Multiplicative-Additive Proof Equivalence is Logspace-complete

Marc Bagnol — *JSPS postdoc at MMM, University of Tokyo*

Proof equivalence and proofnets

The equivalence problem

In formal systems, different ways of describing the “same” object.

The equivalence problem

In formal systems, different ways of describing the "same" object.

At a semantic level: imagine a logic $\mathbf{L}$ in sequent calculus, with a `cut`-elimination procedure. We say that

Two $\mathbf{L}$ proofs π and ν (`cut`-free) are equivalent *iff* they have the same interpretation in all *denotational semantics** of $\mathbf{L}$

*(*categorical interpretations that collapse* `cut`*-elimination to identity)*

The equivalence problem

In formal systems, different ways of describing the "same" object.

At a semantic level: imagine a logic $\mathbf{L}$ in sequent calculus, with a `cut`-elimination procedure. We say that

Two $\mathbf{L}$ proofs π and ν (`cut`-free) are equivalent *iff* they have the same interpretation in all *denotational semantics** of $\mathbf{L}$

*(*categorical interpretations that collapse* `cut`*-elimination to identity)*

Example:

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D} \qquad \dfrac{\langle\mu\rangle}{B, C \vdash D}}{A \oplus B, C \vdash D}\,\oplus^\star}{A \oplus B \vdash C \multimap D}\,\multimap$$

The equivalence problem

In formal systems, different ways of describing the "same" object.

At a semantic level: imagine a logic $\mathbf{L}$ in sequent calculus, with a `cut`-elimination procedure. We say that

Two $\mathbf{L}$ proofs π and ν (cut-free) are equivalent *iff* they have the same interpretation in all *denotational semantics** of $\mathbf{L}$

*(*categorical interpretations that collapse* `cut`*-elimination to identity)*

Example:

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D} \qquad \dfrac{\langle\mu\rangle}{B, C \vdash D}}{A \oplus B, C \vdash D}\,\oplus^\star}{A \oplus B \vdash C \multimap D}\,\multimap \qquad \sim \qquad \dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D}}{A \vdash C \multimap D}\,\multimap \qquad \dfrac{\dfrac{\langle\mu\rangle}{B, C \vdash D}}{B \vdash C \multimap D}\,\multimap}{A \oplus B \vdash C \multimap D}\,\oplus^\star$$

The equivalence problem

Example:

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D} \qquad \dfrac{\langle\mu\rangle}{B, C \vdash D}}{A \oplus B, C \vdash D}\,\oplus^{\star}}{A \oplus B \vdash C \multimap D}\,\multimap \qquad \sim \qquad \dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D}}{A \vdash C \multimap D}\,\multimap \qquad \dfrac{\dfrac{\langle\mu\rangle}{B, C \vdash D}}{B \vdash C \multimap D}\,\multimap}{A \oplus B, C \vdash D}\,\oplus^{\star}$$

In certain cases, the notion can be captured syntactically by a list of similar *rule permuations*.

The equivalence problem

Example:

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D} \qquad \dfrac{\langle\mu\rangle}{B, C \vdash D}}{A \oplus B, C \vdash D}\,\oplus^{\star}}{A \oplus B \vdash C \multimap D}\,\multimap \qquad \sim \qquad \dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D}}{A \vdash C \multimap D}\,\multimap \qquad \dfrac{\dfrac{\langle\mu\rangle}{B, C \vdash D}}{B \vdash C \multimap D}\,\multimap}{A \oplus B, C \vdash D}\,\oplus^{\star}$$

In certain cases, the notion can be captured syntactically by a list of similar *rule permuations*. Equivalence becomes a syntactic notion.

Equivalence problem

The equivalence problem of a logic $\mathbf{L}$ is the decision problem:

"*Given two* $\mathbf{L}$ *proofs* π *and* ν, *are they equivalent?*"

Equivalence and commutative conversions

Curry-Howard: proofs as programs, `cut`-elimination as evaluation.

Equivalence and commutative conversions

Curry-Howard: proofs as programs, `cut`-elimination as evaluation.

Equivalent but syntaxically different proofs/terms are an issue: need to switch between equivalent representation to perform a reduction step.

Equivalence and commutative conversions

Curry-Howard: proofs as programs, `cut`-elimination as evaluation.

Equivalent but syntactically different proofs/terms are an issue: need to switch between equivalent representation to perform a reduction step.

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, C \vdash D} \qquad \dfrac{\langle\mu\rangle}{B, C \vdash D}}{A \oplus B, C \vdash D}\,\oplus^{\star}}{A \oplus B \vdash C \multimap D}\,\multimap \qquad \dfrac{\dfrac{\langle\nu\rangle}{\vdash A}}{\vdash A \oplus B}\,\oplus}{\vdash C \multimap D}\,\texttt{cut}$$

(no elimination possible)

Equivalence and commutative conversions

Curry-Howard: proofs as programs, `cut`-elimination as evaluation.

Equivalent but syntaxically different proofs/terms are an issue: need to switch between equivalent representation to perform a reduction step.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\langle\pi\rangle\\ A,C\vdash D\end{matrix}\qquad\begin{matrix}\langle\mu\rangle\\ B,C\vdash D\end{matrix}}{A\oplus B,C\vdash D}\,\oplus^\star}{A\oplus B\vdash C\multimap D}\,\multimap\qquad\dfrac{\begin{matrix}\langle\nu\rangle\\ \vdash A\end{matrix}}{\vdash A\oplus B}\,\oplus}{\vdash C\multimap D}\,\texttt{cut}$$

(no elimination possible)

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\langle\pi\rangle\\ A,C\vdash D\end{matrix}}{A\vdash C\multimap D}\,\multimap\qquad\dfrac{\begin{matrix}\langle\mu\rangle\\ B,C\vdash D\end{matrix}}{B\vdash C\multimap D}\,\multimap}{A\oplus B\vdash C\multimap D}\,\oplus^\star\qquad\dfrac{\begin{matrix}\langle\nu\rangle\\ \vdash A\end{matrix}}{\vdash A\oplus B}\,\oplus}{\vdash C\multimap D}\,\texttt{cut}$$

($\oplus/\oplus^\star$ elimination)

Down with commutations: proofnets

Commutative conversion complexify the study of `cut`-elimination, one has to work *modulo* rule permuation.

By construction (associativity of composition in a categorical model) the `cut` rule commutes with itself.

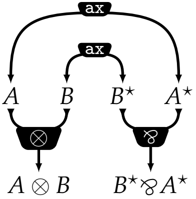
Down with commutations: proofnets

Commutative conversion complexify the study of `cut`-elimination, one has to work *modulo* rule permuation.

By construction (associativity of composition in a categorical model) the `cut` rule commutes with itself.

Proofnets (Girard): an approach to this issue, with combinatorial objects (graph structures) canonically representing equivalence classes of proofs.

$\hookrightarrow$ `cut`-elimination free of commutative conversions, better theoretical understanding of the logic studied.



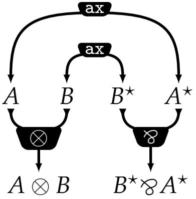
Down with commutations: proofnets

Commutative conversion complexify the study of cut-elimination, one has to work *modulo* rule permuation.

By construction (associativity of composition in a categorical model) the cut rule commutes with itself.

Proofnets (Girard): an approach to this issue, with combinatorial objects (graph structures) canonically representing equivalence classes of proofs.

$\hookrightarrow$ cut-elimination free of commutative conversions, better theoretical understanding of the logic studied.



But is this goal always achievable?

No proofnets for MLL

But is this goal always achievable?

For some time the **MLL** (with units) case remained open: notions of proofnets with jumps, complex "rewiring equivalence" (not canonical).

No proofnets for MLL

But is this goal always achievable?

For some time the **MLL** (with units) case remained open: notions of proofnets with jumps, complex "rewiring equivalence" (not canonical).

Heijltjes and Houston recently settled the question (negatively).

No proofnets for MLL

But is this goal always achievable?

For some time the **MLL** (with units) case remained open: notions of proofnets with jumps, complex "rewiring equivalence" (not canonical).

Heijltjes and Houston recently settled the question (negatively).

- Proofnet entail solving the equivalence problem: convert proofs to proofnets, then check for *equality*.

No proofnets for MLL

But is this goal always achievable?

For some time the **MLL** (with units) case remained open: notions of proofnets with jumps, complex "rewiring equivalence" (not canonical).

Heijltjes and Houston recently settled the question (negatively).

- Proofnet entail solving the equivalence problem: convert proofs to proofnets, then check for *equality*.
- They studied the **MLL** equivalence problem and showed:

Theorem (Heijltjes and Houston, 2014)

The equivalence problem of MLL is Pspace-complete.

No proofnets for MLL

But is this goal always achievable?

For some time the **MLL** (with units) case remained open: notions of proofnets with jumps, complex "rewiring equivalence" (not canonical).

Heijltjes and Houston recently settled the question (negatively).

- Proofnet entail solving the equivalence problem: convert proofs to proofnets, then check for *equality*.
- They studied the **MLL** equivalence problem and showed:

Theorem (Heijltjes and Houston, 2014)

The equivalence problem of MLL is Pspace-complete.

$\hookrightarrow$ no notion of proofnet for **MLL** both canonical and low-complexity.

Multiplicative-additive logic

Multiplicative-additive logic

A basic fragment of (intutionnistic) linear logic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{l} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{r} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus^\star$$

in addition to linear implication $\multimap$.

Multiplicative-additive logic

A basic fragment of (intutionnistic) linear logic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{l} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{r} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus^{\star}$$

in addition to linear implication $\multimap$.

Similarly to **MLL**, no 100% satisfying notion of proofnet so far.

Multiplicative-additive logic

A basic fragment of (intutionnistic) linear logic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{l} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{r} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus^\star$$

in addition to linear implication $\multimap$.

Similarly to **MLL**, no 100% satisfying notion of proofnet so far.
Source of the difficulty:

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, E \vdash C} \qquad \dfrac{\langle\mu\rangle}{B, E \vdash C}}{A \oplus B, E \vdash C}\ \oplus^\star \qquad \dfrac{\langle\nu\rangle}{\vdash D}}{A \oplus B, D \multimap E \vdash C}\ \multimap^\star \qquad \sim$$

Multiplicative-additive logic

A basic fragment of (intutionnistic) linear logic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{l} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{r} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus^\star$$

in addition to linear implication $\multimap$.

Similarly to **MLL**, no 100% satisfying notion of proofnet so far.

Source of the difficulty:

$$\dfrac{\dfrac{\genfrac{}{}{0pt}{}{\langle\pi\rangle}{A, E \vdash C} \qquad \genfrac{}{}{0pt}{}{\langle\mu\rangle}{B, E \vdash C}}{A \oplus B, E \vdash C}\ \oplus^\star \qquad \genfrac{}{}{0pt}{}{\langle\nu\rangle}{\vdash D}}{A \oplus B, D \multimap E \vdash C}\ \multimap^\star \quad \sim \quad \dfrac{\dfrac{\genfrac{}{}{0pt}{}{\langle\pi\rangle}{A, E \vdash C} \qquad \genfrac{}{}{0pt}{}{\langle\nu\rangle}{\vdash D}}{A, D \multimap E \vdash C} \qquad \dfrac{\genfrac{}{}{0pt}{}{\langle\mu\rangle}{B \vdash C} \qquad \genfrac{}{}{0pt}{}{\langle\nu\rangle}{\vdash D}}{B, D \multimap E \vdash C}\ \multimap^\star}{A \oplus B, D \multimap E \vdash C}\ \oplus^\star$$

Multiplicative-additive logic

A basic fragment of (intutionnistic) linear logic with

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{l} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\mathtt{r} \qquad \frac{\Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C}\ \oplus^\star$$

in addition to linear implication $\multimap$.

Similarly to **MLL**, no 100% satisfying notion of proofnet so far.

Source of the difficulty:

$$\dfrac{\dfrac{\overset{\langle\pi\rangle}{A, E \vdash C} \qquad \overset{\langle\mu\rangle}{B, E \vdash C}}{A \oplus B, E \vdash C}\ \oplus^\star \qquad \overset{\langle\nu\rangle}{\vdash D}}{A \oplus B, D \multimap E \vdash C}\ \multimap^\star \quad \sim \quad \dfrac{\dfrac{\overset{\langle\pi\rangle}{A, E \vdash C} \qquad \overset{\langle\nu\rangle}{\vdash D}}{A, D \multimap E \vdash C} \qquad \dfrac{\overset{\langle\mu\rangle}{B \vdash C} \qquad \overset{\langle\nu\rangle}{\vdash D}}{B, D \multimap E \vdash C}\ \multimap^\star}{A \oplus B, D \multimap E \vdash C}\ \oplus^\star$$

(equates two objects of wildly different sizes)

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^{\star}$.

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

- **Slice nets (Hughes-van Glabbeek):** represent proofs as list of "slices" (list of atoms paired by axiom rules), different variants of the proof.

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

- **Slice nets (Hughes-van Glabbeek):** represent proofs as list of “slices” (list of atoms paired by axiom rules), different variants of the proof. For instance a proof of $\alpha \oplus \beta \vdash \alpha \oplus \beta$ will have two slices:

$\alpha \oplus \beta \vdash \alpha \oplus \beta$ and $\alpha \oplus \beta \vdash \alpha \oplus \beta$

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

- **Slice nets (Hughes-van Glabbeek):** represent proofs as list of "slices" (list of atoms paired by axiom rules), different variants of the proof. For instance a proof of $\alpha \oplus \beta \vdash \alpha \oplus \beta$ will have two slices:

  $\alpha \oplus \beta \vdash \alpha \oplus \beta$ and $\alpha \oplus \beta \vdash \alpha \oplus \beta$

  $\hookrightarrow$ canonical but exponential blowup. *(on $\otimes$ rules, the number of slice is multiplied)*

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

- **Slice nets (Hughes-van Glabbeek):** represent proofs as list of "slices" (list of atoms paired by axiom rules), different variants of the proof. For instance a proof of $\alpha \oplus \beta \vdash \alpha \oplus \beta$ will have two slices:

  $\alpha \oplus \beta \vdash \alpha \oplus \beta$ and $\alpha \oplus \beta \vdash \alpha \oplus \beta$

  $\hookrightarrow$ canonical but exponential blowup. *(on $\otimes$ rules, the number of slice is multiplied)*

- **Conflict nets (Heijltjes-Houston):** mechanism to remember which axiom links cannot be present at the same time.

Multiplicative-additive proofnets

Some approaches to multiplicative-additive proofnets:

- **Monomial nets (Girard, Laurent-Maielli):** attach Boolean weights to the edges of a graph, telling its presence/absence depending on $\oplus^\star$.

  $\hookrightarrow$ partially **Ptime** operations, not canonical.

- **Slice nets (Hughes-van Glabbeek):** represent proofs as list of "slices" (list of atoms paired by axiom rules), different variants of the proof. For instance a proof of $\alpha \oplus \beta \vdash \alpha \oplus \beta$ will have two slices:

  $\alpha \oplus \beta \vdash \alpha \oplus \beta$ and $\alpha \oplus \beta \vdash \alpha \oplus \beta$

  $\hookrightarrow$ canonical but exponential blowup. *(on $\otimes$ rules, the number of slice is multiplied)*

- **Conflict nets (Heijltjes-Houston):** mechanism to remember which axiom links cannot be present at the same time.

  $\hookrightarrow$ **Ptime** operations, not canonical. *(but better quotient than monomial nets)*

Complexity of multiplicative-additive proof equivalence

Leads us to our main story:

- What is the complexity of multiplicative-additive proof equivalence?
- Do we get an impossibility result as in the **MLL** case?

Complexity of multiplicative-additive proof equivalence

Leads us to our main story:

- What is the complexity of multiplicative-additive proof equivalence?
- Do we get an impossibility result as in the **MLL** case?

  Heijltjes-Hughes argue that we cannot have ***both*** *canonical and* **Ptime**

Complexity of multiplicative-additive proof equivalence

Leads us to our main story:

- What is the complexity of multiplicative-additive proof equivalence?
- Do we get an impossibility result as in the **MLL** case?

*Heijltjes-Hughes argue that we cannot have **both** canonical and* **Ptime**

What was known so far:

- Decidable (Cockett and Pastro) *via* a term calculus with decision procedure for commutative conversions.

Complexity of multiplicative-additive proof equivalence

Leads us to our main story:

- What is the complexity of multiplicative-additive proof equivalence?
- Do we get an impossibility result as in the **MLL** case?

Heijltjes-Hughes argue that we cannot have ***both*** *canonical and* **Ptime**

What was known so far:

- Decidable (Cockett and Pastro) *via* a term calculus with decision procedure for commutative conversions.
- Slice nets imply **Exptime** equivalence.

Complexity of multiplicative-additive proof equivalence

Leads us to our main story:

- What is the complexity of multiplicative-additive proof equivalence?
- Do we get an impossibility result as in the **MLL** case?

*Heijltjes-Hughes argue that we cannot have **both** canonical and* **Ptime**

What was known so far:

- Decidable (Cockett and Pastro) *via* a term calculus with decision procedure for commutative conversions.
- Slice nets imply **Exptime** equivalence.
- Subsumed by `cut`-elimination equivalence, showed **coNP**-complete. (Mairson-Terui)

Binary Decision Slicing

Binary Decision Trees

First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

Binary Decision Trees

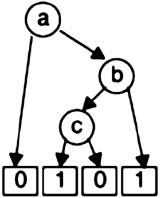
First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

A closer look reveals that they involve only a specific type of formulas, similar to binary decision trees (BDT).

Binary Decision Trees

First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

A closer look reveals that they involve only a specific type of formulas, similar to binary decision trees (BDT).



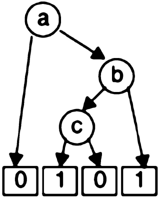
Definition

A BDT **is a binary tree with nodes labeled by Boolean variables and leaves labelled by 1 and 0.**

Binary Decision Trees

First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

A closer look reveals that they involve only a specific type of formulas, similar to binary decision trees (BDT).



Definition

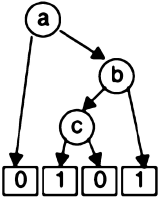
A BDT **is a binary tree with nodes labeled by Boolean variables and leaves labelled by $\mathbf{1}$ and $\mathbf{0}$.**

(notation $a \rhd \cdot \, [\!] \cdot$*)*

Binary Decision Trees

First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

A closer look reveals that they involve only a specific type of formulas, similar to binary decision trees (BDT).



Definition

A BDT **is a binary tree with nodes labeled by Boolean variables and leaves labelled by 1 and 0.**

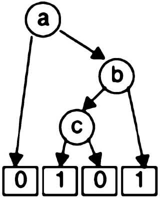
(notation $a \triangleright \cdot \,[\!]\, \cdot$*)*

Two BDT are *equivalent* ($\phi \sim \psi$) if they represent the same boolean function.

Binary Decision Trees

First (wrong) intuition: equivalence of monomial nets amounts to equivalence of Boolean formulas (**coNP**-complete).

A closer look reveals that they involve only a specific type of formulas, similar to binary decision trees (BDT).



Definition

A BDT **is a binary tree with nodes labeled by Boolean variables and leaves labelled by $\mathbf{1}$ and $\mathbf{0}$.**

(notation $a \rhd \cdot \, [\!] \, \cdot$*)*

Two BDT are *equivalent* ($\phi \sim \psi$) if they represent the same boolean function. (*e.g.* $a \rhd \mathbf{1} \, [\!] \, \mathbf{1}$ is equivalent to $\mathbf{1}$).

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}\,/\,\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}$ / $\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \dfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\ \oplus^\star_x$$

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right $(\mathbf{0}\,/\,\mathbf{1})$ branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \cfrac{\cfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \cfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\,\oplus^\star_x \qquad \mapsto \qquad \alpha \oplus_x \beta \vdash \alpha \oplus \beta$$

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right (**0** / **1**) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \dfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\ \oplus^\star_x \qquad \mapsto \qquad \alpha \oplus_x \beta \vdash \alpha \oplus \beta$$

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}$ / $\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \dfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\,\oplus^\star_x \qquad \mapsto \qquad \overset{x \rhd \mathbf{1} \,[]\, \mathbf{0}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}$$

Binary Decision Slicing

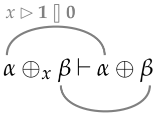
Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}$ / $\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \dfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\,\oplus^\star_x \qquad \mapsto$$



Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}\,/\,\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \cfrac{\cfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \cfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\ \oplus^\star_x \qquad \mapsto \qquad \overset{x \rhd \mathbf{1} \,[\,]\, \mathbf{0}}{\underset{x \rhd \mathbf{0} \,[\,]\, \mathbf{1}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}}$$

Binary Decision Slicing

Intermediate notion between monomial and slice nets:

To a proof π of $\Gamma \vdash A$ we associate a function $\mathcal{B}_\pi$ that maps each pair of atoms $[u,v]$ of $\Gamma \vdash A$ to a BDT $\mathcal{B}_\pi[u,v]$.

Main idea: label each $\oplus$ connectives in Γ with a boolean variable. The BDT associated to the pair $[u,v]$ tells the presence of an $[u,v]$ axiom link depending on which left/right ($\mathbf{0}\,/\,\mathbf{1}$) branch of the $\oplus^\star$ rule we are sitting.

Example:

$$\pi = \dfrac{\dfrac{\alpha \vdash \alpha}{\alpha \vdash \alpha \oplus \beta} \qquad \dfrac{\beta \vdash \beta}{\beta \vdash \alpha \oplus \beta}}{\alpha \oplus_x \beta \vdash \alpha \oplus \beta}\ \oplus^\star_x \qquad \mapsto \qquad \overset{x \,\triangleright\, \mathbf{1} \,[\!]\, \mathbf{0}}{} \; \alpha \oplus_x \beta \vdash \alpha \oplus \beta \; \underset{x \,\triangleright\, \mathbf{0} \,[\!]\, \mathbf{1}}{}$$

Definition (equivalence)

Define $\mathcal{B} \sim \mathcal{B}'$ as pointwise equivalence.
(for each pair of atoms $[\alpha,\beta]$, $\mathcal{B}[\alpha,\beta] \sim \mathcal{B}'[\alpha,\beta]$)

Binary Decision slicing and proof equivalence

The usefulness of this notion comes from:

Theorem

Equivalence of slicing captures proof equivalence: $\pi \sim \mu$ ***iff*** $\mathcal{B}_\pi \sim \mathcal{B}_\mu$ **.**

Binary Decision slicing and proof equivalence

The usefulness of this notion comes from:

Theorem

Equivalence of slicing captures proof equivalence: $\pi \sim \mu$ ***iff*** $\mathcal{B}_\pi \sim \mathcal{B}_\mu$ **.**

An idea of the proof: for any valuation v of the variables define a slice

$$v(\mathcal{B}) = \{\ [\alpha,\beta] \mid v(\mathcal{B}[\alpha,\beta]) = \mathbf{1}\ \}$$

then show that the set of $v(\mathcal{B})$ slices is the same as the set of slices in the Hughes-van Glabbeek proofnets.

Binary Decision slicing and proof equivalence

The usefulness of this notion comes from:

Theorem

Equivalence of slicing captures proof equivalence: $\pi \sim \mu$ ***iff*** $\mathcal{B}_\pi \sim \mathcal{B}_\mu$**.**

An idea of the proof: for any valuation v of the variables define a slice

$$v(\mathcal{B}) = \{ \, [\alpha, \beta] \mid v(\mathcal{B}[\alpha, \beta]) = \mathbf{1} \, \}$$

then show that the set of $v(\mathcal{B})$ slices is the same as the set of slices in the Hughes-van Glabbeek proofnets.

$\hookrightarrow$ multiplicative-additive proof equivalence reduces to BDT equivalence.

Complexity

Equivalence of BDT

Theorem

The equivalence problem of BDT is in Logspace.
(simpler than full Boolean formulas)

Equivalence of BDT

Theorem

The equivalence problem of BDT is in Logspace.
(simpler than full Boolean formulas)

Idea of the proof:

Define compatible leafs of two BDT: they do not need opposite valuation of a variable to be reached.

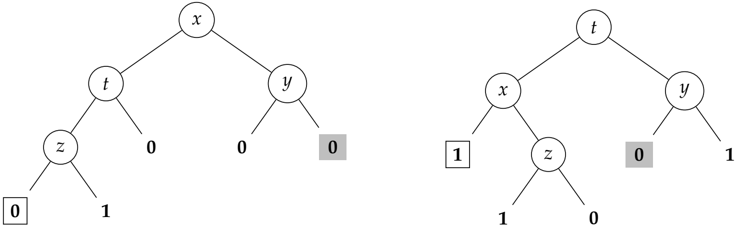
Equivalence of BDT

Theorem

The equivalence problem of BDT is in Logspace.
(simpler than full Boolean formulas)

Idea of the proof:

Define compatible leafs of two BDT: they do not need opposite valuation of a variable to be reached. Example: compatible and incompatible leaves.



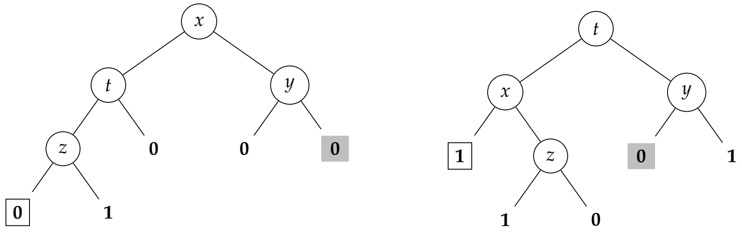
Equivalence of BDT

Theorem

The equivalence problem of BDT is in Logspace.
(simpler than full Boolean formulas)

Idea of the proof:

Define compatible leafs of two BDT: they do not need opposite valuation of a variable to be reached. Example: compatible and incompatible leaves.



Two BDT are **not** equivalent *iff* they have compatible leafs holding opposite **0**/**1** values. Easily checked in **Logspace**.

Multiplicative-additive equivalence is in Logspace

Theorem

The equivalence problem of BDT **is in Logspace.**

Multiplicative-additive equivalence is in Logspace

Theorem

The equivalence problem of BDT **is in Logspace.**

The BDT slicing of a multiplicative-additive proof can be computed in logarithmic space, therefore we get

Theorem

Multiplicative-additive equivalence is in Logspace.

Multiplicative-additive equivalence is in Logspace

Theorem

The equivalence problem of BDT **is in Logspace.**

The BDT slicing of a multiplicative-additive proof can be computed in logarithmic space, therefore we get

Theorem

Multiplicative-additive equivalence is in Logspace.

The equivalence problem is low-complexity (in contrast with **MLL***) and does not yield an impossibility result for proofnets.*

Multiplicative-additive equivalence is in Logspace

Theorem

The equivalence problem of BDT **is in Logspace.**

The BDT slicing of a multiplicative-additive proof can be computed in logarithmic space, therefore we get

Theorem

Multiplicative-additive equivalence is in Logspace.

The equivalence problem is low-complexity (in contrast with **MLL***) and does not yield an impossibility result for proofnets. (⚠ does not solve the problem positively either, proofnets for this fragment could be impossible for other reasons)*

Hardness

Equivalence for multiplicative-additive linear logic is in **Logspace**. But is this result the best possible?

In other words: is the problem **Logspace**-hard?

Hardness

Equivalence for multiplicative-additive linear logic is in **Logspace**. But is this result the best possible?

In other words: is the problem **Logspace**-hard?

It turns out that exchange and $\otimes$ (or left $\multimap$) are enough to encode permutation problems, order problems *etc.* and these are **Logspace**-hard.

MLL$^-$ proof equivalence is **Logspace**-hard.

Hardness

Equivalence for multiplicative-additive linear logic is in **Logspace**. But is this result the best possible?

In other words: is the problem **Logspace**-hard?

It turns out that exchange and $\otimes$ (or left $\multimap$) are enough to encode permutation problems, order problems *etc.* and these are **Logspace**-hard.

MLL$^-$ proof equivalence is **Logspace**-hard.

Because **MLL$^-$** is a subsystem of multiplicative-additive linear logic, we get

Theorem

Proof equivalence of multiplicative-additive linear logic is Logspace-complete.

Conclusion:

- Multiplicative-additive equivalence problem is **Logspace**-complete

Conclusion:

- Multiplicative-additive equivalence problem is **Logspace**-complete
- Limitation result?

Conclusion:

- Multiplicative-additive equivalence problem is **Logspace**-complete
- Limitation result?
- Proofnets with a logspace equivalence, but non canonical?

Conclusion:

- Multiplicative-additive equivalence problem is **Logspace**-complete
- Limitation result?
- Proofnets with a logspace equivalence, but non canonical?

... Thank you for your attention !