Unique perfect matchings and proof nets

Lê Thành Dũng Nguyễn[1]
Département d'informatique, École normale supérieure, Paris Sciences et Lettres
45 rue d'Ulm, 75005 Paris, France
Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS
99 avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
nltd@nguyentito.eu
https://orcid.org/0000-0002-6900-5577

Abstract

This paper establishes a bridge between linear logic and mainstream graph theory, building previous work by Retoré (2003). We show that the problem of correctness for MLL+Mix proof nets is equivalent to the problem of uniqueness of a perfect matching. By applying matching theory, we obtain new results for MLL+Mix proof nets: a linear-time correctness criterion, a quasi-linear sequentialization algorithm, and a characterization of the sub-polynomial complexity of the correctness problem. We also use graph algorithms to compute the dependency relation of Bagnol et al. (2015) and the kingdom ordering of Bellin (1997), and relate them to the notion of blossom which is central to combinatorial maximum matching algorithms.

2012 ACM Subject Classification Theory of computation → Linear logic, Mathematics of computing → Matchings and factors, Mathematics of computing → Graph algorithms

Keywords and phrases correctness criteria, matching algorithms

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.25

Acknowledgements This work started as a side project during an internship in the Operations Research team at the Laboratoire d'Informatique de Paris 6, supervised by Christoph Dürr, who taught the author the expressive power of perfect matchings; this paper would not exist without him. Thanks also to Kenji Maillard, Michele Pagani, Marc Bagnol, Antoine Amarilli, Alexis Saurin, Stefano Guerrini and Virgile Mogbil for discussions, references and encouragements, and to Thomas Seiller for his writing advice.

1 Introduction

One of the major novelties introduced at the birth of linear logic [13] was a representation of proofs as *graphs*, instead of trees as in natural deduction or sequent calculus. A distinctive property of these *proof nets* is that checking that a proof is correct cannot be done merely by a local verification of inference steps: among the graphs which locally look like proof nets, called *proof structures*, some are invalid proofs. Hence the problem of *correctness*: given a proof structure, is it a real proof net?

A lot of work has been devoted to this decision problem, and in the case of the multiplicative fragment of linear logic (MLL), whose proof nets are the most satisfactory, it can be considered solved from an algorithmic point of view. Indeed, Guerrini [14] and Murawski and Ong [22] have found linear-time tests for MLL correctness; the problem has also been shown to be NL-complete by Jacobé de Naurois and Mogbil [17]. Both the linear-time algorithms

[1] Partially supported by the ANR project Elica (ANR-14-CE25-0005).

we mentioned also solve the corresponding search problem: computing a *sequentialization* of a MLL proof net, i.e. a translation into sequent calculus.

However, for MLL extended with the *Mix rule* [10] (MLL+Mix), the precise complexity of deciding correctness has remained unknown (though a polynomial-time algorithm was given by Danos [7]). Thus, one of our goals in this paper is to study the following problems:

▶ **Problem** (MixCorr)**.** Given a proof structure π, is it an MLL+Mix proof net?

▶ **Problem** (MixSeq)**.** Reconstruct a sequent calculus proof for an MLL+Mix proof net.

It turns out that a *linear-time* algorithm for MixCorr follows immediately from already known results. The key is to use a construction by Retoré [26, 27] to reduce it to the problem of *uniqueness of a given perfect matching*, which can be solved in linear time [11]:

▶ **Problem** (UniquenessPM)**.** Given a graph G, together with a *perfect matching* M of G, is M the only perfect matching of G? Equivalently, is there no *alternating cycle* for M?

This brings us to the central idea of this paper: *from the point of view of algorithmics, MLL+Mix proof nets and unique perfect matchings are essentially the same thing.* This allows us to apply matching theory to the study of proof nets, leading to several new results. Indeed, one would expect graph algorithms to be of use in solving problems on proof structures, since they are graphs! But for this purpose, a bridge between the theory of proof nets and mainstream graph theory is needed, whereas previous work on the former mostly made use of "homemade" objects such as *paired graphs* (an exception being Murawski and Ong's use of *dominator trees*). By building on Retoré's discovery of a connection with perfect matchings, this paper proposes such a bridge.

Plan of the paper and contributions. First, we establish our equivalence by giving a translation from proof structures to graphs equipped with perfect matchings and vice versa (§3). In the first direction, instead of reusing Retoré's construction, we propose an alternative having better properties with respect to sequentialization.

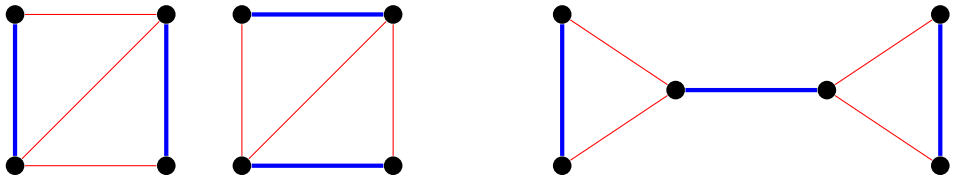
As already mentioned, we give the first linear-time algorithm for MixCorr (§4.1). As for its sub-polynomial complexity (§4.2), we show that MixCorr is in randomized NC and in quasi-NC (informally, NC is the class of problems with efficient *parallel* algorithms). On the other hand, we have a sort of hardness result: if MixCorr were in NC – in particular, if it were in NL, as for MLL without Mix – this would imply a solution to a long-standing conjecture concerning the related *unique perfect matching* problem:

▶ **Problem** (UniquePM [19, 11, 15])**.** Given a graph G, determine whether it admits exactly one perfect matching and, if so, find this matching.

We then turn to the sequentialization problem, for which we provide a graph-theoretic reformulation, and an algorithm for this reformulation. This gives us a *quasi-linear* time[2] solution to MixSeq (§5); to our knowledge, this beats previous algorithms for MixSeq.

As a demonstration of our matching-theoretic toolbox, we also show how to compute some information on the set of *all* sequentializations, namely Bellin's *kingdom ordering* [4] of the links of a MLL+Mix proof net (rediscovered by Bagnol et al. [1] under the name of *order*

[2] More precisely, $O(n(\log n)^2(\log\log n)^2)$ time. Both this and our quasi-NC algorithms rely on very recent advances, respectively on dynamic bridge-finding data structures [16] and on the perfect matching existence problem [28]. Any further progress on these problems would lead to an improvement of our complexity bounds.



(a) Two perfect matchings of the same graph. **(b)** A graph with a unique perfect matching.

Figure 1 Examples of perfect matchings. The edges in the matchings are thick and blue.

of introduction). We give a polynomial time and a quasi-NC algorithm (§6.1), both relying on an effective characterization of this ordering. By rephrasing this characterization (§6.2), we get a purely graph-theoretic new theorem of independent interest about objects which play a major role in matching algorithms, namely *blossoms* [9].

2 Preliminaries

Graph-theoretic terminology. By default, "graph" refers to an *undirected* graph. Our *paths* and *cycles* are not allowed to contain repeated vertices[3]; we will sometimes identify them with their sets of edges (which characterize them) and apply set operations on them. A *bridge* of a graph is an edge whose removal increases the number of connected components.

For directed graphs, the notion of connectedness we consider is *weak connectedness*, i.e. connectedness of the graph obtained by forgetting the edge directions. A *predecessor* (resp. *successor*) of a vertex is the source (resp. target) of some incoming (resp. outgoing) edge.

Complexity classes. We refer to [17, §1.4] for the logarithmic space classes L and NL and to [6] for the class AC^0 of constant-depth circuits. The class NC^k (resp. quasi-NC^k [3]) consists of the problems which can be solved by a uniform family of circuits of depth $O(\log^k n)$ and polynomial (resp. quasi-polynomial, i.e. $2^{O(\log^c n)}$) size; $\mathsf{NC} = \bigcup_k \mathsf{NC}^k$ and $\text{quasi-}\mathsf{NC} = \bigcup_k \text{quasi-}\mathsf{NC}^k$. It is well-known that $\mathsf{AC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{NC}^2 \subseteq \mathsf{NC} \subseteq \mathsf{P}$.

2.1 Perfect matchings, alternating cycles and sequentialization

▶ **Definition 2.1.** Let $G = (V, E)$ be a graph. A *matching* (resp. *perfect matching*) M in G is a subset of E such that every vertex in V is incident to *at most one* (resp. *exactly one*) edge in M. An *alternating path* (resp. *cycle*) for M is a path (resp. cycle) where, for every pair of consecutive edges, one of them is in the matching and the other one is not.

Testing the existence of a perfect matching in a graph – or, more generally, finding a maximum cardinality matching – is one of the central computational problems in graph theory. Combinatorial maximum matching algorithms, starting[4] with Edmonds's *blossom algorithm* [9][5], use alternating paths to iteratively increase the size of the matching; similarly, alternating cycles are important for the problems UniquenessPM and UniquePM because they witness the *non-uniqueness* of perfect matchings.

[3] This choice of terminology is common, see e.g. [2, §1.4].

[4] Note that the problem was solved long before in the special case of bipartite graphs. In fact, a solution for this case was found in Jacobi's posthumous papers.

[5] This paper is one of the first to propose defining efficient algorithms as polynomial-time algorithms; it also contributed to the birth of the field of polyhedral combinatorics.

▶ **Lemma 2.2** (Berge). *Let G be a graph and M be a perfect matching of G. Then if $M' \neq M$ is a perfect matching, the symmetric difference $M \triangle M'$ is a vertex-disjoint union of cycles, which are alternating for both M and M'. Conversely, if C is an alternating cycle for M, then $M \triangle C$ is another perfect matching.*

As an example, consider Figure 1a. The matching on the left admits an alternating cycle, the outer square; by taking the symmetric difference between this matching and the set of edges of the cycle, one gets the matching on the right. Conversely, the symmetric difference between both matchings (which, in this case, is their union) is the square. Note also that in Figure 1b, there is no alternating cycle because vertex repetitions are disallowed.

Another approach to finding perfect matchings, using linear algebra, was initiated by Lovász [20] and leads to a *randomized* NC algorithm by Mulmuley et al. [21]. Recently, Svensson and Tarnawski have shown that this algorithm can be derandomized to run in deterministic quasi-NC [28].

There is also a considerable body of purely mathematical work on matchings, starting from the 19th century. Let us mention for our purposes a result dating from 1959.

▶ **Theorem 2.3** (Kotzig [18]). *Let G be a graph. Suppose that G admits a unique perfect matching M. Then M contains a bridge of G.*

As remarked by Retoré [27], Kotzig's theorem leads to an inductive characterization of the set of graphs equipped with a unique perfect matching.

▶ **Theorem 2.4** (Sequentialization for unique perfect matchings [27]). *The class $\mathcal{UPM}$ of graphs equipped with an unique perfect matching is inductively generated as follows:*

- *The empty graph (with the empty matching) is in $\mathcal{UPM}$.*
- *The disjoint union of two non-empty members of $\mathcal{UPM}$ is in $\mathcal{UPM}$.*
- *Let $(G = (V, E), M \subseteq E) \in \mathcal{UPM}$ and $(G' = (V', E'), M' \subseteq E') \in \mathcal{UPM}$, with V and V' disjoint. Let $U \subseteq V$, $U' \subseteq V'$ such that $U \neq \emptyset$ (resp. $U' \neq \emptyset$) unless G (resp. G') is the empty graph, and let x, x' be two fresh vertices not in V nor V'. Then $(G'' = (V'', E''), M'' \subseteq E'') \in \mathcal{UPM}$, where*
  - $V'' = V \cup V' \cup \{x, x'\}$
  - $E'' = E \cup E' \cup \{(x, x')\} \cup (U \times \{x\}) \cup (U' \times \{x'\})$
  - $M'' = M \cup M' \cup \{(x, x')\}$

▶ Remark. By relaxing the non-emptiness condition on U and U', the disjoint union operation becomes unnecessary; this is actually the original statement [27, Theorem 1].

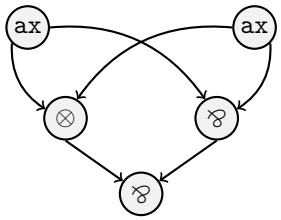
The inspiration for the above theorem comes from linear logic: it is a graph-theoretic version of the sequentialization theorems for proof nets, with Kotzig's theorem being analogous to the "splitting lemmas" which appear in various proofs of sequentialization.

2.2 Proof structures, proof nets and the correctness criterion

A proof structure is some kind of graph-like object made of "nodes" (or "formulae") and "links", with the precise definition varying in the literature. Since our aim is to apply results from graph theory, it will be helpful to commit to a representation of proof structures as graphs. (We write $\deg^-$ for the indegree and $\deg^+$ for the outdegree of a vertex.)

▶ **Definition 2.5.** A *proof structure* is a non-empty directed acyclic multigraph (V, A) with a labeling of the vertices $l : V \to \{\mathtt{ax}, \otimes, \parr\}$ such that, for $v \in V$:

- if $l(v) = \mathtt{ax}$, then $\deg^-(v) = 0$ and $\deg^+(v) \leq 2$,
- if $l(v) \in \{\otimes, \parr\}$, then $\deg^-(v) = 2$ and $\deg^+(v) \leq 1$.



$$\dfrac{\dfrac{\dfrac{}{\vdash A, A^\perp}\,\mathtt{ax} \quad \dfrac{}{\vdash B, B^\perp}\,\mathtt{ax}}{\vdash A \otimes B, A^\perp, B^\perp}\,\otimes}{\vdash A \otimes B, A^\perp \parr B^\perp}\,\parr}{\vdash (A \otimes B) \parr (A^\perp \parr B^\perp)}\,\parr$$

Figure 2 A proof net (left) and its sequentialization (right), written as a sequent calculus proof. Edges are usually labeled by the MLL formulae appearing in the sequentialization; since we focus on the combinatorics of proof structures and not on their logical meaning, we omit them here.

$$\frac{}{\vdash A, A^\perp}(\mathtt{ax}\text{-rule}) \quad \frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}(\otimes\text{-rule}) \quad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}(\parr\text{-rule}) \quad \frac{\vdash \Gamma \quad \vdash \Delta}{\vdash \Gamma, \Delta}(\text{Mix rule})$$

Figure 3 Rules for the MLL+Mix sequent calculus; note the correspondence with Definition 2.6.

Vertices of a proof structure will also be called *links*. A *terminal link* is a link with outdegree 0. A *sub-proof structure* is a vertex-induced subgraph which is a proof structure.

▶ **Definition 2.6.** The set of *MLL proof nets* is the subset of proof structures inductively generated by the following rules:

- **ax-rule**: a proof structure with a single ax-link is a proof net.
- **⊗-rule**: if N and N' are proof nets, u is a link of N and v is a link of N', then taking the disjoint union of N and N', adding a new ⊗-link w, an edge from u to w and an edge from v to w gives a proof net, as long as the resulting graph is a proof structure (i.e. the degree constraints are satisfied).
- **⅋-rule**: if N is a proof net and u, v are links of N, then adding a new ⅋-link w, an edge from u to w and an edge from v to w gives a proof net, with the same proviso as above.

The set of *MLL+Mix proof nets* is inductively generated by the above rules together with the **Mix rule**: if N and N' are proof nets, their disjoint union is a proof net.

A proof structure is said to be *correct* if it is a MLL+Mix proof net.

▶ Remark. As with any inductively defined set, membership proofs for the set of MLL (resp. MLL+Mix) proof nets may be presented as inductive derivation trees, which are isomorphic to the usual *sequent calculus proofs* of MLL (resp. MLL+Mix): see Figure 2 for an example, and Figure 3 for the inference rules of the sequent calculus.

▶ Remark. The proof structures and proof nets defined here are *cut-free*. This restriction is without loss of generality, since cut link has exactly the same behavior as a terminal ⊗-link with respect to correctness and sequentialization.

To tackle the problem of correctness, it is useful to have non-inductive characterizations of proof nets, called *correctness criteria*, at our disposal. Many of them are formulated using the notion of *paired graphs*. We will state a criterion first discovered by Danos and Regnier for MLL [8] and extended to MLL+Mix by Fleury and Retoré [10].

▶ **Definition 2.7.** A *paired graph* consists of an undirected graph $G = (V, E)$ and a set $\mathcal{P}$ of unordered pairs of edges such that:

- if $\{e, f\} \in \mathcal{P}$, then e and f have a vertex in common;
- the pairs are disjoint: if $p, p' \in \mathcal{P}$ and $p \neq p'$, then $p \cap p' = \emptyset$.

When $\{e, f\} \in \mathcal{P}$, the edges e and f are said to be *paired*.

A *switching* of this paired graph is a spanning subgraph of G which intersects each pair of $\mathcal{P}$ exactly once. A *feasible cycle* is a cycle which intersects each pair of $\mathcal{P}$ at most once.

▶ Remark. Equivalently, feasible cycles are cycles which exist in some switching.

▶ **Definition 2.8.** Let π be a proof structure. Its *correctness graph* $C(\pi)$ is the paired graph obtained by forgetting the directions of the edges and the labels of the vertices in π, and pairing together two edges when their targets[6] are the same $⅋$-link.

A *feasible cycle* in π is a sequence of edges of π whose image in $C(\pi)$ is a feasible cycle.

▶ **Theorem 2.9** (Danos–Regnier correctness criterion)**.** *π is a MLL (resp. MLL+Mix) proof net if and only if all the switchings of $C(\pi)$ are trees (resp. forests).*

▶ Remark. Equivalently, π is a MLL+Mix proof net iff it contains no feasible cycle.

The above is usually called a *sequentialization theorem*: it means that a proof structure which satisfies the correctness criterion admits a sequent calculus derivation.

The analogy with Theorem 2.4 is that proof nets are to proof structures what *unique* perfect matchings are to perfect matchings. The next section is dedicated to formalizing this analogy into an equivalence.

3 An equivalence through mutual reductions

We will now see how to turn a proof structure into a graph equipped with a perfect matching, in such a way that feasible cycles become alternating cycles, and vice versa.

Such a translation from proof structures to perfect matchings was first proposed by Retoré [27], under the name of *R&B-graphs*. However, we would like to deduce Theorem 2.9 as an immediate corollary of sequentialization for unique perfect matchings (Theorem 2.4), which is not possible with R&B-graphs – instead, one must resort to a proof of induction using Kotzig's theorem (Theorem 2.3), see [26, §2.4]. Thus, we propose here our own *graphification* construction. We also define the *proofification* construction, going from perfect matchings to proof structures.

▶ Remark. The nature of the object corresponding to a matching edge in a proof structure will vary depending on the translation considered: for graphifications, they correspond to links, whereas in the case of proofifications, they are translated into $⊗$-links (and for R&B-graphs, they correspond to edges or terminal links).

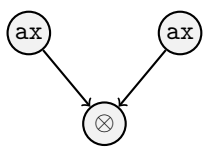
Thus, by taking the proofification of a graphification of a proof structure, one gets a different proof structure, with the ax-links and $⅋$-links of the former being sent to $⊗$-links of the latter (see Figure 4 for an example). It is unclear whether this transformation has any meaning in terms of linear logic; in particular it does not preserve correctness for MLL without Mix.

3.1 From proof structures to perfect matchings

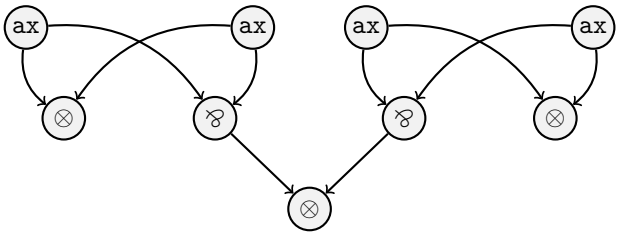
▶ **Definition 3.1.** Let π be a proof structure and L be its set of links. The *graphification* of π is the graph $G = (V, E)$ equipped with a perfect matching $M \subseteq E$ with

- the matching edges corresponding to the links: $V = \bigcup_{l \in L} \{a_l, b_l\}$, $M = \{(a_l, b_l) \mid l \in L\}$,
- and the remaining edges in $E \setminus M$ reflect the incoming edges of the $⊗$-links and $⅋$-links, as specified by Figure 5a.

[6] That is, the targets of the directed edges in π they come from.

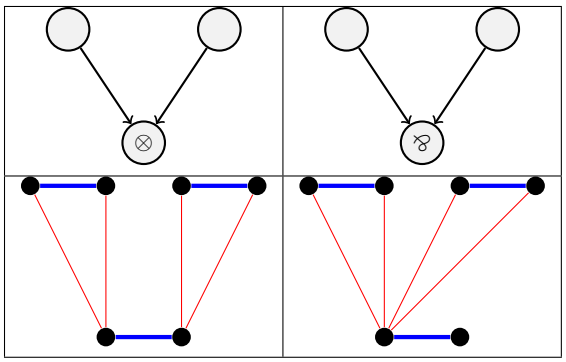


(a) Before graphification. The left ⊗-link and the right ⊗-link of (b) correspond to the ax-links here.

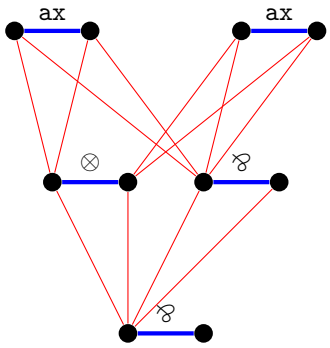


(b) After proofification.

Figure 4 Composing graphification and proofification: as we will see, the graphification of the proof net of (a) is the graph of Figure 1b, and, in turn, the proofification of Figure 1b is (b).



(a) Translation rules for sets of incoming edges.



(b) Graphification of the proof structure of Figure 2

Figure 5 The graphification construction.

Figure 5b shows an example of this construction. As another example, Figure 1b is the graphification of Figure 4a.

▶ **Proposition 3.2** (Graphification-based correctness criterion)**.** *A proof structure satisfies the Danos–Regnier criterion for MLL+Mix if and only if the perfect matching of its graphification is unique.*

In the case of R&B-graphs, there is an actual bijection between the feasible cycles of a proof structure and the alternating cycles of its R&B-graph. That said, the main technical advantages of graphifications over R&B-graphs are summarized by the following properties.

▶ **Lemma 3.3.** *Let π be a proof structure with graphification (G, M) and l be a link of π such that $(a_l, b_l) \in M$ is a bridge of G. Then l is a terminal link in π, and if l is a $\otimes$-link, then removing l from π disconnects its predecessors.*

▶ **Theorem 3.4.** *Let π be a proof structure and (G, M) be its graphification. There is a bijection between the sequent calculus proofs corresponding to π (if any) and the sequentializations (i.e. the derivation trees for the inductive definition of Theorem 2.4) of (G, M) (if any), through which occurrences of Mix rules correspond to disjoint unions and conversely.*

In particular, π is a MLL+Mix proof net if and only (G, M) admits a sequentialization, that is, according to Theorem 2.4, if and only if M is the only perfect matching of G. Proposition 3.2 tells us that this is equivalent to π satisfying the Danos–Regnier acyclicity criterion. Therefore, this criterion characterizes MLL+Mix proof nets: as we wanted, we just proved the sequentialization theorem for MLL+Mix (Theorem 2.9).

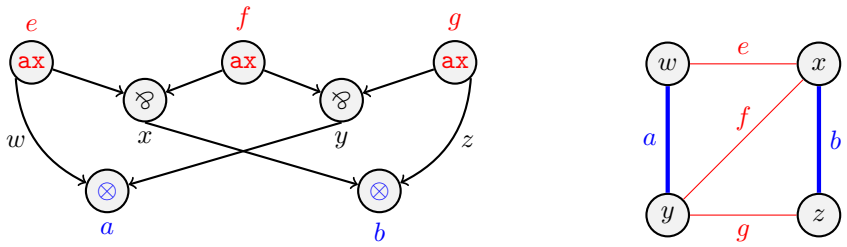


Figure 6 The proofification of the graph of Figure 1a.

3.2 From perfect matchings to proof structures

The translation we present below involves "k-ary ⅋-links". When $k > 1$, these are just binary trees of $k-1$ ⅋-links (correctness is independent of the choice of binary tree: semantically, this is associativity of ⅋) with k leaves (incoming edges) and a single root (outgoing edge); the $k = 1$ case corresponds to a single edge and no link.

▶ **Definition 3.5.** Let $G = (V, E)$ be a graph and M be a perfect matching of G. We define the *proofification* of (G, M) as the proof structure π built as follows:

- For each non-matching edge $e = (u, v) \in E \setminus M$, we create an `ax`-link $\texttt{ax}_e$ whose two outgoing edges we will call $A_{u,v}$ and $A_{v,u}$.
- For each vertex $u \in V$, if $\deg(u) > 1$, we add a k-ary ⅋-link with $k = \deg(u) - 1$, whose incoming edges are the $A_{u,v}$ for all neighbors v of u such that $(u, v) \notin M$, and we call its outgoing edge B_u. If $\deg(u) = 1$, we add an `ax`-link calling one of its outgoing edges B_u.
- For each matching edge $(u, v) \in M$, we add an ⊗-link whose incoming edges are B_u and B_v. These ⊗-links are the terminal links of π.

See Figure 6 for an annotated example of proofification. The reader may also check that the proof net in Figure 4b is the proofification of the graph in Figure 1b.

▶ **Proposition 3.6.** *Let G be a graph and M be a perfect matching of G. The alternating cycles for M in G are in bijection with the feasible cycles in the proofification of (G, M).*

▶ **Proposition 3.7.** *Let G be a graph with a unique perfect matching M and let π be the proofification of (G, M). A matching edge $e \in M$ is a bridge of G if and only if its corresponding ⊗-link is introduced by the last rule of some sequentialization of π.*

However, unlike the case of graphifications, this does not give us a bijection between the sequentializations of a unique perfect matching and those of its proofification.

4 On the complexity of MLL+Mix correctness

Through the translations of the previous section, MLL+Mix proof *nets* become *unique* perfect matchings and conversely: these translations provide *reductions* between the problems MixCorr and UniquenessPM, allowing us to draw complexity-theoretic conclusions on proof nets from known results in graph theory. We first look at the time complexity of MixCorr, then turn to its complexity under constant-depth (AC^0) reductions.

4.1 A linear-time algorithm

Since graphifications (§3.1) can be computed in linear time, and UniquenessPM can also be decided in linear time [11, §3], we immediately get:

▶ **Theorem 4.1.** MixCorr *can be decided in linear time.*

▶ Remark. By using the "Euler–Poincaré lemma" [1] to count the uses of the Mix rule in a proof net, this also allows us to decide the correctness of a proof structure for MLL without Mix in linear time. Our decision procedure has the advantage of being simpler to describe than the previously known linear-time algorithms for MLL correctness [14, 22].

That said, this apparent simplicity is due to our use of the algorithm of Gabow et al. [11] as a black box. Looking inside the black box reveals, for instance, that it uses the *incremental tree set union* data structure of Gabow and Tarjan [12], which is also a crucial ingredient of the above-mentioned previous algorithms.

▶ Remark. This algorithm for UniquenessPM relies on the technique of *blossom shrinking* pioneered by Edmonds [9], a kind of graph contraction which may remind us of the *contractibility* correctness criterion [7] for MLL without Mix. Indeed, there exists a formal connection: a rewrite step of *big-step contractibility* [1] corresponds, when translated to graphifications, to contracting a blossom. However, not all blossoms are redexes for big-step contractibility.

4.2 Characterizing the sub-polynomial complexity

For MLL proof nets without Mix, correctness is known to be NL-complete under AC^0 reductions thanks to the Mogbil–Naurois criterion [17]. What about MLL+Mix? Since the reductions of §3 can be computed in constant depth, we have:

▶ **Theorem 4.2.** MixCorr *and* UniquenessPM *are equivalent under* AC^0 *reductions.*

Thus, it will suffice to study the complexity of UniquenessPM. Let us start with a positive result, using the parallel algorithms for finding a perfect matching mentioned in §2.1.

▶ **Proposition 4.3.** UniquenessPM *is in* randomized NC *and in* deterministic quasi-NC.

Proof. Let $G = (V, E)$ be a graph and M be a perfect matching of G. M is *not* unique if and only if, for some $e \in M$, the graph $G_e = (V, E \setminus \{e\})$ has a perfect matching. To test the uniqueness of M, run the $|M|$ parallel instances, one for each G_e, of a randomized NC [21] or deterministic quasi-NC [28] algorithm for deciding the existence of a perfect matching, and compute the disjunction of their answers in AC^0. ◀

Being in quasi-NC is a much weaker[7] result than being in NL. But as we shall now see, even showing that UniquenessPM is in NC (recall that NL $\subset$ NC) would be a major result. It would answer in the affirmative the following conjecture dating back from the 1980's:

▶ **Conjecture 4.4** (Lovász[8])**.** UniquePM *is in* NC.

Indeed, the following shows that UniquenessPM $\in$ NC $\Rightarrow$ UniquePM $\in$ NC (and the converse follows from the definitions).

▶ **Proposition 4.5.** *There is a* NC^2 *reduction from* UniquePM *to* UniquenessPM.

[7] In fact, one can show that NL $\subsetneq$ NSPACE$(O(\log^{3/2} n)) \subseteq$ quasi-NC^3, and the latter is where Svensson and Tarnawski's analysis puts finding a perfect matching.

[8] The conjecture is attributed to Lovász by a paper by Kozen et al. [19] which claims to solve it. But Hoang et al. [15] note that "this was later retracted in a personal communication by the authors". Still, the proposed solution works for bipartite graphs.

Proof. This is a consequence of a NC^2 algorithm by Rabin and Vazirani [25, §4] which, given a graph G, computes a set of edges M such that if G admits a unique perfect matching, then M is this matching. Starting from any graph G, run this algorithm and test whether its output is a perfect matching. If not, then G does not admit a unique perfect matching; if it is, then G is a positive instance of UniquePM if and only if (G, M) is a positive instance of UniquenessPM. ◀

To sum up these results about UniquenessPM, which apply to MixCorr:

▶ **Theorem 4.6.** MixCorr *is in randomized* NC *and in deterministic* quasi-NC*; it is in deterministic* NC *if and only if Conjecture 4.4 is true.*

5 Sequentializing MLL+Mix proof nets

In §4.1, we managed to solve MLL+Mix correctness in linear time, matching the known time complexity for MLL correctness. But the algorithms for MLL correctness still have an advantage: they can compute a sequentialization in linear time, whereas we only have a decision procedure for MixCorr which returns a yes/no answer[9]. We do not know how to compute MLL+Mix sequentializations in linear time. Nevertheless, by applying our bridge between proof nets and graph theory, we get the first *quasi-linear* time algorithm for MixSeq. The beginning of the next section will discuss why the problem seems harder with Mix.

Our algorithm proceeds in a "top-down" way: it starts by determining the root of the derivation tree and the link it introduces. To obtain the children of the root, it suffices to recurse on the connected components created by removing this link.

Furthermore, through the correspondence of Theorem 3.4, finding a link which is introduced by the last rule of some sequentialization amounts to finding a bridge in the matching of the graphification of the proof net (cf. §3.1). This is in fact a bit more convenient with graphifications than with general unique perfect matchings, thanks to the following property:

▶ **Lemma 5.1.** *All bridges in the graphification of some proof structure are matching edges.*

The algorithm will alternate between finding and deleting bridges; a deletion may cut cycles and thus create new bridges, which we want to detect without traversing the entire graph each time. To do so, we use a *dynamic bridge-finding data structure* designed for this kind of use case by Holm et al. [16]. It keeps an internal state corresponding to a graph, whose set of n vertices is immutable but whose set of edges may vary, and supports the following operations in $O((\log n)^2(\log\log n)^2)$ amortized time:

- updating the graph by inserting or deleting an edge;
- computing the number of vertices of the connected component of a given vertex;
- finding a bridge in the connected component of a given vertex;
- determining whether two vertices are in the same connected component.

▶ **Theorem 5.2.** MixSeq *can be solved in* $O(n(\log n)^2(\log\log n)^2)$ *time.*

Proof. Let π be a MLL+Mix proof net with n links, and $(G = (V, E), M)$ be its graphification. Both V and E are have cardinality $O(n)$ (in fact, $|V| = 2n$ and $|M| = n$).

The algorithm starts by initializing the bridge-finding data structure D with the graph G, computing the weakly connected components of π in linear time, and selecting a link in each

[9] It can find a feasible cycle, witnessing incorrectness, but cannot produce a certificate of correctness.

component. On each selected link l, we call the following recursive procedure; its role is to sequentialize the sub-proof net of π containing l whose graphification is a current connected component of G (G and D being mutable global variables):

- Let u be one endpoint of the matching edge corresponding to l. Using the bridge-finding structure, find a bridge $e = (v, w)$ in the component of u; necessarily, $e \in M$. Remove the edge e from G (and reflect this change on D with a deletion operation).
- If both v and w are isolated vertices, e corresponds to an `ax`-link and the entire sub-proof net consisted of this link. In this case, return a sequentialization with a single `ax`-rule.
- If one of v and w is isolated, and the other is not – by symmetry, let us assume the latter is v – then e corresponds to a $\parr$-link l'. Let p and p' be its predecessors.
  - Remove all edges incident to v.
  - If the matching edges corresponding to p and p' are in the same connected component of G, recurse on p, add a final $\parr$-link and return the resulting sequentialization.
  - If p and p' are in different connected components of G, recurse on p and p', use the results as the two premises of a Mix rule, add a final $\parr$-link and return the resulting sequentialization.
- If neither v nor w is isolated, e corresponds to a $\otimes$-link. This is handled similarly to the $\parr$+Mix case above.

Let us evaluate the time complexity. At each recursive call, one bridge is eliminated from G, so the number of recursive calls is n. The cost of each recursive call is $O(1)$ except for the updates and queries of the bridge-finding data structure. In total, there are $|E| = O(n)$ deletions, $|M| = n$ bridge queries, and at most n connectedness tests, and each of those takes $O((\log n)^2(\log\log n)^2)$ amortized time. Hence the $O(n(\log n)^2(\log\log n)^2)$ bound. ◀

▶ Remark. If we want to compute a sequentialization for a unique perfect matching, in general, a complication is the existence of bridges which are not in the matching.

Interestingly, one can determine whether a bridge e is in M *without looking at* M: it is the case if and only if both of the connected components created by removing e have an odd number of vertices. This leads to an algorithm for UniquePM; it is virtually the same as the one proposed by Gabow et al. [11, §2][10], from which we took our inspiration.

▶ Remark. One needs to use a sparse representation for derivation trees: the size of a fully written-out sequent calculus proof is, in general, not linear in the size of its proof net.

6 On the kingdom ordering of links

One may wonder if we could not have just tweaked an algorithm for MLL sequentialization into an algorithm for MixSeq. In order to argue to the contrary, let us briefly mention a difference between Bellin and van de Wiele's study of the sub-proof nets of MLL proof nets [5] and its extension to the MLL+Mix case by Bellin [4]. Any MLL sub-proof net of a MLL proof net may appear in the sequentialization of the latter; however, for MLL+Mix, Figure 7 serves as a counterexample: the sub-proof structure containing all links but the $\otimes$-link is correct for MLL+Mix, but it cannot be an intermediate step in a sequentialization of the entire proof net. A *normality* condition is needed to distinguish those sub-proof nets

[10] Not to be confused with their algorithm for UniquenessPM [11, §3] that we used in §4.1. They only claim a bound of $O(m\log^4 n)$ because the best dynamic 2-edge-connectivity data structure known at the time has operations in $O(\log^4 n)$ amortized time.

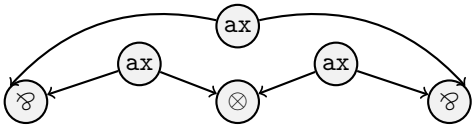


Figure 7 A MLL+Mix proof net which highlights a difficulty in solving MixSeq.

which may appear in a sequentialization, and this is why sequentialization algorithms which are morally based on a greedy parsing strategy, such as Guerrini's linear-time algorithm [14], do not adapt well to the presence of the Mix rule.

Any link l in a MLL+Mix proof net π admits a minimum normal sub-proof net of π containing l, its *kingdom* [4]. Bellin's *kingdom ordering* is the partial order on links corresponding to the inclusion between kingdoms. We give an algorithm to compute this order for any MLL+Mix proof net: this is yet another application of matching theory. It uses a characterization of the kingdom ordering in terms of a relation called *dependency* by Bagnol et al. [1] (who, in turn, take this name from the closely related *dependency graph* of Mogbil and Naurois [17]). We will also see how this dependency relation can be reformulated, through our correspondence between proof structures and perfect matchings, in terms of the *blossoms* mentioned in §2.1 and §4.1.

One may in fact define the kingdom ordering, written $\ll_\pi$, without reference to the notion of normal sub-proof net (we will not introduce the latter formally here):

▶ **Definition 6.1.** Let π be a MLL+Mix proof net. For any two links p, q of π, $p \ll_\pi q$ if and only if, in any sequentialization of π, the rule introducing q has, among its premises, a proof net containing p.

From this point of view, the kingdom ordering gives us information about the set of all sequentializations. Let us give some examples. The proof net of Figure 4b admits a unique sequentialization, so this directly gives us the kingdom ordering: for instance the middle $\otimes$-link is the greatest element. On the other hand, in the proof net of Figure 7, both $\parr$-links may be introduced by a last rule, so there is no greatest element. In fact, the kingdom ordering coincides with the predecessor relation. So it does not distinguish between the 3 terminal links even though, unlike the 2 others, the $\otimes$-link cannot be introduced last.

Before proceeding further, here is another property of MLL proof nets which is contradicted by Figure 7 for MLL+Mix proof nets, providing more evidence that MixSeq is trickier than MLL sequentialization.

▶ **Proposition 6.2.** *Let π be a MLL proof net and l be a maximal link for $\ll_\pi$. Then there exists a sequentialization of π whose last rule introduces l.*

6.1 Computing the kingdom ordering

▶ **Definition 6.3.** Let π be a proof structure. We write $D(\pi)$ for the *dependency relation* defined as follows: for any two links $p \neq q$ of π, *p is a dependency of q* when q is a $\parr$-link and there exists a feasible path between the predecessors of q going through p.

For instance, in the proof net of Figure 4b, the left $\parr$-link depends on the left $\otimes$-link, but not on the other $\otimes$-links or $\parr$-links; the middle $\otimes$-link has no dependency. In the case of Figure 7, the dependency relation is empty.

▶ **Theorem 6.4** (Bellin [4, Lemma 2][11]). *Let π be a* MLL+Mix *proof net. The transitive closure of $D(\pi) \cup S(\pi)$ is $\ll_\pi$, where $(p, q) \in S(\pi)$ means that p is a predecessor of q.*

The dependency relation can be computed by reduction to a matching problem *in the case of MLL+Mix proof nets*: even though it is well-defined in arbitrary proof structures, we need MLL+Mix correctness to compute it, because our matching algorithm relies on the absence of alternating cycles.

▶ **Lemma 6.5.** *Let G be a graph with a* unique *perfect matching M, and $e, f, g \in M$ be pairwise distinct edges. The existence of an alternating path starting with e, ending with f and crossing g can be reduced to the existence of a perfect matching.*

Proof. Let $(u, v) = e$, $(u', v') = f$ and $(a, b) = g$. Let G' be the graph obtained by adding new vertices s and s', new edges (s, u), (s, v), (s', u') and (s', v'), and by removing g. This graph admits a matching $M' = M \setminus \{g\}$ leaving the 4 vertices s, s', a and b unmatched.

Suppose G' admits a perfect matching M''. Then the symmetric difference $M' \triangle M''$ consists of two vertex-disjoint alternating paths for M', whose endpoints are $\{s, s', a, b\}$ by the same reasoning as Lemma 2.2. (In general, the symmetric difference may also contain alternating cycles, but if it were the case here, M would not be unique.)

We claim that these paths either go from s to a and b to s', or from s to b and a to s'. Otherwise, there would be an alternating path from a to b for M', and together with the matching edge g we removed earlier, this would give us an alternating cycle for M in G.

In both cases, let us join the two paths together by adding g, and remove the edges incident to s and s'. We get a path starting with e, ending with f, crossing g and alternating for M in G. Conversely, from such a path, one can get a perfect matching in G'. ◀

▶ **Theorem 6.6.** *Let π be a* MLL+Mix proof net *with a link p and a $\parr$-link q. Deciding whether $(p, q) \in D(\pi)$ can be done in linear time, in randomized* NC *and in* quasi-NC.

Proof. A degenerate case is when p is a predecessor of q: in this case, p depending on q is equivalent to π becoming incorrect if q is turned into a $\otimes$-link, and thus the complexity is the same as that of (the complement of) the correctness problem.

When p is not a predecessor of q, the definition of dependency translates into the problem defined in the above lemma by taking the graphification of π. Since the existence of a perfect matching can be decided in randomized NC or quasi-NC (cf. §2.1), so can our problem. To get a linear time complexity, we exploit the fact that we know a matching of G' leaving $O(1)$ vertices unmatched: a perfect matching can then be found with $O(1)$ iterations of an algorithm using *augmenting paths*, each iteration taking linear time, see e.g. [31, Chapter 9]. ◀

A transitive closure can be computed in polynomial time, and reachability in a directed graph can be decided in NL $\subset$ quasi-NC, so we get in the end:

▶ **Corollary 6.7.** *There are a polynomial-time algorithm and a* quasi-NC *algorithm to compute the kingdom ordering $\ll_\pi$ of any MLL+Mix proof net π.*

[11] This theorem was rediscovered in the special case of MLL proof nets by Bagnol et al. [1, Theorem 11], who refer to the kingdom ordering as the "order of introduction". We borrow the notations $D(\pi)$ and $S(\pi)$ from them.

6.2 Dependencies and blossoms in unique perfect matchings

We will now see how, through the correspondence of §3, Bellin's theorem can be rephrased as a statement on unique perfect matchings.

▶ **Definition 6.8.** Let G be a graph and M be a perfect matching of G. A *blossom* for M is a cycle whose vertices are all matched within the cycle, except for one, its *root*. The matching edge incident to the root, is called the *stem* of the blossom.

That is, a blossom consists of an alternating path between two vertices, starting and ending with a matching edge, together with a non-matching edge from the root to each of these two vertices; for instance, in Figure 1b, the two triangles are blossoms with a common stem. The stem of a blossom is not part of the cycle. Blossom are central to combinatorial matching algorithms, e.g. [9, 11], as we have previously mentioned.

▶ **Definition 6.9.** When $e \in M$ is in some blossom with stem $f \in M$, we write $e \to f$.

This is the graph-theoretical counterpart of the dependency relation, as is shown by the following two propositions.

▶ **Proposition 6.10.** *Let π be a MLL+Mix proof net and (G, M) be its graphification. Let p, q be links in π with corresponding matching edges $e_p, e_q \in M$. Then $e_p \to e_q$ if and only if p is a dependency of q or a predecessor of q, i.e. $(p, q) \in D(\pi) \cup S(\pi)$.*

▶ **Proposition 6.11.** *Let G be a graph, M be a perfect matching of G and π be the proofification of (G, M). Let $e, f \in M$ with corresponding $\otimes$-links $l_e, l_f \in M$. Then $e \to f$ if and only if l_e is a dependency of some $\parr$-link q from which l_f is reachable (by a directed path).*

▶ Remark. In Proposition 6.10, the "if" direction holds even for incorrect proof structures; in Proposition 6.11, note that no uniqueness property is required of the perfect matching.

Thus, we see that Bellin's theorem is equivalent to the following theorem where $\to^+$ is the transitive closure of $\to$. As far as we know, this is a new result in graph theory.

▶ **Theorem 6.12.** *Let G be a graph with a* unique *perfect matching M, and $e, f \in M$. The edge e occurs before f in all sequentializations for M if and only if $e \to^+ f$.*

We can also formulate the theorem without mentioning sequentializations. Fix an edge $e \in M$, and iteratively remove the endpoints of any bridge in M, except e, until we end up with a vertex-induced subgraph of G where no bridge remains except e. This is the graph-theoretic analogue of the *kingdom* of e. Bellin's theorem says that for any edge f in the kingdom of e, $f \to^+ e$, that is:

▶ **Theorem 6.13.** *Let G be a graph with a* unique *perfect matching M. Suppose that M contains only one bridge e. Then for all $f \in M \setminus \{e\}$, $f \to^+ e$.*

This is the case in Figure 1b: the middle edge e is the only bridge, and it is the stem of the two triangular blossoms which contain the other matching edges.

The graph-theoretic versions are somewhat simpler to state than the original theorem: one takes the transitive closure of a single relation, instead of a union of two unrelated relations. We give a direct proof of the last formulation in Appendix D, based on the *blossom shrinking* operation mentioned in §4.1.

7 Conclusion

We have presented a correspondence between proof nets and perfect matchings, and demonstrated its usefulness through several applications of graph theory to linear logic: our results give the best known complexity for MLL+Mix correctness and sequentialization, by taking advantage of sophisticated graph algorithms. We also expect linear logic to eventually lead to new results in graph theory through this correspondence – in fact, the rephrasing of Bellin's theorem in the last section is one such example – and in general, we hope to see fruitful interactions arise between those two domains.

Perspectives. Now that we have shed a new light on MLL+Mix proof nets, it would be interesting to revisit the well-studied theory of MLL proof nets. Therefore, we would like to find the right graph-theoretical counterpart to the connectedness condition in the Danos–Regnier criterion for MLL, but unique perfect matchings do not seem to be the right setting to do so. Indeed, there are many objects in graph theory which admit "structure from acyclicity" theorems equivalent to Kotzig's theorem on unique perfect matchings (cf. [30]) – that is, to MLL+Mix sequentialization – and some of these may be better suited to go beyond MLL+Mix proof nets and extend the correspondence, either to MLL or to larger fragments of linear logic.

In particular, we have already taken inspiration from *edge-colored graphs* (see e.g. [2, §16]), which are rather close to the usual paired graphs, to prove the coNP-hardness of Pagani's *visible acyclicity* condition [24] on MELL proof structures (cf. the workshop abstract [23]). Let us also mention that we have found an interpretation of Retoré's construction [27] in terms of graphs with *forbidden transitions* [29], which can be seen as the generalization of paired graphs by dropping the disjointness condition.

A closely related question is that of finding a natural graph-theoretic decision problem equivalent to correctness for MLL without Mix through low-complexity reductions – hopefully computable both in linear time and in AC^0, like the equivalence between MixSeq and UniquenessPM exhibited in this paper. Though both Murawski & Ong [22] and Mogbil & Naurois [17] reduce MLL correctness to problems on directed graphs, the complexity of these reductions is higher than we would like: the first is not computed in logarithmic space, the second uses a subroutine for undirected connectivity, a L-complete problem whose membership in L is highly non-trivial. An answer to this question may help clarify why all known linear-time correctness criteria for MLL rely on the same sophisiticated data structure, as mentioned in §4.1.

References

1 Marc Bagnol, Amina Doumane, and Alexis Saurin. On the dependencies of logical rules. In *FOSSACS, 18th International Conference on Foundations of Software Science and Computation Structures*, 2015.

2 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs. Theory, algorithms and applications. 2nd ed.* London: Springer, 2nd ed. edition, 2009.

3 David A. Mix Barrington. Quasipolynomial size circuit classes. In *[1992] Proceedings of the Seventh Annual Structure in Complexity Theory Conference*, pages 86–93, 1992.

4 Gianluigi Bellin. Subnets of proof-nets in multiplicative linear logic with MIX. *Mathematical Structures in Computer Science*, 7(6):663–669, 1997.

5 Gianluigi Bellin and Jacques van de Wiele. Subnets of Proof-nets in MLL-. In *Proceedings of the Workshop on Advances in Linear Logic*, pages 249–270, New York, NY, USA, 1995. Cambridge University Press.

6 Ashok K. Chandra, Larry Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

7 Vincent Danos. *La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du Lambda-calcul)*. PhD thesis, Université Paris-Diderot – Paris VII, 1990.

8 Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3):181–203, 1989.

9 Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(0):449–467, 1965.

10 Arnaud Fleury and Christian Retoré. The mix rule. *Mathematical Structures in Computer Science*, 4(2):273–285, 1994.

11 Harold N. Gabow, Haim Kaplan, and Robert E. Tarjan. Unique maximum matching algorithms. *Journal of Algorithms*, 40(2):159–183, 2001.

12 Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences*, 30(2):209–221, apr 1985.

13 Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, jan 1987.

14 Stefano Guerrini. A linear algorithm for MLL proof net correctness and sequentialization. *Theoretical Computer Science*, 412(20):1958–1978, apr 2011.

15 Thanh Minh Hoang, Meena Mahajan, and Thomas Thierauf. On the bipartite unique perfect matching problem. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I*, pages 453–464, 2006.

16 Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Dynamic bridge-finding in $\tilde{O}(\log^2 n)$ amortized time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, Proceedings, pages 35–52. Society for Industrial and Applied Mathematics, jan 2018.

17 Paulin Jacobé de Naurois and Virgile Mogbil. Correctness of linear logic proof structures is NL-complete. *Theoretical Computer Science*, 412(20):1941–1957, apr 2011.

18 Anton Kotzig. Z teórie konečných grafov s lineárnym faktorom. II. *Matematicko-fyzikálny časopis*, 09(3):136–159, 1959.

19 Dexter Kozen, Umesh V. Vazirani, and Vijay V. Vazirani. NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching. In *Foundations of Software Technology and Theoretical Computer Science, Fifth Conference, New Delhi, India, December 16-18, 1985, Proceedings*, pages 496–503, 1985.

20 László Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory*, pages 565–574, 1979.

21 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

22 Andrzej S. Murawski and C.-H. Luke Ong. Fast verification of MLL proof nets via IMLL. *ACM Transactions on Computational Logic*, 7(3):473–498, 2006.

23 Lê Thành Dũng Nguyễn. On the complexity of finding cycles in proof nets. Extended abstract for the workshop Developments in Implicit Computational Complexity 2018.

24 Michele Pagani. Visible acyclic differential nets, Part I: Semantics. *Annals of Pure and Applied Logic*, 163(3):238–265, 2012.

25 Michael O. Rabin and Vijay V. Vazirani. Maximum matchings in general graphs through randomization. *Journal of Algorithms*, 10(4):557–567, dec 1989.

26 Christian Retoré. Handsome proof-nets: R&B-graphs, perfect matchings and series-parallel graphs. Research Report, INRIA, 1999.

27 Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, 2003.

28 Ola Svensson and Jakub Tarnawski. The matching problem in general graphs is in quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017.

29 Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003.

30 Stefan Szeider. On theorems equivalent with Kotzig's result on graphs with unique 1-factors. *Ars Combinatoria*, 73:53–64, 2004.

31 Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1983.

A Proofs of §3

Proof of Proposition 3.2. By negating the two sides of the equivalence, the goal becomes proving that a proof structure π contains a feasible cycle if and only if its graphification (G, M) contains an alternating cycle.

Consider any alternating cycle for M in G of length $2n$, and take the $\mathbb{Z}/(n)$-indexed sequence of vertices corresponding to the matching edges in the cycle. By construction of the graphification, if two edges in M are incident to a common non-matching edge, then the corresponding links in π are adjacent: thus, in our sequence, each vertex is adjacent to the previous and the next one, and thus we have a cycle. If it were not feasible, it would contain three consecutive links p, q, r with q a $\parr$-link and p, r its predecessors[12]; but then the alternating cycle would have to cross two incident non-matching edges (from p to q and from q to r), which is impossible. Thus, π contains a feasible cycle.

To show the converse we will exhibit a right inverse to the map from alternating cycles to feasible cycles defined above. Consider a feasible cycle: it can be partitioned into directed paths from `ax`-links to $\otimes$-links. Let l be an intermediate link in such a path, and e, p, s be matching edges corresponding respectively to l, its predecessor, and its successor in the directed path. s has a unique endpoint u which is incident to both endpoints of e; e has a unique endpoint v which is *not* incident to both endpoints of p. To join e with s, we use the edge (u, v). By taking all these non-matching edges for all maximal directed paths in the cycle, as well as a choice of two edges incident to each matching edge corresponding to an `ax`-link, and the matching edges (a_l, b_l) corresponding to all the links l in the cycle, we get an alternating cycle. ◀

Proof of Lemma 3.3. Suppose for contradiction that l is not a terminal link, and let l' be a successor of l. Then for some endpoint v of $(a_{l'}, b_{l'})$, (a_l, v) and (b_l, v) are both edges in G, and they make up a path between a_l and b_l not going through (a_l, b_l). Thus, (a_l, b_l) cannot be a bridge.

The fact that (a_l, b_l) is a bridge means that by removing this edge, a_l and b_l are in different connected components; if l is a $\otimes$-link, each of these connected components contain the matching edge corresponding to one premise of l. ◀

[12] To expand on this point: this is because we have prohibited vertex repetitions in our definition of cycles. This is legitimate since a graph is a forest if and only if it does not contain a non-vertex-repeating cycle.

Proof of Theorem 3.4. We convert a sequentialization S of (G, M) into a sequentialization Σ of π inductively as follows. Since $G \neq \emptyset$, the last rule of S is either a disjoint union or the introduction of a bridge $e = (a_l, b_l) \in M$ by joining together (G_a, M_a) and (G_b, M_b) with respective sequentializations S_a and S_b. In the latter case, l is a terminal link of π.

- If $G_a = G_b = \emptyset$, then l is an $\mathtt{ax}$-link, and Σ consists of a single $\mathtt{ax}$-rule.
- If $G_a \neq \emptyset$ and $G_b = \emptyset$, then l is a $\parr$-link, and the removal of l from π yields a proof structure π' whose graphification is (G_a, M_a). Σ then consists of a $\parr$-rule introducing l applied to the sequentialization of π' corresponding to S_a.
- If $G_a \neq \emptyset$ and $G_b \neq \emptyset$, then l is a $\otimes$-link. Since e is a bridge, the removal of l from π yields two proof structures π_a and π_b whose respective graphifications are (G_a, M_a) and (G_b, M_b). Σ then consists of an $\otimes$-rule applied to the translations of S_a and S_b.

If the last rule of S is a disjoint union rule, it is translated into a Mix rule in Σ.

The bijectivity can be proven by defining the inverse transformation and by checking that it is indeed its inverse. ◀

Proof of Proposition 3.6. Let π be the proofification of (G, M). Any feasible cycle in π changes direction only at $\mathtt{ax}$-links and $\otimes$-links, and therefore can be partitioned into an alternation of $\otimes$-links, corresponding to matching edges, and of paths starting with some B_u, ending with some B_v and crossing some $\mathtt{ax}_e$, corresponding to non-matching edges $e = (u, v)$. Therefore, it corresponds to an alternating cycle for M, and the mapping defined this way is bijective. ◀

Proof of Proposition 3.7. This follows from the fact that a $\otimes$-link may be introduced by the last rule of a sequentialization if and only if it is splitting, i.e. its removal disconnects its two precedessors. ◀

B Omitted proof in §5

Proof of Lemma 5.1. Let e be a non-matching edge. Then there are matching edges (u, v) and (s, t) such that the link corresponding to (u, v) is the predecessor of the one for (s, t), and $e = (u, s)$. The non-matching edge (v, s) is then also present in the graph, and so e cannot be a bridge. ◀

C Omitted proofs in §6

Proof of Proposition 6.2. If l is a terminal $\parr$-link, no other assumption is needed for the existence of such a sequentialization. Else, l is a terminal $\otimes$-link and it suffices to show that l is *splitting*, i.e. that the removal of l splits π into two connected components.

Suppose that it is not the case, and consider some sequentialization of π: it must contain a $\parr$-rule, applied to a sub-proof net π' for which l is splitting, which turns it into a sub-proof net for which l is not splitting anymore. Let p be the $\parr$-link introduced by that rule; its predecessors lie in different connected components of $\pi' \setminus \{l\}$. Since π' is a MLL proof net, the predecessors of p are connected by a feasible path in π', which must cross l. This shows that l is a dependency of p in the sense of Definition 6.3, contradicting the maximality of l. (This only uses the fact that $D(\pi) \subseteq \ll_\pi$, which is the "easy" part of Bellin's theorem.) ◀

Proof of Proposition 6.10. If $(p, q) \in S(\pi)$, then by construction there exists a blossom of length 3 containing p with stem q. If $(p, q) \in D(\pi)$, then for the same reason as Proposition 3.2, we can get, from the feasible path between the predecessors of q visiting p, an alternating path for M starting and ending with the edges corresponding to those predecessors and crossing

the edge corresponding to p. By adding two non-matching edges to the same endpoint of the matching edge for q, we get a blossom with stem q.

Conversely, let q be a link, e the corresponding matching edge, and B be a blossom with stem q. Let us first note that if B contains an non-matching edge joining e with the matching edge corresponding to a successor of q, then by replacing this non-matching edge with its twin incident to the other endpoint of q, we get an alternating cycle; this is impossible because we have assumed π to be a MLL+Mix proof net. Therefore, the first and last matching edges in B are both premises of q. If they are the same – that is, if B has length 3 and contains a single matching edge – then this edge corresponds to a predecessor p of q. Otherwise, B gives an alternating path between two distinct premises of q; necessarily q is a $\parr$-link (otherwise, there would be an alternating cycle), and all links corresponding to matching edges in B are dependencies of q. ◀

Proof of Proposition 6.11. Let B be a blossom with stem f, whose two non-matching edges incident to f are a and b. B translates into a feasible path between $\mathtt{ax}_a$ and $\mathtt{ax}_b$ in π. Now, $\mathtt{ax}_a$ and $\mathtt{ax}_b$ are also leaves of a binary tree of $\parr$-links whose root has the single successor l_f; by taking q to be the lowest common ancestor of $\mathtt{ax}_a$ and $\mathtt{ax}_b$ in this tree, l_f is reachable from q, and every link in the path between $\mathtt{ax}_a$ and $\mathtt{ax}_b$ depends on q. Conversely, any feasible path between the two premises of a $\parr$-link corresponds to a blossom for M in G. ◀

D Proof of the graph-theoretic Bellin theorem (Theorem 6.13)

Let G be a graph with a unique perfect matching M, containing a single bridge e. Removing the edge e, *but not its endpoints*, results in two connected components which both have a unique *near-perfect matching* (leaving one vertex unmatched) containing no bridge. If both these components have a single vertex, then the theorem is vacuously true; else, we have reduced it to the following proposition, where $a \leftarrow b$ means that $b \to a$, $\leftarrow^*$ is the reflexive transitive closure of $\leftarrow$, and $u \twoheadleftarrow f$ means that f is contained in a blossom with root u.

▶ **Proposition.** *Let G be a graph with a near-perfect matching M and let u be the unmatched vertex. Suppose G has no bridge in M and no alternating cycle for M. Then for all $f \in M$, there exists $g \in M$ such that $u \twoheadleftarrow g \leftarrow^* f$.*

The proof of this proposition relies on the *blossom shrinking* operation: starting from the graph G with a matching M, this consists in taking the quotient graph G' where all the vertices of the blossom have been identified; M induces a matching M' in G'.

▶ **Lemma.** *Under the hypotheses of the proposition, if $M \neq \emptyset$, then:*

1. *There exists a blossom in G for M with root u.*
2. *Let G' be the graph obtained by shrinking this blossom, with induced matching $M' \subset M$. There is no bridge in M' and no alternating cycle for M'.*
3. *Let u' be the exposed vertex in G', corresponding to the shrunk blossom. For all $f \in M'$ with $u' \twoheadleftarrow f$ in G', there exists $g \in M$ such that $u \twoheadleftarrow g \leftarrow^* f$ in G.*

Proof of (1). The absence of alternating cycle amounts to saying that M is the unique perfect matching of $G[V \setminus \{u\}]$ where V is the vertex set of G. (Note that $M \neq \emptyset \Leftrightarrow V \setminus \{u\} \neq \emptyset$.) By Kotzig's theorem, M contains a bridge e of $G[V \setminus \{u\}]$; let V_1 and V_2 be the connected components created by the removal of e (but keeping its endpoints) from $G[V \setminus \{u\}]$. We create a new graph H by starting from $G[V \setminus \{u\}]$, adding two new vertices u_1 and u_2, and adding the edges (u_i, v) for all $v \in V_i$ with v adjacent to u in G ($i = 1, 2$), and the edge (u_1, u_2).

The perfect matching $M \cup \{(u_1, u_2)\}$ of H contains no bridge of H: since e is not a bridge of G, there is at least one edge between u_1 and V_1 and one edge between u_2 and V_2, so (u_1, u_2) is not a bridge in H; and any edge of M would be a bridge of G if it were a bridge of H. Let us apply Kotzig's theorem again: this perfect matching admits an alternating cycle, which cannot be contained in $H[V \setminus \{u\}] = G[V \setminus \{u\}]$. Therefore, it contains an alternating path from u_1 to u_2, from which we retrieve a blossom with root u in G. ◀

Proof of (2). If there existed a bridge $e \in M'$ of G', then $G' \setminus \{e\}$ would be disconnected while $G \setminus \{e\}$ would be connected; this is impossible. An alternating cycle for M' would not visit u' because it is unmatched, and therefore would be an alternating cycle for M in G. ◀

Proof of (3). Let B be the blossom with root u in G that has been shrunk, and B' be the blossom with root u' in G' containing f. There are two non-matching edges e'_1 and e'_2 in B' incident to u'; let $e_1 = (u_1, v_1)$ be a preimage of e'_1 and $e_2 = (u_2, v_2)$ be a preimage of e'_2 in G, with $u_1, u_2 \in B$.

The blossom B can be decomposed into $P_1 \cup Q \cup P_2$, where P_1 is an alternating path from u to u_1 (possibly empty, if $u = u_1$), Q is an alternating path from u_1 to u_2 (possibly empty, if $u_1 = u_2$), and P_2 is an alternating path from u_2 to u. As for B', it lifts to an alternating path R between u_1 and u_2 starting and ending with a non-matching edge, so that $|R|$ is odd and $f \in R$. We proceed by case analysis on the parity of $|P_1|$ and $|P_2|$.

- If they are both even, then $P_1 \cup R \cup P_2$ is a blossom: $u \twoheadleftarrow f \leftarrow^* f$.
- If $|P_1|$ is even and $|P_2|$ is odd, then $Q \cup R$ is a blossom with root u_1. Either $u_1 = u$ and then $u \twoheadleftarrow f$, or there is an edge $g \in B \cap M$ incident to u_1 and then $u \twoheadleftarrow g \leftarrow f$.
- The case $|P_1|$ even and $|P_2|$ odd is symmetric to the previous one.
- If they were both odd, $Q \cup R$ would be an alternating cycle. ◀

Proof of the proposition. By induction on the size of G.

Let us take a blossom using lemma (1). If it contains f, then $u \twoheadleftarrow f$ and we are done. Else, we shrink the blossom and get G', M' and u'; by lemma (2), they satisfy the assumptions of the proposition. By the induction hypothesis, there exists g such that $u' \twoheadleftarrow g \leftarrow^* f$ in G'. Thanks to lemma (3), $u' \twoheadleftarrow g$ entails $u \twoheadleftarrow h \leftarrow^* g$ in G for some $h \in M$. Also, $g \leftarrow^* f$ in G' entails $g \leftarrow^* f$ in G because the (possibly empty) sequence of blossoms which binds f to g in G' cannot contain the vertex u', and therefore lifts to exactly the same edges in G. Thus, $u \twoheadleftarrow h \leftarrow^* g \leftarrow^* f$ and therefore $u \twoheadleftarrow h \leftarrow^* f$ in G. ◀