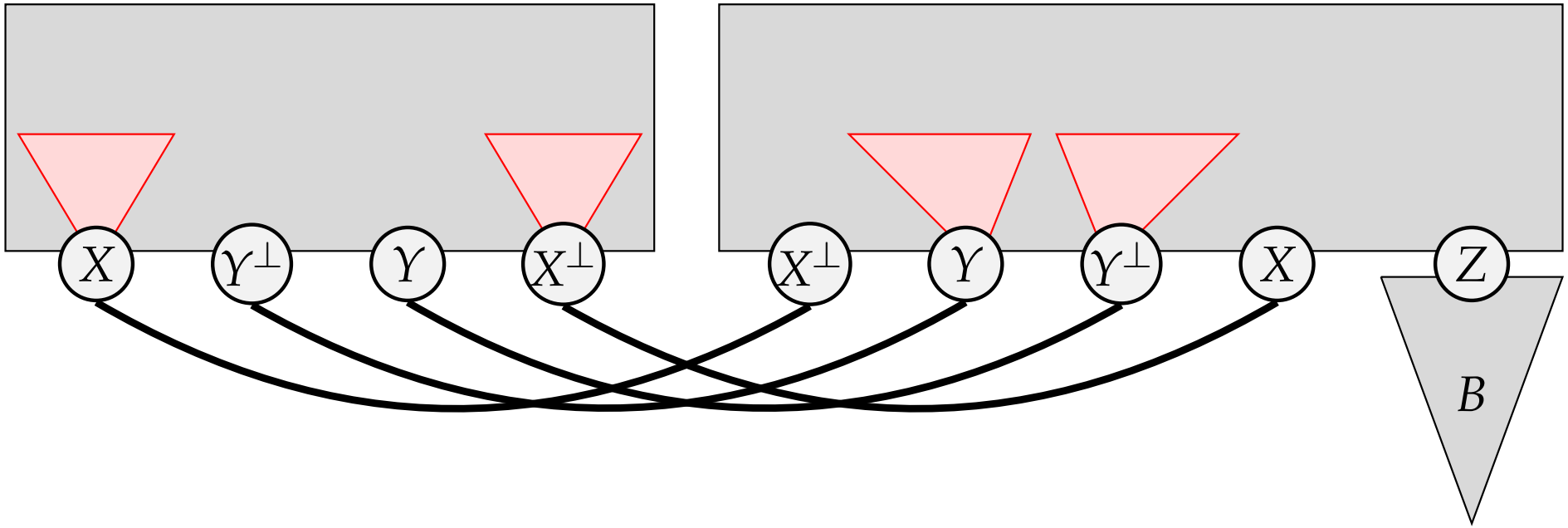
Finiteness theorem (1): proof/observation interaction

- A: MLL2 formula; B: propositional MLL formula

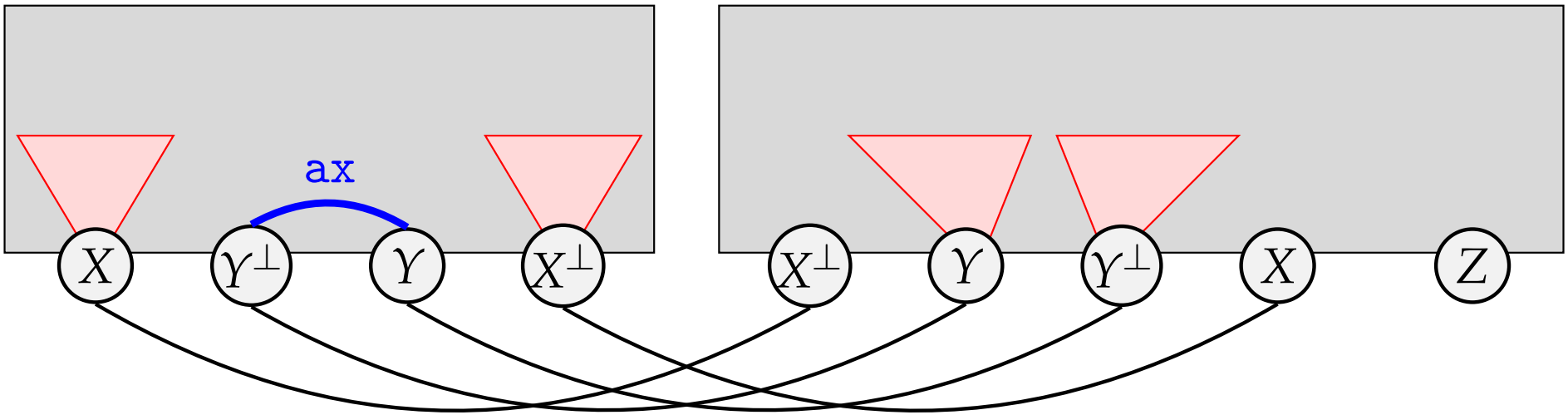


Finiteness theorem (1): proof/observation interaction

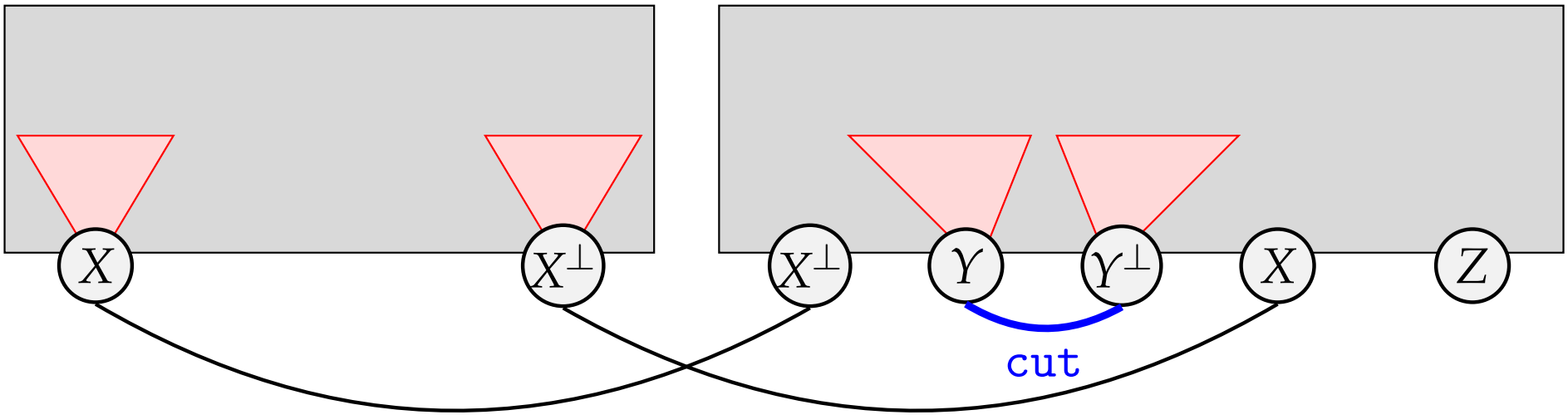
- A: MLL2 formula; B: propositional MLL formula



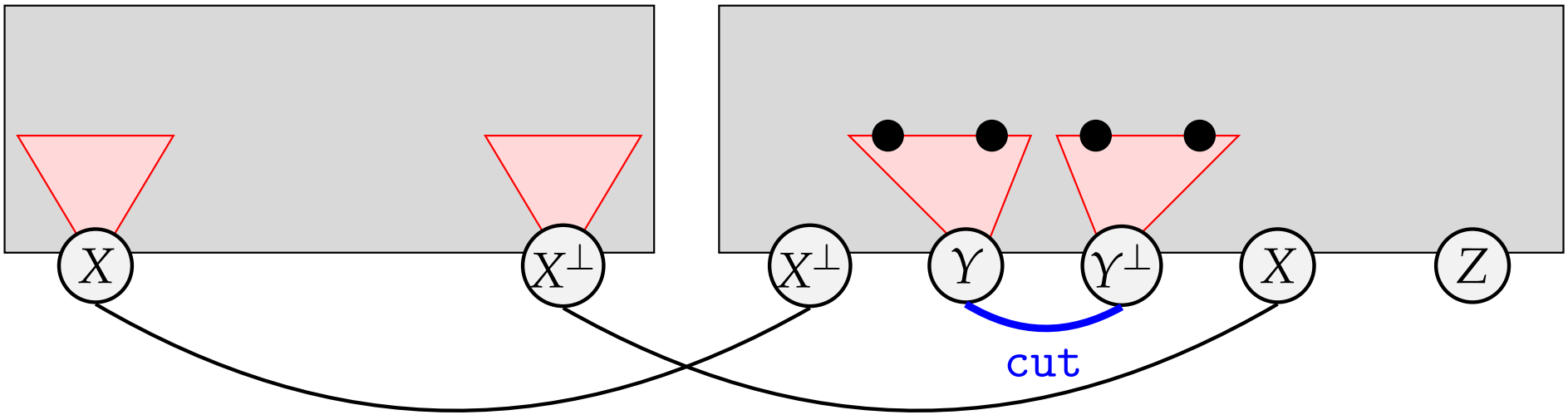
Finiteness theorem (2): eliminating two cuts



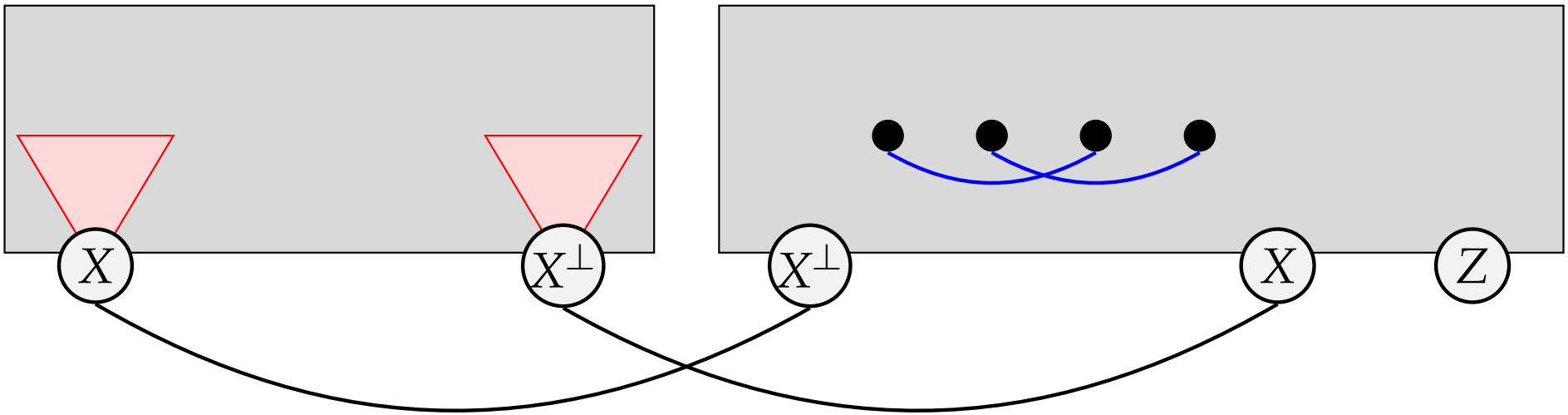
Finiteness theorem (2): eliminating two cuts



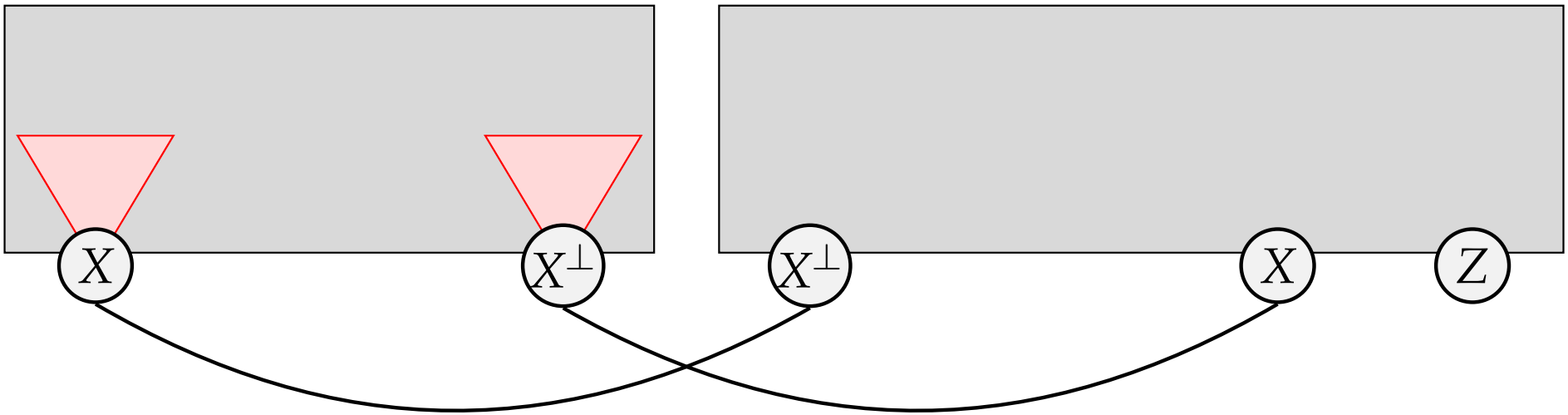
Finiteness theorem (2): eliminating two cuts



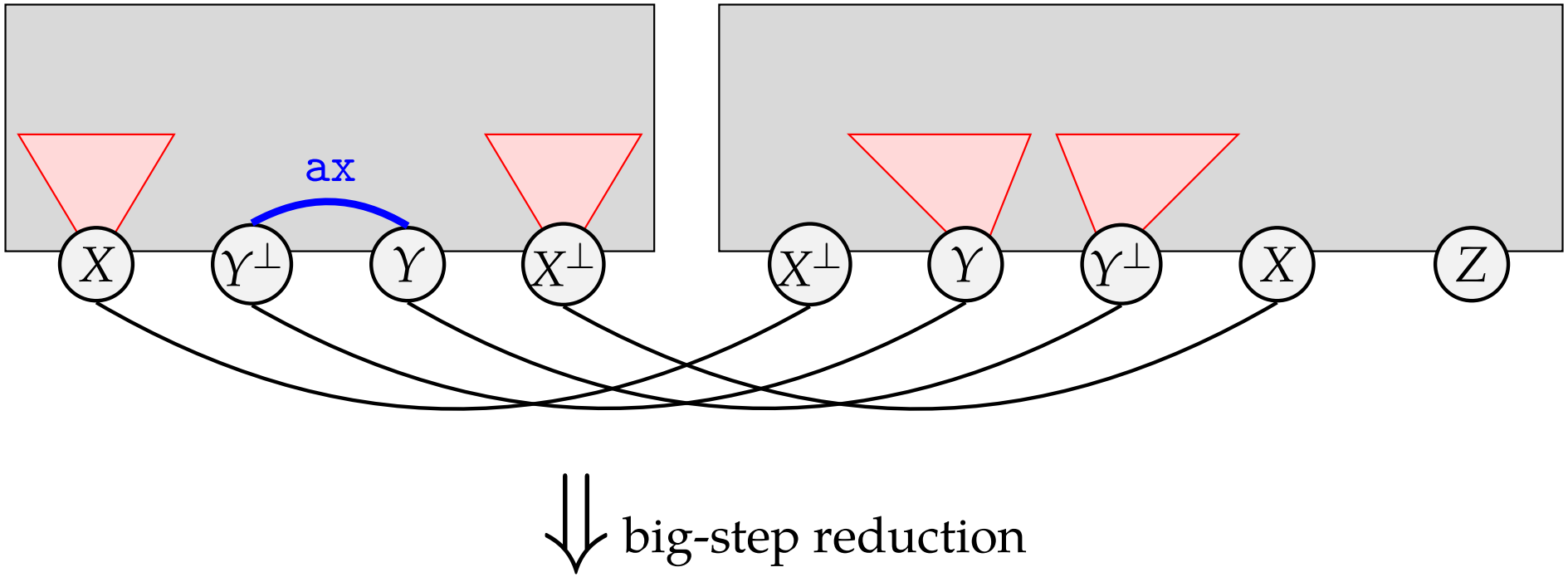
Finiteness theorem (2): eliminating two cuts



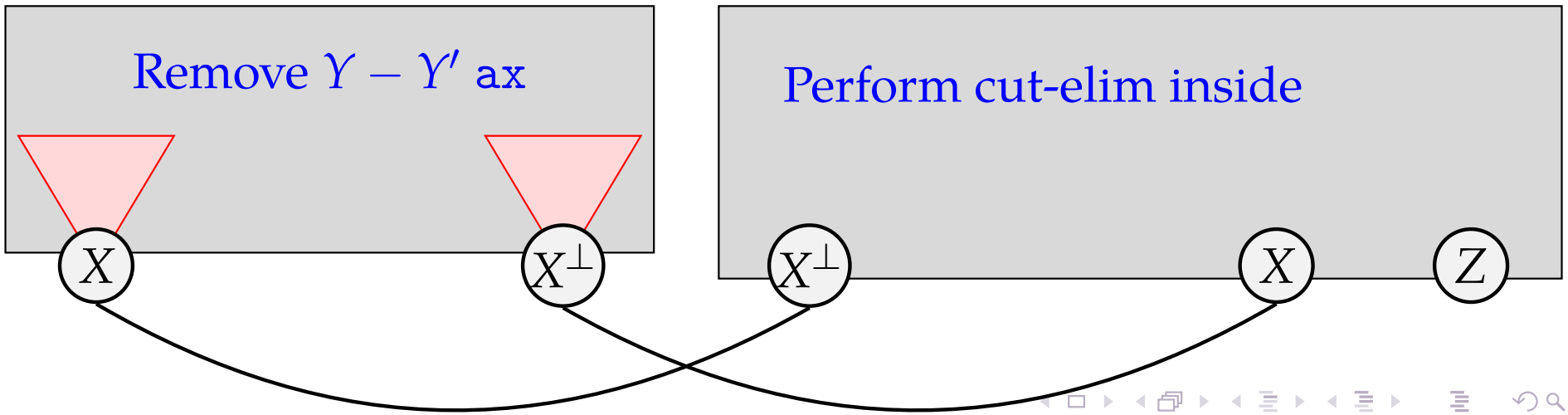
Finiteness theorem (2): eliminating two cuts



Finiteness theorem (3): big-step reduction



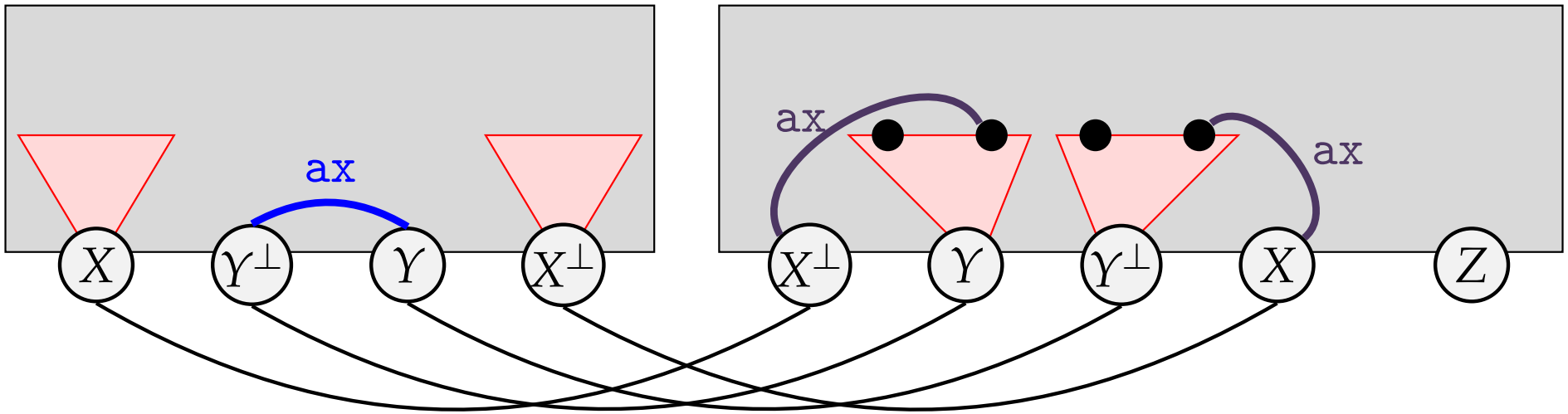
big-step reduction



Finiteness theorem (4): normalization

Lemma (Progress)

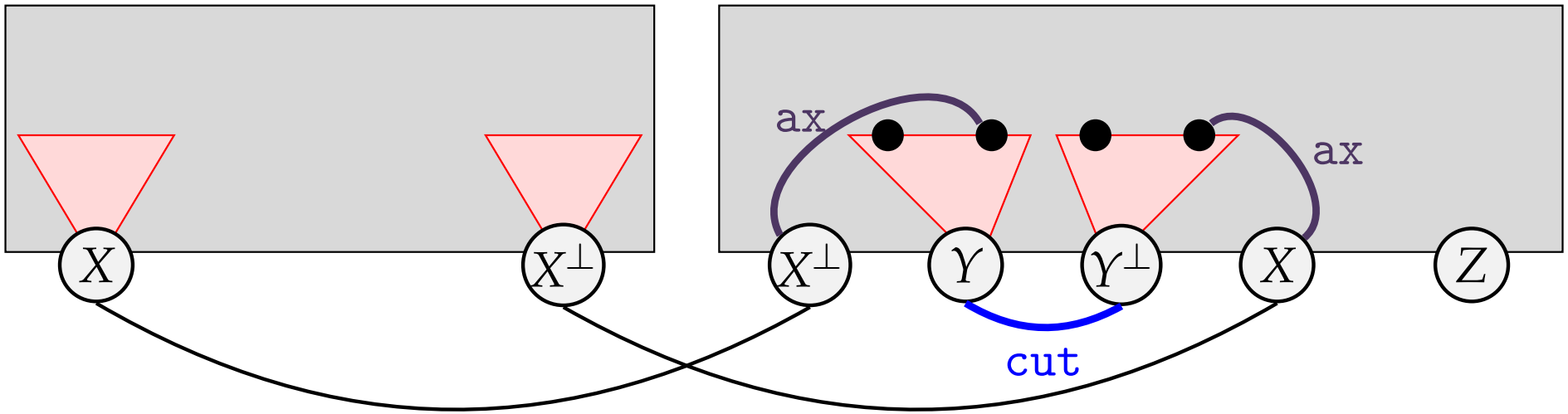
As long as there remains a `cut`*-link, there is a redex for* $\Rightarrow$.



Finiteness theorem (4): normalization

Lemma (Progress)

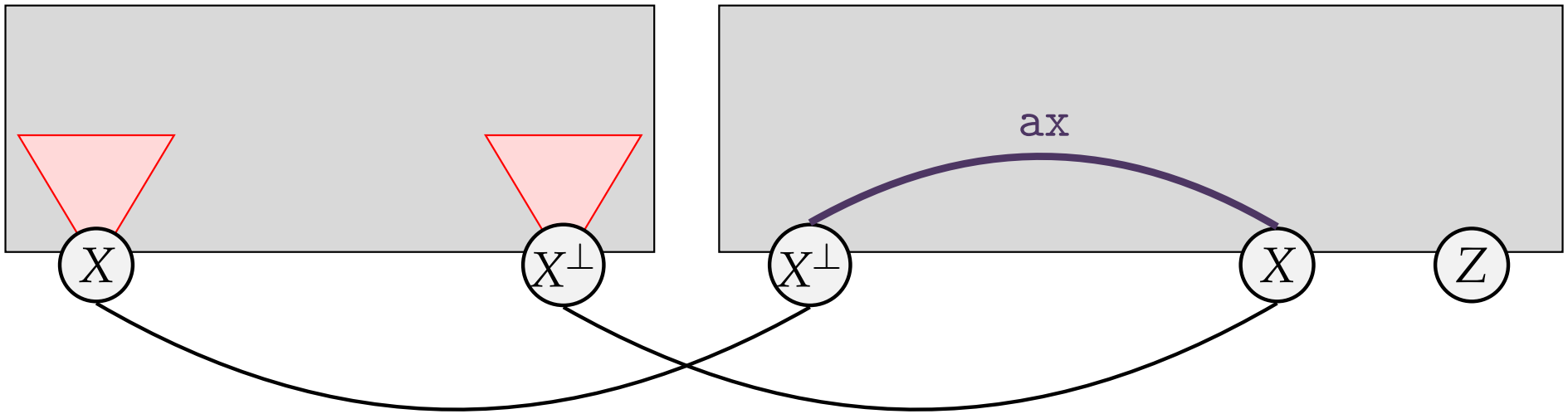
As long as there remains a `cut`*-link, there is a redex for* $\Rightarrow$.



Finiteness theorem (4): normalization

Lemma (Progress)

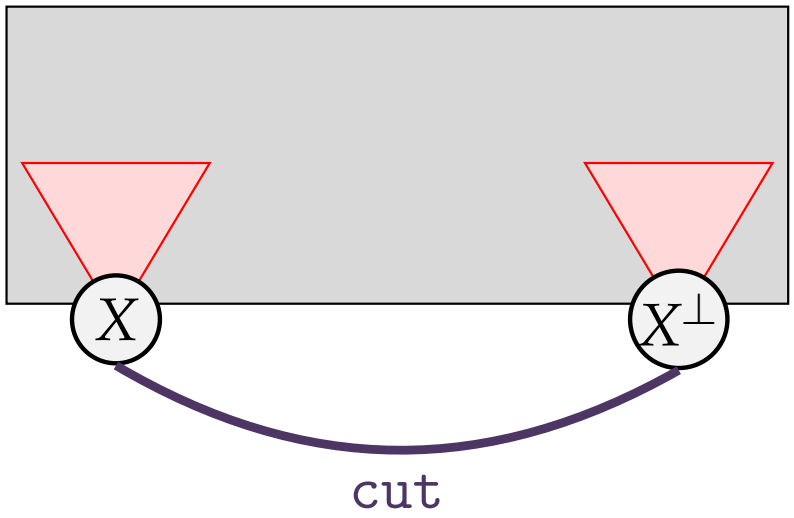
As long as there remains a `cut`*-link, there is a redex for* $\Rightarrow$.



Finiteness theorem (4): normalization

Lemma (Progress)

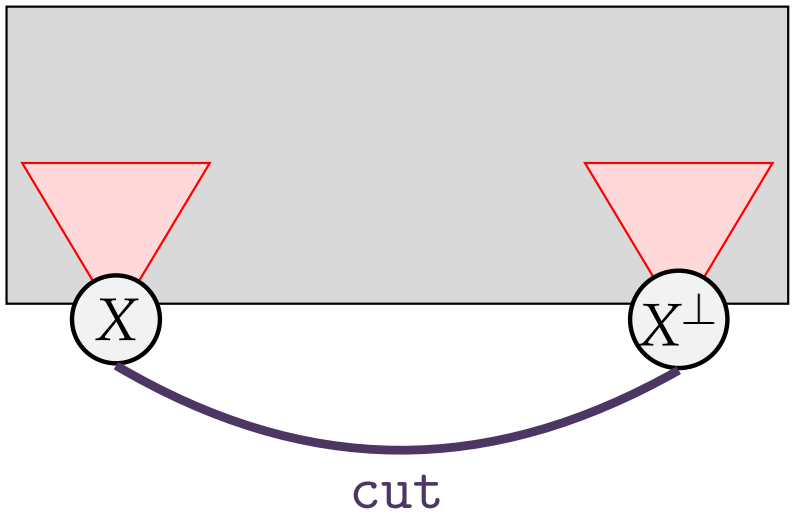
As long as there remains a `cut`*-link, there is a redex for* $\Rightarrow$.



Finiteness theorem (4): normalization

Lemma (Progress)

As long as there remains a `cut`*-link, there is a redex for* $\Rightarrow$.



- Progress is the tricky part of the proof

Finiteness theorem (4): concluding the proof

Lemma

$\Rightarrow^*$ *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

□

Conclusion

- A finite observational quotient of MLL2
- An implicit complexity application:
  characterization of regular languages in ELL
- Future work on semantics:
  - Additives, weakening
  - Decidability of $\sim_A$ for *varying* A?
    (For *fixed* A, it's a corollary of "effective" finiteness)
- Future work on complexity:
  - Languages decided by $!\mathsf{Str}_{\mathrm{EAL}} \multimap !^k\mathsf{Bool}_{\mathrm{EAL}}$ for $k \geq 3$?
    (For EAL with recursive types, $(k-2)$-ExpTime)
  - Alternative input representations

Conclusion

- A finite observational quotient of MLL2
- An implicit complexity application:
  characterization of regular languages in ELL
- Future work on semantics:
  - Additives, weakening
  - Decidability of $\sim_A$ for *varying* A?
    (For *fixed* A, it's a corollary of "effective" finiteness)
- Future work on complexity:
  - Languages decided by $!\mathsf{Str}_{\mathrm{EAL}} \multimap !^k\mathsf{Bool}_{\mathrm{EAL}}$ for $k \geq 3$?
    (For EAL with recursive types, $(k-2)$-ExpTime)
  - Alternative input representations
- One thing to remember from this talk:

Theorem (Hillebrand & Kanellakis 1996)

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