MALL Proof Equivalence is Logspace-Complete, via Binary Decision Diagrams

Marc Bagnol

Department of Mathematics and Statistics, University of Ottawa, Canada

Abstract

Proof equivalence in a logic is the problem of deciding whether two proofs are equivalent *modulo* a set of permutation of rules that reflects the commutative conversions of its cut-elimination procedure. As such, it is related to the question of proofnets: finding canonical representatives of equivalence classes of proofs that have good computational properties. It can also be seen as the word problem for the notion of free category corresponding to the logic.

It has been recently shown that proof equivalence in MLL (the multiplicative with units fragment of linear logic) is Pspace-complete, which rules out any low-complexity notion of proofnet for this particular logic.

Since it is another fragment of linear logic for which attempts to define a fully satisfactory low-complexity notion of proofnet have not been successful so far, we study proof equivalence in MALL$^-$ (multiplicative-additive without units fragment of linear logic) and discover a situation that is totally different from the MLL case. Indeed, we show that proof equivalence in MALL$^-$ corresponds (under AC_0 reductions) to equivalence of binary decision diagrams, a data structure widely used to represent and analyze Boolean functions efficiently.

We show these two equivalent problems to be Logspace-complete. If this technically leaves open the possibility for a complete solution to the question of proofnets for MALL$^-$, the established relation with binary decision diagrams actually suggests a negative solution to this problem.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic

Keywords and phrases linear logic, proof equivalence, additive connectives, proofnets, binary decision diagrams, logarithmic space, AC_0 reductions

Digital Object Identifier 10.4230/LIPIcs.TLCA.2015.60

1 Introduction

Proofnets: from commutative conversions to canonicity

From the perspective of the Curry-Howard (or propositions-as-types) correspondence [5], a proof of $A \Rightarrow B$ in a logic enjoying a `cut`-elimination procedure can be seen as a program that inputs (through the `cut` rule) a proof of A and outputs a `cut`-free proof of B.

Coming from this dynamic point of view, linear logic [6] makes apparent the distinction between data that can or cannot be copied/erased via its exponential modalities and retains the symmetry of classical logic: the linear negation $(\cdot)^\star$ is an involutive operation. The study of `cut`-elimination is easier in this setting thanks to the linearity constraint. However, in its sequent calculus presentation, the `cut`-elimination procedure of linear logic still suffers from the common flaw of these type of calculi: commutative conversions.

13th International Conference on Typed Lambda Calculi and Applications (TLCA'15).
Editor: Thorsten Altenkirch; pp. 60–75

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A^\star, B^\star, C, D, \Gamma}}{A^\star \parr B^\star, C, D, \Gamma}\parr}{A^\star \parr B^\star, C \parr D, \Gamma}\parr \qquad \dfrac{\langle\mu\rangle \quad \langle\nu\rangle}{A \quad B}\otimes \qquad \dfrac{}{C \parr D, \Gamma}\text{cut} \quad \to \quad \dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A^\star, B^\star, C, D, \Gamma}}{A^\star \parr B^\star, C, D, \Gamma}\parr \qquad \dfrac{\langle\mu\rangle \quad \langle\nu\rangle}{A \quad B}\otimes}{C, D, \Gamma}\text{cut}}{C \parr D, \Gamma}\parr$$

In the above reduction, one of the two formulas related by the `cut` rule is introduced deeper in the proof, making it impossible to perform an actual elimination step right away: one needs first to *permute* the rules in order to be able to go on.

This type of step is called a *commutative conversion* and their presence complexify a lot the study of the `cut`-elimination procedure, as one needs to work *modulo* an equivalence relation on proofs that is not orientable into a rewriting procedure in an obvious way: there are for instance situations of the form

$$\dfrac{\dfrac{\langle\pi_1\rangle \quad \langle\pi_2\rangle}{A^\star, B^\star, \Gamma \qquad A}\text{cut}}{\vdash B^\star, \Gamma} \qquad \dfrac{\langle\pi_3\rangle}{B}\qquad \dfrac{}{\vdash \Gamma}\text{cut} \quad \leftrightarrow \quad \dfrac{\dfrac{\langle\pi_1\rangle \quad \langle\pi_3\rangle}{A^\star, B^\star, \Gamma \qquad B}\text{cut}}{\vdash A^\star, \Gamma} \qquad \dfrac{\langle\pi_2\rangle}{A}\qquad \dfrac{}{\vdash \Gamma}\text{cut}$$

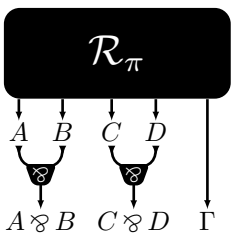
where it is not possible to favor one side of the equivalence without further non-local knowledge of the proof. The point here is that, as a language for describing proofs, sequent calculus is somewhat *too explicit*. For instance, the fact that the two proofs

$$\dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, B, C, D, \Gamma}}{A \parr B, C, D, \Gamma}\parr}{A \parr B, C \parr D, \Gamma}\parr \qquad \text{and} \qquad \dfrac{\dfrac{\dfrac{\langle\pi\rangle}{A, B, C, D, \Gamma}}{A, B, C \parr D, \Gamma}\parr}{A \parr B, C \parr D, \Gamma}\parr$$

are different objects from the point of view of sequent calculus generates the first commutative conversion we saw above.

A possible solution to this issue is to look for more intrinsic description of proofs, to find a language that is more *synthetic*; if possible to the point where we have no commutative conversions to perform anymore.

Introduced at the same time as linear logic, the theory of *proofnets* [6, 7] partially addresses this issue. The basic idea is to describe proofs as graphs rather than trees, where application of logical rules become local graph construction, thus erasing some inessential sequential informations. Indeed, the two proofs above would translate into the same proofnet:



(where $\mathcal{R}_\pi$ is the proofnet translation of the rest of the proof) and the corresponding commutative conversion disappears.

For the multiplicative without units fragment of linear logic (MLL$^-$), proofnets yield an entirely satisfactory solution to the problem, and constitute a low-complexity canonical representation of proofs based on local operations on graphs.

By canonical, we mean here that two proofs are equivalent *modulo* the permutations of rules induced by the commutative conversions if and only if they have the same proofnet

translation. From a categorical perspective, this means that proofnets constitute a syntactical presentation of the free semi-$*$-autonomous category and a solution to the associated word problem [10].

Contrastingly, the linear logic community has struggled to extend the notion of proofnets to wider fragment: even the question of MLL (that is, MLL$^-$ plus the multiplicative units) could not find a satisfactory answer. A recent result [9] helps to understand this situation: proof equivalence of MLL is actually a Pspace-complete problem. Hence, there is no hope for a satisfactory notion of low-complexity proofnet for this fragment[1].

In this article, we consider the same question, but in the case of MALL$^-$: the multiplicative-additive without units fragment of linear logic. Indeed, this fragment has so far also resisted the attempts to build a notion of proofnet that at the same time characterizes proof equivalence and has basic operations of tractable complexity: we have either canonical nets of exponential size [13] or tractable nets that are not canonical [7]. Therefore, it would have not been too surprising to have a similar result of completeness for some untractable complexity class. An obvious candidate in that respect would be coNP: as we will see, one of these two approaches to proofnets for MALL$^-$ is related to Boolean formulas, which equivalence problem is coNP-complete.

It turns out in the end that this is not the case: our investigation concludes that the equivalence problem in MALL$^-$ is Logspace-complete under AC_0 reductions. But maybe more importantly, we uncover in the course of the proof an unexpected connexion of this theoretical problem with a very practical issue: indeed we show that MALL$^-$ proofs are closely related to binary decision diagrams.

Binary decision diagrams

The problem of the representation of Boolean functions is of central importance in circuit design and has a large range of practical applications. Over the years, binary decision diagrams (BDD) [2] became the most widely used data structure to treat this question.

Roughly speaking, a BDD is a binary tree with nodes labeled by Boolean variables and leaves labeled by values 0 and 1. Such a tree represents a Boolean function in the sense that once an assignment of the variable is chosen, then following the left or right path at each node according to the value 0 or 1 chosen for its variable eventually leads to a leave, which is the output of the function.

This representation has many advantages which justify its popularity [15]: most basic operations (negation and other logical connectives) on BDD can be implemented efficiently, in many practical cases the size of the BDD representing a Boolean function remains compact (thanks to the possibility to have shared subtrees) and once a variable ordering is chosen they enjoy a notion of normal form.

In this article, we consider both BDD and ordered BDD with no sharing of subtrees and write them as special kinds of Boolean functions by introducing an `IfThenElse` constructor. However, when manipulating them from a complexity point of view we will keep the binary tree presentation in mind.

[1] Of course, this applies only to the standard formulation of units: the equivalence problem for any notion of multiplicative units enjoying less permutations of rules could potentially still be tractable via proofnets: see for instance the work of S. Guerrini and A. Masini [8] and D. Hughes [11].

AC_0 reductions

To show that a problem is complete for some complexity class C, one needs to specify the notion of reduction functions considered, and of course this needs to be a class of functions supposed to be smaller than C itself (indeed *any* problem in C is complete under C reductions).

A standard notion of reduction for the class Logspace is (uniform) AC_0 reduction [3], formally defined in terms of uniform circuits of fixed depth and unbounded fan-in. We will not be getting in the details about this complexity class and, as we will consider only graph transformations, we will rely on the following intuitive principle: if a graph transformation locally replaces each vertex by a bounded number of vertices and the replacement depends *only* on the vertex considered and eventually its direct neighbors, then the transformation is in AC_0. Typical examples of such a transformation are certain simple cases of so-called "gadget" reductions used in complexity theory to prove hardness results.

Outline of the paper

Section 2 covers some background material on MALL$^-$ and notions of proofnet for this fragment: monomial proofnets and the associated vocabulary for Boolean formulas and BDD in Section 2.1 and the notion of slicing in Section 2.2. Then, we introduce in Section 2.3 an intermediary notion of proof representation that will help us to relate proofs in MALL$^-$ and BDD. In Section 3, we prove that proof equivalence in MALL$^-$ and equivalence of BDD relate to each other through AC_0 reductions and that they are both Logspace-complete.

2 Proof equivalence in MALL$^-$

▶ **Notation 1.** *The formulas of* MALL$^-$ *are built inductively from atoms which we write* $\alpha, \beta, \gamma, \dots$ *their duals* $\alpha^\star, \beta^\star, \gamma^\star, \dots$ *and the binary connectives* $\parr, \otimes, \&_x, \oplus$ (we consider that the $\&$ connectives carry a label x to simplify some reasonings, but we will omit it when it is not relevant). *We write formulas as uppercase letters* $A, B, C, \dots$ *unless we want to specify they are atoms.* Sequents *are sequences of formulas, written as greek uppercase letters* $\Gamma, \Delta, \Lambda, \dots$ *such that all occurrences of the connective* $\&$ *in a sequent carry a different label. The concatenation of two sequents* Γ *and* Δ *is simply written* Γ, Δ.

Let us recall the rules of MALL$^-$[2]. We do not include the `cut` rule in our study, since in a static situation (we are not looking at the `cut`-elimination procedure of MALL$^-$) it can always be encoded w.l.o.g. using the $\otimes$ rule.

$$\alpha, \alpha^\star \qquad \frac{\Gamma, A, B}{\Gamma, A \parr B}\parr \qquad \frac{\Gamma, A \qquad \Delta, B}{\Gamma, \Delta, A \otimes B}\otimes \qquad \frac{\Gamma, A}{\Gamma, A \oplus B}\oplus_1 \qquad \frac{\Gamma, B}{\Gamma, A \oplus B}\oplus_r \qquad \frac{\Gamma, A \qquad \Gamma, B}{\Gamma, A \,\&_x\, B}\&_x$$

(by convention, we leave the axiom rule implicit to lighten notations. Also, we will use the notation $\begin{matrix}\langle\pi\rangle\\ \Gamma\end{matrix}$ *for "the proof* π *of conclusion* Γ*".)*

▶ Remark 2. Any time we will look at a MALL$^-$ proof from a complexity perspective, we will consider they are represented as trees with nodes corresponding to rules, labeled by the connective introduced and the sequent that is the conclusion of the rule.

[2] We consider a η-expanded version of MALL$^-$, which simplifies proofs and definitions, but the extension of our results to a version with non-atomic axioms would be straightforward. Also, we work *modulo* the exchange rule.

Two MALL⁻ proofs π and ν are said to be *equivalent* (notation $\pi \sim \nu$) if one can pass from one to the other via permutations of rules [14]. We have an associated decision problem.

▶ **Definition 3** (MALL⁻**equ**). MALL⁻**equ** is the decision problem:

"Given two MALL⁻ *proofs π and ν with the same conclusion, do we have $\pi \sim \nu$?"*

We will not go through all the details about this syntactic way to define proof equivalence in MALL⁻. The reason for this is that we already have an available equivalent characterization in terms of *slicing* [14] which we review in Section 2.2. Instead, let us focus only on the most significant case.

$$\dfrac{\dfrac{\langle\pi\rangle \quad \langle\mu\rangle}{\Gamma, A, C \quad \Gamma, B, C}\,\&}{\Gamma, A \,\&\, B, C} \quad \langle\nu\rangle \;\; \Delta, D \;\otimes}{\Gamma, \Delta, A \,\&\, B, C \otimes D} \quad \sim \quad \dfrac{\dfrac{\langle\pi\rangle \quad \langle\nu\rangle}{\Gamma, A, C \quad \Delta, D}\,\otimes \qquad \dfrac{\langle\mu\rangle \quad \langle\nu\rangle}{\Gamma, B, C \quad \Delta, D}\,\otimes}{\Gamma, \Delta, A, C \otimes D \qquad \Gamma, \Delta, B, C \otimes D}\,\&}{\Gamma, \Delta, A \,\&\, B, C \otimes D}$$

In the above equivalence, the $\otimes$ rule gets lifted above the $\&$ rule. But doing so, notice that we created two copies of ν instead of one, therefore the size of the prooftree has grown. Iterating on this observation, it is not hard to build pairs of proofs that are equivalent, but one of which is exponentially bigger than the other. This is indeed where the difficulty of proof equivalence in MALL⁻ lies. As a matter of fact, this permutation of rules *alone* would be enough to build the encoding of the equivalence problem of binary decision diagrams presented in Section 3.1.

A way to attack proof equivalence in a logic, as we exposed in Section 1, is to try to setup a notion of proofnet for this logic. In the following, we will review the main two approaches to this idea in the case of MALL⁻: *monomial proofnets* by J.-Y. Girard [7, 16] and *slicing proofnets* by D. Hughes and R. van Glabbeek [13, 14]. We will then design an intermediate notion of BDD *slicing* that will be more suited to our needs.

2.1 Monomial proofnets

The first attempt in the direction of a notion of proofnet for MALL⁻ is due to J.-Y.Girard [7], followed by a version with a full `cut`-elimination procedure by O. Laurent and R. Maielli [16].

While proofnets for multiplicative linear logic without units were introduced along linear logic itself [6], extending the notion to the multiplicative-additive without units fragment proved to be a true challenge, mainly because of the *superposition* at work in the $\&$ rule.

Girard's idea was to represent the superposed "versions" of the proofnet by attributing a Boolean formula (called a *weight*) to each link, with one Boolean variable for each $\&$ connective in the conclusion Γ. To retrieve the version of the proofnet corresponding to some selection of the left/right branches of each $\&$, one then just needs to evaluate the Boolean formulas with the corresponding valuation of their variables.

This is the occasion to introduce the vocabulary to speak about Boolean formulas.

▶ **Definition 4** (Boolean formula). Given a finite set of variables $V = \{x_1, \ldots, x_n\}$, a *Boolean formula* over V is inductively defined from the elements of V; the constants 0 (*"false"*) and 1(*"true"*); the unary symbol $\bar{\cdot}$ (*"negation"*); the binary symbols $+$ and $.$ (*"sum/or/disjuction"* and *"product/and/conjunction"* respectively).

We consider a syntactic notion of *equality* of Boolean formulas: for instance $0.x \neq 0$. The real important notion, that we therefore state separately, is *equivalence*: the fact that if we replace the variables with actual values, we gets the same output.

▶ **Definition 5** (equivalence)**.** A *valuation* v of V is a choice of 0 or 1 for any element of V. A valuation induces a an *evaluation* function $v(\cdot)$ from Boolean formulas over V to $\{0,1\}$ in the obvious way. Two Boolean formulas ϕ and ψ over V are *equivalent* (notation $\phi \sim \psi$) when for any valuation v of V, we have $v(\phi) = v(\psi)$.

▶ **Definition 6** (monomial)**.** We write $\overline{V} = \{\overline{x_1}, \dots, \overline{x}_n\}$. A *monomial* over V is a Boolean formula of the form $y_1. \dots .y_k$ with $\{y_1, \dots, y_k\} \subseteq V \cup \overline{V}$.

Two monomials m and m' are in *conflict* if $m.m' \sim 0$.

▶ Remark 7. Two monomials are in conflict if and only if there is a variable x such that x appears in one of them and $\overline{x}$ appears in the other.

While monomials are a specific type of Boolean formula, the *binary decision diagrams* we are about to introduce are not defined directly as Boolean formulas. Of course, they relate to each other in an obvious way, but having a specific syntax for binary decision diagrams will prove more convenient to solve the problems we will be facing.

▶ **Definition 8** (BDD)**.** A *binary decision diagram* (BDD) is defined inductively as:

- The constants 0 and 1 are BDD
- If ϕ, ψ are BDD and X is either a variable, 0 or 1, `If` X `Then` ϕ `Else` ψ is a BDD
- If ϕ is a BDD and X is either a variable, 0 or 1, `DontCare` X `Then` ϕ is a BDD

Moreover, suppose we have an ordered set of variables $V = \{x_1, \dots, x_n\}$ with the convention that variables are listed in the reverse order: x_n is first, then x_{n-1}, etc. We define a subclass of BDD which we call *ordered binary decision diagrams over* V (°BDD/V) by restricting to the following inductive cases (we let $V' = \{x_1, \dots, x_{n-1}\}$)

- The constants 0 and 1 are °BDD/$\varnothing$
- If ϕ and ψ are °BDD/V', `If` x_n `Then` ϕ `Else` ψ is a °BDD/V
- If ϕ is a °BDD/V', `DontCare` x_n `Then` ϕ is a °BDD/V

The notions of valuation and equivalence are extended to BDD and °BDD the obvious way so that `DontCare` X `Then` $\phi \sim \phi$ and `If` X `Then` ϕ `Else` $\psi \sim X.\phi + \overline{X}.\psi$.

▶ Remark 9. Any time we will look at BDD and °BDD from a complexity perspective, we will consider they are represented as labeled trees. The cases of `If` 0, `DontCare` 1,... will be useful to obtain AC_0 reductions in Section 2.3 and Section 3.2, since erasing a whole subpart of a graph of which we do not know the address in advance is not something that is doable in this complexity class. The absence of sharing implied by the tree representation is also crucial to get low-complexity reductions.

▶ **Example 10.** The Boolean formula $x.y.\overline{z}$ is a monomial, while $x.y + z$ and $x.1$ are not.

The BDD `If` x_2 `Then` (`If` x_1 `Then` 0 `Else` 1) `Else` 1 (which is *not* a °BDD/$\{x_1, x_2\}$ by the way) is equivalent to the Boolean formula $x_2.\overline{x_1} + \overline{x_2}$: both evaluate to 1 only for the valuations $\{x_1 \mapsto 1, x_2 \mapsto 0\}$, $\{x_1 \mapsto 0, x_2 \mapsto 0\}$ and $\{x_1 \mapsto 0, x_2 \mapsto 1\}$. There exist an equivalent °BDD/$\{x_1, x_2\}$: `If` x_2 `Then` (`If` x_1 `Then` 0 `Else` 1) `Else` (`DontCare` x_1 `Then` 1).

▶ **Definition 11.** BDD**equ** is the following decision problem:

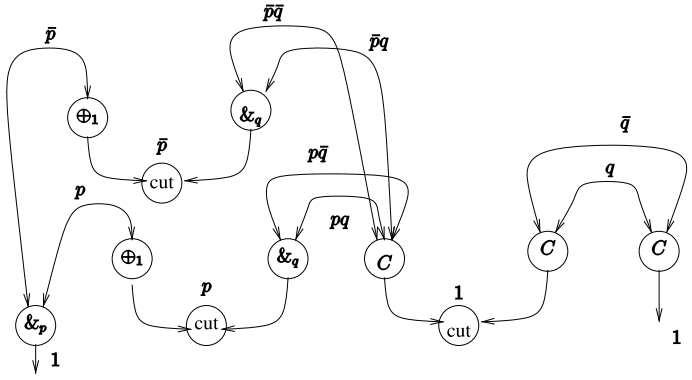
"given two BDD ϕ *and* ψ*, do we have* $\phi \sim \psi$*?"*

°BDD**equ** is the following decision problem:

"given two °BDD/V ϕ *and* ψ*, do we have* $\phi \sim \psi$*?"*

Girard's proofnets are called monomial because the only Boolean formulas that are allowed are monomials. This is, as of the state of the art, the only known way to have a notion of proofnet that enjoys a satisfying correctness criterion.

For our purposes, we do not need to get into the technical details of monomial proofnets. Still, let us end this section with an example of proofnet from the article by Laurent and Maielli, where the monomial weight of a link is pictured just above it:



2.2 Slicings and proof equivalence

The idea of slicing dates back to J.-Y. Girard's original article on proofnets for MALL⁻ [7], and was even present in the original article on linear logic [6]. It amounts to the natural point of view already evoked in Section 2.1, seeing the & rule as introducing superposed variants of the proof, which are eventually to be selected from in the course of `cut`-elimination. If we have two alternative *slices* for each & connective of a sequent Γ and all combinations of slices can be selected independently, we readily see that the global number of slices will be exponential in the number of & connectives in Γ.

This is indeed the major drawback of the representation of proofs as set of slices: the size of objects representing proofs may grow exponentially in the size of the original proofs. This of course impairs any fine-grained analysis in terms of complexity.

▶ **Definition 12** (slicing). Given a MALL⁻ sequent Γ, a *linking* of Γ is a subset

$$\big\{[\alpha_1, \alpha_1^\star], \ldots, [\alpha_n, \alpha_n^\star]\big\}$$

of the set of (unordered) pairs of occurrences of dual atoms in Γ.

Then, a *slicing* of Γ is a finite set of linkings of Γ.

To any MALL⁻ proof π, we associate a slicing $\mathcal{S}_\pi$ by induction:

- If $\pi = \alpha, \alpha^\star$ then $\mathcal{S}_\pi$ is the set containing only the linking $\big\{[\alpha, \alpha^\star]\big\}$
- If $\pi = \dfrac{\langle\mu\rangle}{\Gamma, A, B}\,{}^{\parr}$ then $\mathcal{S}_\pi = \mathcal{S}_\mu$, where we see atoms of $A \parr B$ as the corresponding atoms of A and B
- If $\pi = \dfrac{\langle\mu\rangle \quad \langle\nu\rangle}{\Gamma, A \quad \Delta, B}\,{}^{\otimes}$ then $\mathcal{S}_\pi = \{\, \lambda \cup \lambda' \mid \lambda \in \mathcal{S}_\mu\,,\ \lambda' \in \mathcal{S}_\nu \,\}$
- If $\pi = \dfrac{\langle\mu\rangle}{\Gamma, A}\,{}^{\oplus_1}$ then $\mathcal{S}_\pi = \mathcal{S}_\mu$, and likewise for $\oplus_{\mathtt{r}}$

- If $\pi = \dfrac{\begin{array}{cc}\langle\mu\rangle & \langle\nu\rangle \\ \Gamma, A & \Gamma, B\end{array}}{\Gamma, \Delta, A \,\&\, B}\,\&$ then $\mathcal{S}_\pi = \mathcal{S}_\mu \cup \mathcal{S}_\nu$

▶ Remark 13. In the $\otimes$ rule, it is clearly seen that the number of slices are multiplied. This is just what is needed in order to have a combinatorial explosion: for any n, a proof π_n of

$$\underbrace{\alpha^\star \otimes \cdots \otimes \alpha^\star}_{n \text{ times}}, \underbrace{\alpha \,\&\, \alpha, \ \ldots\ , \alpha \,\&\, \alpha}_{n \text{ times}}$$

obtained by combining with the $\otimes$ rule n copies of the proof

$$\frac{\alpha^\star, \alpha \qquad \alpha^\star, \alpha}{\alpha^\star, \alpha \,\&\, \alpha}\,\&$$

will be of linear size in n, but with a slicing $\mathcal{S}_n$ containing 2^n linkings.

Slicings (associated to a proof) correspond exactly to the notion of proofnets elaborated by D. Hughes and R. van Glabbeek [13]. While their study was mainly focused on the problems of finding a correctness criterion and designing a `cut`-elimination procedure for these, it also covers the proof equivalence problem. The proof that their notion of proofnet characterizes MALL⁻ proof equivalence can be found in an independent note [14].

▶ **Theorem 14** (slicing equivalence [14, Theorem 1]). *Let π and ν be two* MALL⁻ *proofs. We have that $\pi \sim \nu$ if and only if $\mathcal{S}_\pi = \mathcal{S}_\nu$.*

Let us also end this section with a graphical representation of an example of proofnet from the article of Hughes and van Glabbeek, encoding the proof on the left with three linkings $\lambda_1, \lambda_2, \lambda_3$:

$$\dfrac{\dfrac{\dfrac{P^\star, P}{P^\star \oplus Q^\star, P}\oplus_1}{(P^\star \oplus Q^\star) \oplus R^\star, P}\oplus_1 \quad \dfrac{\dfrac{Q^\star, Q}{P^\star \oplus Q^\star, Q}\oplus_\mathrm{r}}{(P^\star \oplus Q^\star) \oplus R^\star, P}\oplus_1}{(P^\star \oplus Q^\star) \oplus R^\star, P \,\&\, Q}\,\& \quad \dfrac{R^\star, R}{(P^\star \oplus Q^\star) \oplus R^\star, P}\oplus_\mathrm{r}}{(P^\star \oplus Q^\star) \oplus R^\star, (P \,\&\, Q) \,\&\, R}\,\&$$

λ_1 : $(P^\perp \oplus Q^\perp) \oplus R^\perp,\ P \,\&\, (Q \,\&\, R)$

λ_2 : $(P^\perp \oplus Q^\perp) \oplus R^\perp,\ P \,\&\, (Q \,\&\, R)$

λ_3 : $(P^\perp \oplus Q^\perp) \oplus R^\perp,\ P \,\&\, (Q \,\&\, R)$

2.3 BDD slicings

We finally introduce an intermediate notion of representation of proofs which will be a central tool in the next section. In a sense, it is a synthesis of monomial proofnets and slicings: acknowledging the fact that slicing makes the size of the representation explode, we rely on BDD to keep things more compact.

Of course, the canonicity property is lost. But this is exactly the point! Indeed, deciding whether two "BDD slicings" are equivalent is the reformulation of proof equivalence in MALL⁻ we rely on in the reductions between MALL⁻**equ** (Definition 3) and BDD**equ** (Definition 11).

▶ **Definition 15** (BDD slicing). Given a MALL⁻ sequent Γ, a BDD *slicing* of Γ is a function $\mathcal{B}$ that associates a BDD to every element $[\gamma, \gamma^\star]$ of the set of (unordered) pairs of occurrences of dual atoms in Γ.

We say that two BDD slicings M, N of the same Γ are *equivalent* (notation $M \sim N$) if for any pair $[\gamma, \gamma^\star]$, we have $M[\gamma, \gamma^\star] \sim N[\gamma, \gamma^\star]$ in the sense of Definition 5.

To any MALL⁻ proof π, we associate a BDD slicing $\mathcal{B}_\pi$ by induction:

- If $\pi = \alpha, \alpha^\star$ then $\mathcal{B}_\pi[\alpha, \alpha^\star] = 1$.

- If $\pi = \dfrac{\langle\mu\rangle}{\dfrac{\Gamma, A, B}{\Gamma, A \parr B}\parr}$ then $\mathcal{B}_\pi[\gamma, \gamma^\star] = \mathcal{B}_\mu[\gamma, \gamma^\star]$ where we see atoms of $A \parr B$ as the corresponding atoms of A and B
- If $\pi = \dfrac{\dfrac{\langle\mu\rangle}{\Gamma, A} \quad \dfrac{\langle\nu\rangle}{\Delta, B}}{\Gamma, \Delta, A \otimes B}\otimes$ then $\mathcal{B}_\pi[\gamma, \gamma^\star] = \begin{cases} \mathcal{B}_\mu[\gamma, \gamma^\star] & \text{if } \gamma, \gamma^\star \text{ are atoms of } \Gamma, A \\ \mathcal{B}_\nu[\gamma, \gamma^\star] & \text{if } \gamma, \gamma^\star \text{ are atoms of } \Delta, B \\ 0 & \text{otherwise}^3 \end{cases}$
- If $\pi = \dfrac{\dfrac{\langle\mu\rangle}{\Gamma, A}}{\Gamma, A \oplus B}\oplus_1$ then $\mathcal{B}_\pi[\gamma, \gamma^\star] = \begin{cases} \mathcal{B}_\mu[\gamma, \gamma^\star] & \text{if } \gamma, \gamma^\star \text{ are atoms of } \Gamma, A \\ 0 & \text{otherwise} \end{cases}$ and likewise for $\oplus_{\mathtt{r}}$.
- If $\pi = \dfrac{\dfrac{\langle\mu\rangle}{\Gamma, A} \quad \dfrac{\langle\nu\rangle}{\Gamma, B}}{\Gamma, \Delta, A \,\&_x B}\&_x$ then

$$\mathcal{B}_\pi[\gamma, \gamma^\star] = \begin{cases} \mathtt{If}\ x\ \mathtt{Then}\ \mathcal{B}_\mu[\gamma, \gamma^\star]\ \mathtt{Else}\ 0 & \text{if } \gamma \text{ or } \gamma^\star \text{ is an atom of } A \\ \mathtt{If}\ x\ \mathtt{Then}\ 0\ \mathtt{Else}\ \mathcal{B}_\nu[\gamma, \gamma^\star] & \text{if } \gamma \text{ or } \gamma^\star \text{ is an atom of } B \\ \mathtt{If}\ x\ \mathtt{Then}\ \mathcal{B}_\mu[\gamma, \gamma^\star]\ \mathtt{Else}\ \mathcal{B}_\nu[\gamma, \gamma^\star] & \text{otherwise} \end{cases}$$

▶ Remark 16. The BDD we obtain this way are actually of a specific type: they are usually called *read-once* BDD: from the root to any leave, one never crosses two `IfThenElse` nodes asking for the value of the same variable.

▶ **Example 17.** The weight of the pairs $[\alpha, \alpha^\star]$ and $[\delta, \delta^\star]$ in the BDD slicing of the proof

$$\pi = \dfrac{\dfrac{\dfrac{\alpha, \alpha^\star}{\alpha \oplus \beta, \alpha^\star}\oplus_1 \quad \dfrac{\beta, \beta^\star}{\alpha \oplus \beta, \beta^\star}\oplus_{\mathtt{r}}}{\alpha \oplus \beta, \alpha^\star \,\&_x \beta^\star}\&_x \quad \delta, \delta^\star}{\alpha \oplus \beta, (\alpha^\star \,\&_x \beta^\star) \otimes \delta, \delta^\star}\otimes$$

are $\mathcal{B}_\pi[\alpha, \alpha^\star] = \mathtt{If}\ x\ \mathtt{Then}\ 1\ \mathtt{Else}\ 0$ and $\mathcal{B}_\pi[\delta, \delta^\star] = 1$.

It is not hard to see that proof equivalence matches the equivalence of BDD slicings by relating them to slicings from the previous section.

▶ **Theorem 18** (BDD slicing equivalence)**.** *Let π and ν be two* MALL$^-$ *proofs. We have that $\pi \sim \nu$ if and only if $\mathcal{B}_\pi \sim \mathcal{B}_\nu$.*

Proof. We show in fact that $\mathcal{S}_\pi = \mathcal{S}_\nu$ if and only if $\mathcal{B}_\pi \sim \mathcal{B}_\nu$, with Theorem 14 in mind.

To a BDD slicing $\mathcal{B}$, we can associate a linking $v(\mathcal{B})$ for each valuation v of the variables occurring in $\mathcal{B}$ by setting $v(\mathcal{B}) = \{\ [\alpha, \alpha^\star] \mid v(\mathcal{B}[\alpha, \alpha^\star]) = 1\ \}$ and then a slicing $f(\mathcal{B}) = \{\ v(\mathcal{B}) \mid v \text{ valuation}\ \}$. By definition, it is clear that if $\mathcal{B}$ and $\mathcal{B}'$ involve the same variables and $\mathcal{B} \sim \mathcal{B}'$ then $f(\mathcal{B}) = f(\mathcal{B}')$.

Conversely, suppose $\mathcal{B}_\pi \not\sim \mathcal{B}_\nu$, so that there is a v such that $v(\mathcal{B}_\pi) \neq v(\mathcal{B}_\nu)$. To conclude that $f(\mathcal{B}_\pi) \neq f(\mathcal{B}_\nu)$, we must show that there is no other v' such that $v'(\mathcal{B}_\nu) = v(\mathcal{B}_\pi)$.

To do this, we can extend the notion of valuation to proofs: if v is a valuation of the labels x of the $\&_x$ in π, $v(\pi)$ is defined by keeping only the left or right branch of $\&_x$ according to the value of x. Now we can consider the set $v^{\mathtt{ax}}(\pi)$ of axiom rules in $v(\pi)$ and we can show by

[3] Remember we consider *occurrences* of atoms, and as the $\otimes$ rule splits the context into two independent parts that and no axiom rule can cross this splitting.

induction that $v^{\mathtt{ax}}(\pi)$ must contain at least one pair with one atom which is a subformula of the side of each $\&_x$ that has been kept. Therefore for any π and ν with the same conclusion, if $v \neq v'$ we have $v^{\mathtt{ax}}(\nu) \neq v'^{\mathtt{ax}}(\pi)$ no matter what. Then we can remark that $v^{\mathtt{ax}}(\pi)$ is just another name for $v(\mathcal{B}_\pi)$ so that in the end, there cannot be $v \neq v'$ such that $v'(\mathcal{B}_\nu) = v(\mathcal{B}_\pi)$.

Finally, an easy induction shows that $f(\mathcal{B}_\pi) = \mathcal{S}_\pi$ and therefore we are done. ◀

Also, a BDD equivalent to the BDD associated to a pair can be computed in AC_0.

▶ **Lemma 19** (computing BDD slicings). *For any* MALL⁻ *proof* π *and any pair* $[\gamma, \gamma^\star]$*, we can compute in* AC_0 *a* BDD ϕ *such that* $\phi \sim \mathcal{B}_\pi[\gamma, \gamma^\star]$.

Proof. As we see proofs as labeled trees (Remark 2), we will only locally replace the rules of the proof the following way to obtain the corresponding BDD ϕ:

- Replace axiom rules $\gamma, \gamma^\star$ by 1 and other axiom rules by 0
- Replace all $\parr$ and $\oplus$ rules by `DontCare` 1 `Then` $\cdot$ nodes
- In the $\otimes$ case, test which side the atoms $\gamma, \gamma^\star$ are attributed to *(this can be done locally by looking at the conclusions of the premise of the rule)* and replace it by a `If` 0 `Then` $\cdot$ `Else` $\cdot$ or a `If` 1 `Then` $\cdot$ `Else` $\cdot$ node accordingly
- Replace $\&_x$ rules by a `If` x `Then` $\cdot$ `Else` $\cdot$ nodes

We can see by induction that the resulting BDD is equivalent to $\mathcal{B}_\pi[\gamma, \gamma^\star]$. All these operations can be performed by looking only at the rule under treatment (and its immediate neighbors in the case of $\otimes$) and always replaces one rule by exactly one node. Therefore it is in AC_0. ◀

▶ **Corollary 20** (reduction). MALL⁻**equ** *reduces to* BDD**equ** *in* AC_0.

In the next section we focus on the equivalence of BDD and °BDD, proving first that the case of °BDD can be reduced to proof equivalence in MALL⁻. Then, we will show the problem of equivalence of BDD to be in Logspace, and that of °BDD to be Logspace-hard, thus characterizing the intrinsic complexity of proof equivalence in MALL⁻ as Logspace-complete.

Note that this contrasts with the classical result that equivalence of general Boolean formulas is coNP-complete. It turns out indeed that the classes of BDD we consider enjoy a number of properties that allow to solve equivalence more easily.

3 Equivalence of BDD

3.1 Equivalence of °BDD reduces to proof equivalence in MALL⁻

We now show that the converse of Lemma 19 holds for °BDD.

To do this, we rely on a formula B which will serve as the placeholder of a °BDD/V ϕ we want to encode; and a context Γ which contains one $\&_x$ connective for each variable x in V, organized in a way that allows for an inductive specification of °BDD.

Given an °BDD ϕ, we wish to obtain a proof π_ϕ of B, Γ such that dual pairs with one element in B will receive either the value ϕ or a value equivalent to $\overline{\phi}$ in the BDD slicing of π_ϕ; and on the other hand, the other dual pairs of Γ will receive equivalent values whatever the °BDD we encode is. This will lead to the fact that two such encoding proofs are equivalent if and only if the °BDD they encode are equivalent.

▶ **Notation 21.** *We fix atomic formulas* $\alpha_1, \ldots, \alpha_n \ldots$ *and* β *and write* $B = \beta \oplus \beta$*. In what follows, we will use* $\beta^{\mathtt{l}}$ *and* $\beta^{\mathtt{r}}$ *to refer respectively to the left and right copies of* β *in* B*; and likewise* $\alpha_i^{\mathtt{l}}$ *and* $\alpha_i^{\mathtt{r}}$ *for copies of* α_i *in* $\alpha_i \,\&\, \alpha_i$*.*

We write respectively π_0 *and* π_1 *the proofs*

$$\frac{\beta, \beta^\star}{B, \beta^\star}\oplus_1 \qquad \frac{\beta, \beta^\star}{B, \beta^\star}\oplus_\mathtt{r}$$

and for any n, we write $\pi^n_\&$ the proof

$$\frac{\alpha^\star_n, \alpha_n \qquad \alpha^\star_n, \alpha_n}{\alpha^\star_n, \alpha_n \,\&\, \alpha_n}\&$$

▶ **Definition 22** (encoding a °BDD)**.** Let ϕ be an °BDD over the variables $V = \{x_1, \dots, x_n\}$. We define the sequent

$$\Gamma^n = \ \big(\cdots(\beta^\star \otimes \alpha^\star_1) \otimes \alpha^\star_2) \otimes \cdots\big) \otimes \alpha^\star_n \ , \ \alpha_1 \,\&_{x_1} \alpha_1 \ , \ \alpha_2 \,\&_{x_2} \alpha_2 \ , \ \dots \ , \ \alpha_{n-1} \,\&_{x_{n-1}} \alpha_{n-1}$$

with $\Gamma^0 = \beta^\star$ and set $\Delta^n = \Gamma^n, \alpha_n \,\&_{x_n} \alpha_n$ with also $\Delta^0 = \beta^\star$.

We define the proof π_ϕ of conclusion B, Δ^n by induction on n

- The base cases are 0 and 1, encoded respectively as π_0 and π_1.
- Otherwise, if $\phi = \mathtt{If}\ x_n\ \mathtt{Then}\ \psi\ \mathtt{Else}\ \zeta$, with both ψ and ζ being °BDD$/\{x_1, \dots, x_{n-1}\}$, we have π_ψ and π_ζ defined by induction, and then
$$\pi_\phi = \ \dfrac{\dfrac{\langle\pi_\psi\rangle \qquad \langle\pi_\zeta\rangle}{B, \Gamma^n, \alpha_n}\otimes}{B, \Delta^n}\&_n$$
- The last case is $\phi = \mathtt{DontCare}\ x_n\ \mathtt{Then}\ \psi$, with ψ being a °BDD$/\{x_1, \dots, x_{n-1}\}$ so that we have π_ψ defined by induction, and then
$$\pi_\phi = \ \dfrac{\langle\pi_\psi\rangle \qquad \langle\pi^n_\&\rangle}{B, \Delta^n}\otimes$$

We still need to state in what sense π_ϕ is an encoding of ϕ: we turn the statement about the value of atoms of B we made in the beginning of this section into a precise property.

▶ **Lemma 23** (associated BDD slicing)**.** *Writing $\mathcal{B}$ the* BDD *slicing of π_ϕ, we have*

$$\mathcal{B}[\beta^\mathtt{l}, \beta^\star] = \phi \qquad \mathcal{B}[\beta^\mathtt{r}, \beta^\star] \sim \overline{\phi} \qquad \mathcal{B}[\alpha^\star_i, \alpha^\mathtt{l}_i] \sim x_i \qquad \mathcal{B}[\alpha^\star_i, \alpha^\mathtt{r}_i] \sim \overline{x}_i$$

Proof. By a routine inspection of induction cases. Let us only review $\phi = \mathtt{If}\, x_n\, \mathtt{Then}\, \psi\, \mathtt{Else}\, \zeta$. Let us write $\mathcal{B}_\phi$, $\mathcal{B}_\psi$ and $\mathcal{B}_\zeta$ the BDD slicings respectively associated to the proofs π_ϕ, π_ψ and π_ζ.

By induction we have that $\mathcal{B}_\psi[\beta^\mathtt{l}, \beta^\star] = \psi$ and $\mathcal{B}_\zeta[\beta^\mathtt{l}, \beta^\star] = \zeta$, therefore by definition of the BDD slicing associated to a & rule, we have that $\mathcal{B}_\phi[\beta^\mathtt{l}, \beta^\star] = \mathtt{If}\ x_n\ \mathtt{Then}\ \psi\ \mathtt{Else}\ \zeta = \phi$. The case of $\beta^\mathtt{r}$ is similar, but for its use of Lemma 27 from the next section.

As for the other pairs of occurences of dual atoms in the conclusion, let us have a look at the case of $[\alpha^\star_i, \alpha^\mathtt{l}_i]$: if by induction $\mathcal{B}_\psi[\alpha^\star_i, \alpha^\mathtt{l}_i] \sim x_i$ and $\mathcal{B}_\zeta[\alpha^\star_i, \alpha^\mathtt{l}_i] \sim x_i$, then by definition $\mathcal{B}_\phi[\alpha^\star_i, \alpha^\mathtt{l}_i] \sim \mathtt{If}\ x_n\ \mathtt{Then}\ x_i\ \mathtt{Else}\ x_i \sim x_i$. The case of $\alpha^\mathtt{r}_i$ is similar. ◀

▶ **Corollary 24** (equivalence)**.** *If ϕ and ψ are two* °BDD$/\{x_1, \dots, x_n\}$*, then $\pi_\phi \sim \pi_\psi$ if and only if $\phi \sim \psi$.*

▶ **Lemma 25** (computing the encoding)**.** *Given a* °BDD$/\{x_1, \dots, x_n\}$ *ϕ, π_ϕ can be computed in* AC_0*.*

Proof. As in the proof of Lemma 19, we show that the inductive definition can in fact be seen as a local graph transformation introducing nodes of bounded size:

- Replace each 0 node by $\dfrac{\langle\pi_0\rangle}{\vdots\ B, \beta^\star}$ and 1 node by $\dfrac{\langle\pi_1\rangle}{\vdots\ B, \beta^\star}$.
- Replace each $\texttt{DontCare}\ x_i\ \texttt{Then}\ \cdot$ by $\dfrac{\dfrac{\langle\pi_{\&}^{x_i}\rangle}{\vdots\ \ \alpha_i^\star, \alpha_i \,\&_{x_i}\, \alpha_i}\otimes}{B, \Delta^i}$
- Replace each $\texttt{If}\ x_i\ \texttt{Then}\ \cdot\ \texttt{Else}\ \cdot$ by $\dfrac{\dfrac{\vdots\ \ \alpha_i^\star, \alpha_i}{B, \Gamma^i, \alpha_i}\otimes \quad \dfrac{\vdots\ \ \alpha_i^\star, \alpha_i}{B, \Gamma^i, \alpha_i}\otimes}{B, \Delta^i}\,\&_{x_i}$

For any of these replacements, we see that the choice of the case to apply and the label of the resulting block (of bounded size) of rules replacing a node depends only on the label of the node we are replacing, therefore the transformation is in AC_0. ◀

▶ **Corollary 26** (reduction). °BDD**equ** *reduces to* MALL⁻**equ** *in* AC_0.

To sum up, we have so far the following chain of AC_0 reductions:

$$\text{°BDD}\mathbf{equ} \;\to\; \text{MALL}^{-}\mathbf{equ} \;\to\; \text{BDD}\mathbf{equ}$$

3.2 Logspace-completeness

We prove in this section that all these equivalence problems are Logspace-complete. We begin by listing a few useful properties of BDD that will allow to design a Logspace decision procedure for their equivalence. Then, we prove the Logspace-hardness by reducing to °BDD**equ** a Logspace-complete problem on line graph orderings.

The starting point is the good behavior of BDD with respect to negation. In the following lemma, we consider the negation of a BDD which is not strictly speaking a BDD itself: we think of it as the equivalent Boolean formula, the point being precisely to show that this Boolean formula can be easily expressed as a BDD.

▶ **Lemma 27** (negation). *If ϕ and ψ are* BDD *and X is either 0, 1 or a variable, we have*

$$\overline{\texttt{If}\ X\ \texttt{Then}\ \phi\ \texttt{Else}\ \psi} \sim \texttt{If}\ X\ \texttt{Then}\ \overline{\phi}\ \texttt{Else}\ \overline{\psi}$$

$$\overline{\texttt{DontCare}\ X\ \texttt{Then}\ \phi} \sim \texttt{DontCare}\ X\ \texttt{Then}\ \overline{\phi}$$

Proof. First we can transform our expression by

$$\overline{\texttt{If}\ X\ \texttt{Then}\ \phi\ \texttt{Else}\ \psi} \sim \overline{X.\phi + \overline{X}.\psi} \sim (\overline{X} + \overline{\phi}).(X + \overline{\psi}) \sim X.\overline{\psi} + \overline{X}.\overline{\phi} + \psi.\phi$$

then we can apply the so-called "consensus rule" of Boolean formulas

$$X.\overline{\psi} + \overline{X}.\overline{\phi} + \psi.\phi \sim X.\overline{\phi} + \overline{X}.\overline{\psi} \sim \texttt{If}\ X\ \texttt{Then}\ \overline{\phi}\ \texttt{Else}\ \overline{\psi}$$

The case of `DontCare` is obvious. ◀

▶ **Corollary 28.** *If ϕ is a* BDD*, then there is a* BDD *$\widehat{\phi}$ such that $\overline{\phi} \sim \widehat{\phi}$.*
Moreover $\widehat{\phi}$ can be computed in logarithmic space.

Proof. An induction on the previous lemma shows that we can obtain the negation of a BDD simply by flipping the 0 nodes to 1 nodes and conversely. Hence the transformation is even in AC_0. ◀

Then, we show that a BDD can be seen as a sum of monomials through a Logspace transformation.

▶ **Lemma 29** (BDD as sums of monomials). *If ϕ is a* BDD*, then there is a formula $\phi_{\mathtt{m}}$ which is a sum of monomials and is such that $\phi \sim \phi_{\mathtt{m}}$.*

Moreover $\phi_{\mathtt{m}}$ can be computed in logarithmic space.

Proof. For each 1 node in ϕ, go down to the root of ϕ and output one by one the variables of any $\mathtt{If}\, x\, \mathtt{Then} \cdot \mathtt{Else} \cdot$ encountered: this produces the monomial associated to this 1 node. Then $\phi_{\mathtt{m}}$ is the sum of all the monomials obtained this way and is clearly equivalent to ϕ. The procedure is in Logspace because we only need to remember which 1-leave we are treating and where we are in the tree (when going down) at any point. ◀

Putting all this together, we finally obtain a space-efficient decision procedure. Note however that it is totally sub-optimal in terms of time: to keep with the logarithmic space bound, we have to recompute a lot of things rather than store them.

▶ **Corollary 30** (BDD**equ** is in Logspace). *There is a logarithmic space algorithm that, given two* BDD *ϕ and ψ, decides wether they are equivalent.*

Proof. The BDD ϕ and ψ are equivalent if and only if $\phi \Leftrightarrow \psi = (\overline{\phi} + \psi).(\overline{\psi} + \phi) \sim 1$ that is to say (by passing to the negation) $(\overline{\psi}.\phi) + (\overline{\phi}.\psi) \sim 0$, which holds if and only if both $\overline{\psi}.\phi \sim 0$ and $\overline{\phi}.\psi \sim 0$.

But then, considering the first one (the other being similar) we can rewrite it in logarithmic space using the two above lemmas as $(\widehat{\psi})_{\mathtt{m}}.\phi_{\mathtt{m}} \sim 0$. This holds if and only if for all pairs (m, m') of one monomial in $(\widehat{\psi})_{\mathtt{m}}$ and one monomial in $\phi_{\mathtt{m}}$, m and m' are in conflict; which can be checked in logarithmic space using Remark 7. ◀

Let us now introduce an extremely simple, yet Logspace-complete problem [4], which will ease the Logspace-hardness part of our proof.

▶ **Definition 31** (order between vertices). *Order between vertices* (**ORD**) is the following decision problem:

"Given a directed graph $G = (V, E)$ that is a line [4] *and two vertices $f, s \in V$ do we have $f < s$ in the total order induced by G?"*

▶ **Lemma 32. ORD** *reduces to* °BDD**equ** *in* AC_0.

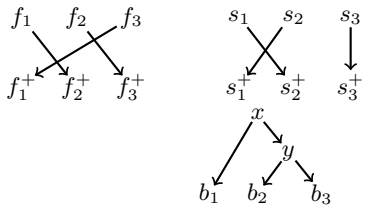
Proof. Again we are going to build a local graph transformation that is in AC_0.

First, we assume w.l.o.g. that the begin b and the exit e vertices of G are different from f and s. We write f^+ and s^+ the vertices immediately after f and s in G.

Then, we perform a first transformation by replacing the graph with three copies of itself (this can be done by locally scanning the graph and create labeled copies of the vertices

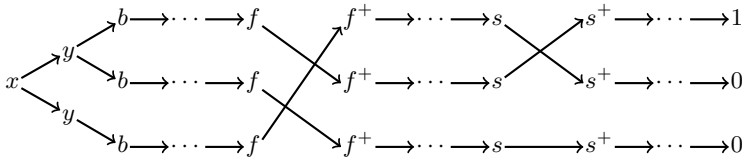
[4] We use the standard definition of graph as a pair (V, E) of sets of vertices and edges (oriented couples of vertices $x \to y$). A graph is a *line* if it is connected and all the vertices have in-degree and out-degree 1, except the *begin* vertex which has in-degree 0 and out-degree 1 and the *exit* vertex which has in-degree 1 and out-degree 0. A line induces a total order on vertices through its transitive closure.

and edges). We write x_i to refer to the copy of the vertex x in the graph i. The second transformation is a rewiring of the graph as follows: erase the edges going out of the f_i and s_i and replace them as pictured in the two first subgraphs:



Let us call G_r the rewired graph and G_n the non-rewired graph. To each of them we add two binary nodes x and y connected to the begin vertices b_i as pictured in the third graph above. Then we can produce two corresponding °BDD ϕ_r and ϕ_n by replacing the exit vertices e_1, e_2, e_3 by 1, 0, 0 respectively, x and y by a `If` x `Then` (`DontCare` y `Then` $\cdot$) `Else` (`If` y `Then` $\cdot$ `Else` $\cdot$) block of nodes; and any other v_i vertex by a `DontCare` v `Then` $\cdot$. It is then easy to see that if $f < s$ in the order induced by G if and only if ϕ_r and ϕ_n are equivalent.

Let us illustrate graphically what happens in the case where $f < s$: we draw the resulting °BDD as a labeled graph with the convention that a node labeled with z with out-degree 1 is a `DontCare` z `Then` $\cdot$ and a node labeled with z with out-degree 2 is a `If` z `Then` $\cdot$ `Else` $\cdot$ node with the upper edge corresponding to the `Then` branch and the lower edge corresponding to the `Else` branch.



◀

▶ Remark 33. The above construction relies on the fact that there are non-commuting permutations on the set of three elements: in a sense we are just attributing two non-commuting σ and τ to f and s and make sure that the order in which they intervene affects the equivalence class of the resulting °BDD. An approach quite similar in spirit with the idea of *permutation branching program* [1].

We can now extend our chain of reductions with the two new elements from this section

ORD (Logspace-hard) $\rightarrow$ °BDD**equ** $\rightarrow$ MALL$^-$**equ** $\rightarrow$ BDD**equ** ($\in$ Logspace)

so in the end we get our main result:

▶ **Theorem 34** (Logspace-completeness). *The decision problems* °BDD**equ**, MALL$^-$**equ** *and* BDD**equ** *are* Logspace*-complete under* AC_0 *reductions.*

4 Conclusion

In this work, we characterized precisely the complexity of proof equivalence in MALL$^-$ as Logspace-complete, contrasting greatly with the situation for the MLL fragment which has a Pspace-complete equivalence problem. We did so by establishing a correspondence between MALL$^-$ proofs and specific classes of BDD.

This path we took for proving our result is interesting in itself since the established correspondence allows a transfer of ideas in both directions. In particular, any progress in the theoretical problem of finding a correct notion of proofnet for MALL$^-$ would yield potential

applications to BDD, a notion of widespread practical use. An idea to explore might be the notion of *conflict net* defined by D. Hughes in an unpublished note [12]. Roughly speaking, the principle is to consider a presentation of proofnets with the information of the links that cannot be present at the same time, rather than giving an explicit formula to compute their presence, as it is the case with monomial proofnets or the BDD slicings we introduced.

On the other hand, since many optimization problems regarding BDD are known to be NP-complete, a finer view at the encoding of Section 3.1 in addition to basic constraints about what we expect from a notion of proofnet should lead to an impossibility result, even though the equivalence problem for MALL^- is only Logspace-complete.

Acknowledgements to people from `cstheory.stackexchange.com` for pointing the author to the notion of BDD; to Dominic Hughes for the live feedback during the redaction of the article; to Clément Aubert, for his help in understanding AC_0 reductions.

References

1 David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. *Journal of Computer and System Sciences*, 38(1):150 – 164, 1989.

2 Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput.*, 35(8):677–691, August 1986.

3 Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM J. Comput.*, 13(2):423–439, 1984.

4 Kousha Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54(3):400 – 411, 1997.

5 Jean Gallier. On the correspondence between proofs and lambda-terms. In Philippe de Groote, editor, *The Curry-Howard isomorphism*, Cahiers du Centre de Logique, pages 55–138. Academia, 1995.

6 Jean-Yves Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987.

7 Jean-Yves Girard. Proof-nets: The parallel syntax for proof-theory. *Logic and Algebra*, 180:97–124, May 1996.

8 Stefano Guerrini and Andrea Masini. Parsing MELL Proof Nets. *Theoretical Computer Science*, 254(1-2):317–335, 2001.

9 Willem Heijltjes and Robin Houston. No Proof Nets for MLL with Units: Proof Equivalence in MLL is PSPACE-complete. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, CSL-LICS '14, pages 50:1–50:10, New York, NY, USA, 2014. ACM.

10 Willem Heijltjes and Lutz Straßburger. Proof nets and semi-star-autonomous categories. *Mathematical Structures in Computer Science*, FirstView:1–40, 11 2014.

11 Dominic J. D. Hughes. Simple multiplicative proof nets with units. Technical report, 2005.

12 Dominic J. D. Hughes. Abstract p-time proof nets for MALL: Conflict nets. *arXiv: math.LO/0801.2421v1*, 2008.

13 Dominic J. D. Hughes and Rob J. van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Trans. Comput. Log.*, 6(4):784–842, 2005.

14 Dominic J. D. Hughes and Rob J. van Glabbeek. MALL proof nets identify proofs modulo rule commutation. `http://boole.stanford.edu/~dominic/MALL-equiv.pdf`, *to appear*, 2015.

15 Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams.* Addison-Wesley Professional, 12th edition, 2009.

16 Olivier Laurent and Roberto Maieli. Cut elimination for monomial MALL proof nets. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, pages 486–497, 2008.