# Commutative Algorithms Approximate the LLL-distribution

**Fotis Iliopoulos**[1]
University of California Berkeley, USA
fotis.iliopoulos@berkeley.edu
https://orcid.org/0000-0002-1825-0097

## Abstract

Following the groundbreaking Moser-Tardos algorithm for the Lovász Local Lemma (LLL), a series of works have exploited a key ingredient of the original analysis, the witness tree lemma, in order to: derive deterministic, parallel and distributed algorithms for the LLL, to estimate the entropy of the output distribution, to partially avoid bad events, to deal with super-polynomially many bad events, and even to devise new algorithmic frameworks. Meanwhile, a parallel line of work has established tools for analyzing stochastic local search algorithms motivated by the LLL that do not fall within the Moser-Tardos framework. Unfortunately, the aforementioned results do not transfer to these more general settings. Mainly, this is because the witness tree lemma, provably, does not longer hold. Here we prove that for commutative algorithms, a class recently introduced by Kolmogorov and which captures the vast majority of LLL applications, the witness tree lemma does hold. Armed with this fact, we extend the main result of Haeupler, Saha, and Srinivasan to commutative algorithms, establishing that the output of such algorithms well-approximates the *LLL-distribution*, i.e., the distribution obtained by conditioning on all bad events being avoided, and give several new applications. For example, we show that the recent algorithm of Molloy for list coloring number of sparse, triangle-free graphs can output exponential many list colorings of the input graph.

**2012 ACM Subject Classification** Mathematics of computing → Probabilistic algorithms

**Keywords and phrases** Lovasz Local Lemma, Local Search, Commutativity, LLL-distribution, Coloring Triangle-free Graphs

**Digital Object Identifier** 10.4230/LIPIcs.APPROX-RANDOM.2018.44

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1704.02796`.

**Acknowledgements** The author is grateful to Dimitris Achlioptas and Alistair Sinclair for detailed comments and feedback, as well as to anonymous reviewers for comments and remarks.

## 1 Introduction

Many problems in combinatorics and computer science can be phrased as finding an object that lacks certain bad properties, or "flaws". In this paper we study algorithms that take as input a flawed object and try to remove all flaws by transforming the object through repeated probabilistic action.

Concretely, let $\Omega$ be a set of objects and let $F = \{f_1, f_2, \ldots, f_m\}$ be a collection of subsets of $\Omega$. We will refer to each $f_i \in F$ as a *flaw* to express that its elements share some negative

[1] Research supported by NSF grant CCF-1514434 and the Onassis Foundation

feature. For example, if a CNF formula $F$ on $n$ variables has clauses $c_1, c_2, \ldots, c_m$, we can define for each clause $c_i$ the flaw (subcube) $f_i \subseteq \{0,1\}^n$ whose elements violate $c_i$. Following linguistic rather than mathematical convention we say that $f$ is present in $\sigma$ if $f \ni \sigma$ and that $\sigma \in \Omega$ is *flawless* (perfect) if no flaw is present in $\sigma$.

To prove the *existence* of flawless objects we can often use the Probabilistic Method. As a matter of fact, in many interesting cases, this is the only way we know how to do so. To employ the Probabilistic Method, we introduce a probability measure $\mu$ over $\Omega$ and consider the collection of "bad" events corresponding to flaws. If we are able to show that the probability to avoid all bad events is strictly positive, then this implies the existence of a flawless object. A trivial example is the case where all the bad events are independent of one another and none of them has probability one. One of the most powerful tools of the Probabilistic Method is the Lovász Local Lemma [13] which weakens the latter restrictive condition of independence to a condition of limited dependence.

Making the LLL constructive was the study of intensive research for over two decades [5, 4, 27, 12, 33]. The breakthrough was made by Moser [28] who gave a very simple algorithm that finds a satisfying assignment of a $k$-CNF formula, under conditions that nearly match the LLL condition for satisfiability. Very shortly afterwards, Moser and Tardos [29] made the general LLL constructive for any *product* probability measure over explicitly presented variables. Specifically, they proved that whenever the general LLL conditon holds, the *Resample* algorithm, which repeatedly selects *any* occurring bad event and resamples all its variables according to the measure, i.e., independently, quickly converges to a flawless object.

The first result that made the LLL constructive in a non-product probability space was due to Harris and Srinivasan in [19], who considered the space of permutations endowed with the uniform measure. Subsequent works by Achlioptas and Iliopoulos [2, 1] introducing the flaws/actions framework, and of Harvey and Vondrák [21] introducing the resampling oracles framework, made the LLL constructive in more general settings. These frameworks [2, 21, 1] provide tools for analyzing *focused* stochastic search algorithms [30], i.e., algorithms which, like *Resample*, search by repeatedly selecting a flaw of the current state and moving to a random nearby state that avoids it, in the hope that, more often than not, more flaws are removed than introduced, so that a flawless object is eventually reached. At this point, all LLL applications we are aware of have efficient algorithms analyzable in these frameworks.

Besides conditions for existence and fast convergence to perfect objects, one could ask further questions regarding properties of focused search algorithms. For instance, "are they parallelizable ?", "how many solutions can they output?", "what is the expected "weight" of a solution?", etc. These questions and more have been answered for the Moser-Tardos algorithm in a long series of work [29, 15, 20, 22, 8, 11, 14, 17]. As a prominent example, the result of Haeupler, Saha and Srinivasan [15], as well as follow-up works of Harris and Srinivasan [20, 16], allow one to argue about the dynamics of the MT process, resulting in several new applications such as estimating the entropy of the output distribution, partially avoiding bad events, dealing with super-polynomially many bad events, and even new frameworks [18, 9].

Unfortunately, most of these follow-up results that further enhance, or exploit, our understanding of the MT process are not transferable to the general settings of [2, 21, 1]. Mainly, this is because a key and elegant technical result of the original analysis of Moser and Tardos, *the witness tree lemma*, does not longer hold under the most general assumptions [21]. Roughly, it states that any tree of bad events growing backwards in time from a certain root bad event $A_i$, with the children of each node $A_j$ being bad events that are adjacent to $A_j$ in the dependency graph, has probability of being consistent with the trajectory of the algorithm that is bounded by the product of the probabilities of all events in this tree. The witness tree lemma and its variations [22, 14] has been used for several other purposes besides those already mentioned, such as designing deterministic, parallel and distributed algorithms for the LLL [29, 8, 11, 14, 17].

On the other hand, Harris and Srinivasan [19] do manage to prove the witness tree lemma for their algorithm for the LLL on the space of permutations, via an analysis that is tailored specifically to this setting. Although their proof does not seem to be easily generalizable to general spaces, their success makes it natural to ask if we can impose mild assumptions in the general settings of [2, 21, 1] under which the witness tree lemma (and most of its byproducts) can be established.

The main contribution of this paper is to answer this question positively by showing that it is possible to prove the witness tree lemma in the *commutative setting*. The latter was introduced by Kolmogorov [24], who showed that under its assumptions one can obtain parallel algorithms, as well as the flexibility of having arbitrary flaw choice strategy in the frameworks of [2, 21, 1]. We note that the commutative setting captures the vast majority of LLL applications, including but not limited to both the variable and the permutation settings.

Subsequently to the present work, Achlioptas, Iliopoulos and Sinclair [3] gave a simpler proof of the witness tree lemma under a more general notion of commutativity (essentially matrix commutativity) at the mild cost of slightly restricting the family of flaw choice strategies (as we will see, in this paper the flaw choice strategy can be arbitrary).

Armed with the witness tree lemma, we are able to study properties of algorithms in the commutative setting and give several applications.

## Distributional Properties

As already mentioned, one of the most important applications of the witness tree lemma is given in the paper of Haeupler, Saha and Srinivasan [15], who study properties of the *MT-distribution*, the output distribution of the MT algorithm. Their main result is that the MT-distribution well-approximates the *LLL-distribution*, i.e., the distribution obtained by conditioning on all bad events being avoided. As an example, an immediate consequence of this fact is that one can argue about the expected *weight* of the output of the MT algorithm, given a weighting function over the space $\Omega$. Furthermore, as shown in the same paper [15] and follow-up papers by Harris and Srinivasan [20, 16], one can lower bound the entropy of the MT distribution, go beyond the LLL conditions (if one is willing to only partially avoid bad events), and deal with applications with super-polynomially many number of bad events.

Here we extend the result of [15] to the commutative setting: Given a commutative algorithm that is perfectly compatible with the underlying probability measure, its output well-approximates the LLL distribution in the same sense the MT-distribution does in the variable setting. For arbitrary commutative algorithms, the quality of the approximation additionally depends on the compatibility of the algorithm with the measure on the event(s) of interest. A simplified, and imprecise, version of our main theorem, which assumes that the initial state of the algorithm is sampled according to the underlying probability distribution $\mu$, is as follows. The formal statement of our main theorem can be found in Section 3.

▶ **Theorem 1** (Informal and Imprecise Statement). *If algorithm $\mathcal{A}$ is commutative and the algorithmic LLL conditions hold then, for each $E \subseteq \Omega$,*

$$\Pr[E] \le \gamma(E)\left(1+\frac{1}{d}\right)^{D_E} ,$$

*where $E$ is independent of all but at most $D_E$ flaws, $d$ is the maximum degree of the dependency graph, $\gamma(E) \geq \mu(E)$ is a measure of the "compatibility" between $\mathcal{A}$ and the underlying probability distribution $\mu$ at $E$, and* $\Pr[E]$ *is the probability that $\mathcal{A}$ ever reaches $E$ during its execution.*

Moreover, we quantitatively improve the bounds of [15] under the weaker assumptions of *Shearer's condition* [32], i.e., the most general LLL criterion under the assumption that the dependency graph is undirected. This allows us to study distributional properties of commutative algorithms using criteria that lie between the General LLL and Shearer's condition such as the Clique LLL [23].

## Algorithmic LLL Without a Slack and Arbitrary Flaw Choice Strategy

The works of Achlioptas, Iliopoulos and Kolmogorov [2, 1, 24] require a multiplicative slack in the generalized LLL conditions in order to establish fast convergence to a perfect object. On the other hand, Harvey and Vondrák [21] dispense with this requirement in the important case of algorithms that are perfectly compatible with the underlying measure under the mild assumption that the dependency graph is undirected.

Using the witness lemma, we are able to dispense with the multiplicative slack requirement for arbitrary algorithms in the commutative setting and also have the flexibility of arbitrary flaw choice strategy, as in the result of Kolmogorov [24].

## Improved Running Time Bounds

We are able to improve the running time bounds of Harvey and Vondrák [21] for commutative algorithms, matching those of Kolipaka and Szegedy [22] for the MT algorithm. Whether this could be done was left as an open question in [21]. We note that while the results of Achlioptas, Iliopoulos and Kolmogorov [2, 1, 24] also manage to give improved running time bounds they require a multiplicative-slack in the LLL conditions.

## Concrete Applications

In the full version of the paper we give concrete applications of commutative algorithms showing new results for the problems of *rainbow matchings, list-coloring* and *acyclic edge coloring*. Each application is chosen so that it demonstrates specific features of our results. Perhaps the most interesting one, which we include in the present version, is to show that the algorithm of Molloy [26] for finding list-colorings in triangle-free graphs with maximum degree $\Delta$ using $(1+\epsilon)\frac{\Delta}{\ln \Delta}$ colors, can actually output exponentially many such colorings with positive probability. First, we show that Molloy's algorithm can be analyzed in the general frameworks of the algorithmic LLL and that it is commutative, a fact that gives us access to properties of its output distribution. Then, we apply results regarding the entropy of the output of commutative algorithms. We show the following theorem.

▶ **Theorem 2.** *For every $\epsilon > 0$ there exists $\Delta_\epsilon$ such that every triangle-free graph $G$ with maximum degree $\Delta \geq \Delta_\epsilon$ has list chromatic number $\chi_\ell(G) \leq (1+\epsilon)\frac{\Delta}{\ln \Delta}$. Moreover, there exists an algorithm $\mathcal{A}$ that takes $G$ as input and list-colors it in expected polynomial time. In addition, $\mathcal{A}$ is able to output $\mathrm{e}^{cn}$ distinct list colorings with positive probability, where $c > 0$ is a constant that depends on $\epsilon$ and $\Delta$.*

We emphasize that the algorithm of Molloy is a sophisticated stochastic local search algorithm whose analysis is far from any standard LLL setting. The fact that our results allow us to state non-trivial facts about its distributional properties almost in a black-box fashion is testament to their flexibility.

## 2 Background and Preliminaries

In this section we present the necessary background and definitions to describe our setting. In Subsection 2.1 we describe the Lovász Local Lemma. In Subsections 2.2 and 2.3 we formally outline the algorithmic assumptions of [2, 21, 1, 24]. In Subsection 2.4 we describe improved Lovász Local Lemma criteria formulated in our setting.

### 2.1 The Lovász Local Lemma

To prove the *existence* of flawless objects we can often use the Probabilistic Method. To do so, we introduce a probability measure $\mu$ over $\Omega$ and consider the collection of "bad" events corresponding to flaws. If we are able to show that the probability to avoid all bad events is strictly positive, then this implies the existence of a flawless object. One of the most powerful tools to establish the latter is the Lovász Local Lemma [13].

▶ **General LLL.** *Let $(\Omega, \mu)$ be a probability space and $\mathcal{A} = \{A_1, A_2, \ldots, A_m\}$ be a set of $m$ (bad) events. For each $i \in [m]$, let $D(i) \subseteq [m] \setminus \{i\}$ be such that $\mu(A_i \mid \cap_{j \in S} \overline{A_j}) = \mu(A_i)$ for every $S \subseteq [m] \setminus (D(i) \cup \{i\})$. If there exist positive real numbers $\{\psi_i\}_{i=1}^{m}$ such that for all $i \in [m]$,*

$$\frac{\mu(A_i)}{\psi_i} \sum_{S \subseteq D(i) \cup \{i\}} \prod_{j \in S} \psi_j \leq 1 \ , \tag{1}$$

*then the probability that none of the events in $\mathcal{A}$ occurs is at least $\prod_{i=1}^{m} 1/(1+\psi_i) > 0$.*

▶ Remark. Condition (1) above is equivalent to the more well-known form $\mu(A_i) \leq x_i \prod_{j \in D(i)}(1 - x_j)$, where $x_i = \psi_i/(1+\psi_i)$. As we will see, formulation (1) facilitates refinements.

Let $G$ be the digraph over the vertex set $[m]$ with an edge from each $i \in [m]$ to each element of $D(i) \cup \{i\}$. We call such a graph a *dependency* graph. Therefore, at a high level, the LLL states that if there exists a sparse dependency graph and each bad event is not too likely, then perfect objects exist.

### 2.2 Algorithmic Framework

Here we describe the class of algorithms we will consider as well as the algorithmic LLL criteria for fast convergence to a perfect object. Since we will be interested in algorithms that search for perfect objects, we sometimes refer to $\Omega$ as a state space and to its elements as states.

For a state $\sigma$, we denote by $U(\sigma) = \{j \in [m] \text{ s.t. } f_j \ni \sigma\}$ the set of indices of flaws that are present at $\sigma$. We consider algorithms which at each flawed state $\sigma$ choose an element of $U(\sigma)$ and randomly move to a nearby state in an effort to *address* the corresponding flaw. Concretely, we will assume that for every flaw $f_i$ and every state $\sigma \in f_i$ there is a probability distribution $\rho_i(\sigma, \cdot)$ with a non-empty support $A(i, \sigma) \subseteq \Omega$ such that addressing flaw $f_i$ at state $\sigma$ amounts to selecting the next state $\sigma'$ from $A(i, \sigma)$ with probability $\rho_i(\sigma, \sigma')$. We call $A(i, \sigma)$ the set of *actions* for addressing flaw $f_i$ at $\sigma$ and note that potentially $A(i, \sigma) \cap f_i \neq \emptyset$, i.e., addressing a flaw does not necessarily imply removing it. The actions for flaw $f_i$ form a digraph $D_i$ on $\Omega$ having an arc $\sigma \xrightarrow{i} \sigma'$ for each pair $(\sigma, \sigma') \in f_i \times A(i, \sigma)$. Let $D$ be the multi-digraph on $\Omega$ that is the union of all $D_i$.

We consider algorithms that start from a state $\sigma \in \Omega$ picked from an initial distribution $\theta$, and then repeatedly pick a flaw that is present in the current state and address it. The algorithm always terminates when it encounters a flawless state.

To state the algorithmic LLL criteria for fast convergence of such algorithms we need to introduce two key ingredients. The first one is a notion of *causality* among flaws that will be used to induce a graph over $[m]$, which will play a role similar to the one of the dependency graph in the existential Local Lemma formulation. We note that there is a formal connection between causality graphs and dependency graphs (for more details see [21]).

▶ **Causality.** *For an arc $\sigma \xrightarrow{i} \sigma'$ in $D_i$ and a flaw $f_j$ present in $\sigma'$ we say that $f_i$ causes $f_j$ if $f_i = f_j$ or $f_j \not\ni \sigma$. If $D_i$ contains* any *arc in which $f_i$ causes $f_j$ we say that $f_i$* potentially causes *$f_j$.*

▶ **Causality Digraph.** *Any digraph $C = C(\Omega, F, D)$ on $[m]$ where $i \to j$ exists whenever $f_i$ potentially causes $f_j$ is called a* causality digraph*. The* neighborhood *of a flaw $f_i$ in $C$ is $\Gamma(i) = \{j : i \to j$ exists in $C\}$.*

The second ingredient is a measure of *compatibility* between the actions of the algorithm for addressing each flaw $f_i$ (that is, digraph $D_i$) and the probability measure $\mu$ over $\Omega$ which we will use for the analysis. As was shown in [21, 1, 24] one can capture compatibility by letting

$$d_i = \max_{\sigma \in \Omega} \frac{\nu_i(\sigma)}{\mu(\sigma)} \geq 1 \ , \tag{2}$$

where $\nu_i(\sigma)$ is the probability of ending up at state $\sigma$ at the end of the following experiment: sample $\omega \in f_i$ according to $\mu$ and address flaw $f_i$ at $\omega$. An algorithm achieving perfect compatibility for flaw $f_i$, i.e., $d_i = 1$, is a *resampling oracle* for flaw $f_i$ (observe that the Moser-Tardos algorithm is trivially a resampling oracle for every flaw). More generally, ascribing to each flaw $f_i$ the *charge*

$$\gamma(f_i) = d_i \cdot \mu(f_i) = \max_{\sigma' \in \Omega} \frac{1}{\mu(\sigma')} \sum_{\sigma \in f_i} \mu(\sigma)\rho_i(\sigma, \sigma') \ ,$$

yields the following algorithmization condition. If for every flaw $f_i \in F$,

$$\frac{\gamma(f_i)}{\psi_i} \sum_{S \subseteq \Gamma(i)} \prod_{j \in S} \psi_j < 1 \tag{3}$$

then there exists a flaw choice strategy under which the algorithm will reach a perfect object fast. (In most applications, that is in $O\left(\log|\Omega| + m \log_2\left(\frac{1+\psi_{\max}}{\psi_{\min}}\right)\right)$ steps with high probability.)

Throughout the paper we assume that we are given an undirected causality graph $C$ (and thus the relation $\Gamma(\cdot)$ is symmetric) and we will sometimes write $i \sim j$ if $j \in \Gamma(i) \leftrightarrow j \in \Gamma(j)$. Furthermore, for a set $S \subseteq [m]$ we define $\Gamma(S) = \bigcup_{i \in S} \Gamma(i)$. Finally, we denote by $\mathrm{Ind}(S) = \mathrm{Ind}_C(S)$ the set of independent subsets of $S$ with respect to $C$.

## 2.3 Commutativity

We will say that $\sigma \xrightarrow{i} \sigma'$ is a *valid trajectory* if it is possible to get from state $\sigma$ to state $\sigma'$ by addressing flaw $f_i$ as described in the algorithm, i.e., if two conditions hold: $i \in U(\sigma)$ and $\sigma' \in A(i, \sigma)$. Kolmogorov [24] described the following *commutativity* condition. We call the setting in which Definition 3 holds *the commutative setting.*

▶ **Definition 3** (Commutativity [24])**.** A tuple $(F, \sim, \rho)$ is called *commutative* if there exists a mapping Swap that sends any trajectory $\Sigma = \sigma_1 \xrightarrow{i} \sigma_2 \xrightarrow{j} \sigma_3$ with $i \nsim j$ to another valid trajectory $\text{Swap}(\Sigma) = \sigma_1 \xrightarrow{j} \sigma_2' \xrightarrow{i} \sigma_3$, and:

1. Swap is injective,
2. $\rho_i(\sigma_1, \sigma_2)\rho_j(\sigma_2, \sigma_3) = \rho_j(\sigma_1, \sigma_2')\rho_i(\sigma_2', \sigma_3)$ .

It is straightforward to check that the Moser Tardos algorithm satisfies the commutativity condition. Furthermore, Kolmogorov showed that the same is true for resampling oracles in the permutation [19] and perfect matchings [21] settings, and Harris [17] designed commutative resampling oracles for hamiltonian cycles.

Finally, as already mentioned, Kolmogorov showed that in the commutativity setting one may choose an arbitrary flaw choice strategy which is a function of the entire past execution history. The same will be true for our results, so we make the convention that given a tuple $(F, \sim, \rho)$ we always fix some arbitrary flaw choice strategy to get a well-defined, commutative algorithm $\mathcal{A} = (F, \sim, \rho)$.

## 2.4 Improved LLL Criteria

Besides the general form of the LLL (1) there exist improved criteria that apply in the full generality of the LLL setting. The most well-known are the *cluster expansion condition* [7] and the *Shearer's condition* [32]. Both of these criteria apply when the dependency graph is undirected and have been made constructive [22, 31, 2, 21, 1, 24] in the most general algorithmic LLL settings.

### Cluster Expansion Condition

The cluster expansion criterion strictly improves upon the General LLL criterion (1) by taking advantage of the local density of the dependency graph.

▶ **Definition 4.** Given a sequence of positive real numbers $\{\psi_i\}_{i=1}^m$, we say that the cluster expansion condition is satisfied if for each $i \in [m]$:

$$\frac{\gamma(f_i)}{\psi_i} \sum_{S \in \text{Ind}(\Gamma(i))} \prod_{j \in S} \psi_j \le 1 \ . \tag{4}$$

### Shearer's Condition

Let $\gamma \in \mathbb{R}^m$ be the real vector such that $\gamma_i = \gamma(f_i)$. Furthermore, for $S \subseteq [m]$ define $\gamma_S = \prod_{j \in S} \gamma_j$ and the polynomial $q_S$:

$$q_S = q_S(\gamma) = \sum_{\substack{I \in \text{Ind}([m]) \\ S \subseteq I}} (-1)^{|I|-|S|} \gamma_I \ .$$

▶ **Definition 5.** We say that the Shearer's condition is satisfied if $q_S(\gamma) \ge 0$ for all $S \subseteq [m]$, and $q_\emptyset(\gamma) > 0$.

## 3 Statement of Results

Assuming that the LLL conditions (1) hold, the *LLL-distribution*, which we denote by $\mu_{\text{LLL}}$, is defined as the distribution induced by the measure $\mu$ conditional on no bad event occurring. The following proposition relates the LLL distribution to measure $\mu$ making it a powerful tool that can be used to argue about properties of flawless objects. The idea is that if an (not necessarily bad) event $E$ is independent from most bad events, then its probability under the LLL distribution is not much larger than its probability under the probability measure $\mu$.

▶ **Proposition 6** ([15]). *If the LLL conditions* (1) *hold, then for any event $E$:*

$$\mu_{\mathrm{LLL}}(E) \le \mu(E) \sum_{S \subseteq D(E)} \prod_{j \in S} \psi_j \ , \tag{5}$$

*where $D(E) \subseteq [m]$ is such that $\mu(E \mid \bigcap_{j \in S} \overline{A}_j) = \mu(E)$ for all $S \subseteq [m] \setminus D(E)$.*

The main result of Haeupler, Saha and Srinivasan [15] is that the Moser-Tardos algorithm approximates well the LLL distribution, in the sense that the left-hanside of (5) bounds the probability that it ever reaches a subspace $E \subseteq \Omega$ during its execution. Building on this fact, [15] and followup works [20, 16] manage to show several new applications.

Here we extend the latter result to arbitrary commutative algorithms. Given an arbitrary set $E \subseteq \Omega$ and a commutative algorithm $\mathcal{A}$, consider an extension, $\mathcal{A}_E$, of $\mathcal{A}$ by defining an extra flaw $f_{m+1} \equiv E$ with its own set of probability distributions $\rho_{m+1}(\sigma, \cdot), \sigma \in E$. If $\mathcal{A}$ is commutative with respect to $\sim$, we will say that $\mathcal{A}_E$ is a *commutative extension* of $\mathcal{A}$ if $\mathcal{A}_E = (F \cup \{m+1\}, \sim, \rho)$ is also commutative.

Commutative extensions should be interpreted as a tool to bound the probability that $\mathcal{A}$ ever reaches a subset $E$ of the state space. That is, they are defined only for the purposes of the analysis and, typically in applications, they are a natural extension of the algorithm. For example, in the case of the Moser-Tardos algorithm applied to $k$-SAT, if one would like to bound the probability that the algorithm ever reaches a state such that variables $x_1, x_2$ of the formula are both set to true, then one could define $f_{m+1} = \{\sigma \in \Omega \text{ s.t. } \sigma(x_1) = \sigma(x_2) = 1\}$ along with the corresponding commutative extension of the Moser-Tardos algorithm that addresses $f_{m+1}$ by resampling variables $x_1, x_2$ according to the product measure over the variables of the formula that the Moser-Tardos algorithm uses whenever it needs to resample a violated clause. Indeed, commutative extensions of this form are implicitly defined in the analysis of [15] for the Moser-Tardos algorithm.

We will use the notation $\Pr[\cdot] = \Pr_{\mathcal{A}}[\cdot]$ to refer to the probability of events in the probability space induced by the execution of algorithm $\mathcal{A}$. For example, the probability that $\mathcal{A}$ ever reaches a set $E \subseteq \Omega$ of the state space during its execution will be denoted by $\Pr[E]$.

▶ **Theorem 7.** *If $\mathcal{A} = (F, \sim, \rho)$ is commutative and the cluster expansion condition is satisfied then:*

1. *for each $i \in [m]$: $\mathbb{E}[N_i] \le \lambda_{\mathrm{init}} \psi_i$ ;*
2. *for each $E \subseteq \Omega$: $\Pr[E] \le \lambda_{\mathrm{init}} \gamma(E) \sum_{S \in \mathrm{Ind}(\Gamma(E))} \prod_{j \in S} \psi_j$ ;*

*where $N_i$ is the number of times flaw $f_i$ is addressed during the execution of $\mathcal{A}$, $\lambda_{\mathrm{init}} = \max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)}$, and $\Gamma(E)$ and $\gamma(E)$ are defined with respect to a fixed commutative extension $\mathcal{A}_E$.*

▶ **Corollary 8.** *Algorithm $\mathcal{A}$ terminates after $O(\lambda_{\mathrm{init}} \sum_{i \in [m]} \psi_i)$ steps in expectation.*

▶ Remark. If the Shearer's condition is satisfied, then one can replace $\psi_i$ in Theorem 7 with $\frac{q_{\{i\}}(\gamma)}{q_{\emptyset}(\gamma)}$.

We note that the first part of Theorem 7 allows us to guarantee fast convergence of $\mathcal{A}$ to a perfect object without having to assume a "slack" in the cluster expansion and Shearer's conditions (unlike the works of [1, 24]) and, moreover, improves upon the (roughly quadratically worse) running bound of [21], matching the one of [22]. Whether the latter could be done was left as an open question in [21].

## 3.1 Entropy of the Output Distribution

Here we present one of the main byproducts of Theorem 7. The reader is referred to the full version of the paper for applications of Theorem 7 in partially avoiding flaws and dealing with settings with super-polynomially many flaws.

An elegant application of the known bounds for the Moser-Tardos distribution is estimating its randomness. In particular, Harris and Srinivasan [20] show that one can give lower bounds on the *Rényi entropy* of the output of the Moser-Tardos algorithm.

▶ **Definition 9** ([10]). Let $\nu$ be a probability measure over a finite set $S$. The *Rényi entropy with parameter* $\rho$ of $\nu$ is defined to be

$$H_\rho[\nu] = \frac{1}{1-\rho} \ln \sum_{s \in S} \nu(s)^\rho \enspace .$$

The *min-entropy* $H_\infty$ is a special case defined as $H_\infty[\nu] = \lim_{\rho \to \infty} H_\rho[\nu] = -\ln \max_{v \in S} \nu(\sigma)$.

Using the results of Section 3 we can show the analogous result in our setting.

▶ **Theorem 10.** *Assume that $\mathcal{A} = (F, \sim, \rho)$ is commutative, the cluster expansion condition is satisfied. Let $\nu$ be the output distribution of $\mathcal{A}$. Then, for $\rho > 1$,*

$$H_\rho[\nu] \geq H_\rho[\mu] - \frac{\rho}{\rho - 1} \ln \left( \sum_{S \in \mathrm{Ind}([m])} \prod_{j \in S} \psi_j \right) - \frac{\rho}{\rho - 1} \ln \lambda_{\mathrm{init}} \enspace .$$

Given Theorem 7, the proof is akin to the analogous result in [20] and can be found in the full version of the paper.

▶ Remark. Using Shearer's condition we can replace $\psi_i$ with $\frac{q_{\{i\}}(\gamma)}{q_\emptyset(\gamma)}$, $i \in [m]$.

A straightforward application of having a lower bound on $H_\rho[\nu]$ (for any $\rho$), where $\nu$ is the output distribution of the algorithm, is that there exist at least $\exp(H_\rho[\nu])$ flawless objects. Before [20], the authors in [25] also used the (existential) LLL for enumeration of combinatorial structures by exploiting the fact that it guarantees a small probability $p$ of avoiding all flaws when sampling from the uniform measure (and, thus, their number is at least $p|\Omega|$).

## 4 List-Coloring of Triangle-Free Graphs

In the problem of list coloring one is given a graph $G = G(V, E)$ over $n$ vertices $V = \{v_1, v_2, \ldots, v_n\}$ and, for each $v \in V$, a list of colors $\mathcal{L}_v$. The goal is to find a list coloring $\sigma \in \mathcal{L}_{v_1} \times \ldots \times \mathcal{L}_{v_n}$ of $G$ such that $\sigma(v) \neq \sigma(u)$ for any pair of adjacent vertices.

The *list chromatic number* $\chi_\ell(G)$ of a graph $G$ is the minimum number of colors for which such a coloring is attainable. A celebrated result of Johansson shows that there exist a large constant $C > 0$ such that every triangle-free graph with maximum degree $\Delta \geq \Delta_0$ can be properly list-colored using $C\Delta/\ln\Delta$ colors. Very recently, Molloy [26] improved Johansson's result showing that $C$ can be replaced by $(1+\epsilon)$ for any $\epsilon > 0$ assuming that $\Delta \geq \Delta_\epsilon$.(We note that, soon after, Bernshteyn [6] established the same bound for the list chromatic number using the LLL. However, his result is not constructive as it uses a sophisticated probability measure for which it is not clear how one could design "efficient" resampling oracles.)

Here we show how that the algorithm of Molloy is amenable to our analysis and, in particular, we prove that it can output exponentially many proper colorings with positive probability.

## 4.1 The Algorithm

The algorithm of [26] works in two stages. First, it finds a partial list-coloring which has the property that (i) each vertex $v \in V$ has "many"' available colors; (ii) there is not "too much competition" for the available colors of $v$, i.e., they do not appear in the list of available colors of its neighbors. Then, it completes the coloring via a fairly straightforward application of the Moser-Tardos algorithm.

To describe the algorithm formally, we will need some further notation. First, it will be convenient to treat Blank as a color that is in the list of every vertex. For each vertex $v$ and *partial* list-coloring $\sigma$ let

- $N_v$ denote the set of vertices adjacent to $v$;
- $L_v(\sigma) \subseteq \mathcal{L}_v$ to be the set of *available* colors for $v$ at state $\sigma$, i.e., the set of colors that we can assign to $v$ in $\sigma$ without making any edge monochromatic. Notice that Blank is always an available color;
- $T_{v,c}(\sigma)$ to be the set of vertices $u \in N_v$ such that $\sigma(u) = \text{Blank}$ and $c \in L_u(\sigma)$.

Let $\Omega = \prod_{v \in V} \mathcal{L}_v$ and define $L = \Delta^{\frac{\epsilon}{2}}$. Given a partial list-coloring, we define the following flaws for any vertex $v$:

$$\begin{aligned} B_v &= \{\sigma \in \Omega : |L_v(\sigma)| < L\}; \\ Z_v &= \left\{ \sigma \in \Omega : \sum_{c \in L_v(\sigma) \setminus \text{Blank}} |T_{v,c}(\sigma)| > \frac{1}{10} L \cdot |L_v(\sigma)| \right\} . \end{aligned}$$

▶ **Lemma 11** (The Second Phase). *Given a flawless partial list coloring, a complete list-coloring of $G$ can be found in expected polynomial time.*

Lemma 11 was proved in [26] via a fairly straightforward application of the Lovász Local Lemma, and can be made constructive via the Moser-Tardos algorithm. What is left is to describe the first phase of the algorithm.

- The initial distribution $\theta$, which is important in this case, is chosen to be the following: Fix an independent set $S$ of $G$ of size at least $n/(\Delta+1)$. (This is trivial to find efficiently via a greedy algorithm). Choose one color from $\mathcal{L}_u \setminus \text{Blank}$, $u \in S$, uniformly at random, and assign it to $u$;
- to address a flaw $f \in \{B_v, Z_v\}$ at state $\sigma$, for each $u \in N_v$, choose uniformly at random a color from $L_u(\sigma)$ and assign it to $u$;
- as a flaw choice strategy, the algorithm first fixes any ordering $\pi$ over flaws. At every step, it chooses the lowest occurring flaw according to $\pi$ and addresses it.

## 4.2 Proving Termination

Let $\mathcal{A}_1$, $\mathcal{A}_2$ denote the first and second phase of our algorithm, respectively. Here we prove that $\mathcal{A}_1$ terminates in expected polynomial time. To do so, we will use the convergence result corresponding to equation (3). (Although, as we will see, $\mathcal{A}_1$ is commutative for an appropriate choice of a causality graph, we won't use Theorem 7 to prove its convergence. This is because $\lambda_{\text{init}}$ is exponentially large in this case).

The measure $\mu$ we use for the analysis is the the uniform measure over proper list-colorings. We will use the following lemma whose proof can be found in Section B of the Appendix.

▶ **Lemma 12.** *For each vertex $v$ and flaw $f \in \{B_v, Z_v\}$ we have that*

$$\gamma(f) \le 2\Delta^{-4} \ .$$

Consider the causality graph such that $f_v \sim f_u$, if $\mathrm{dist}(u,v) \leq 3$, where $f_v, f_u$ are either $B$-flaws or $Z$-flaws. Notice that it has maximum degree at most $2(\Delta^3+1)$. Setting $\psi_f = \psi = \frac{1}{2(\Delta^3+1)}$ for any flaw $f$ and applying (3), we get that the algorithm converges in expected polynomial time since

$$\gamma(f) \sum_{S \subseteq \Gamma(f)} \prod_{g \in S} \psi_g \leq \frac{2}{\Delta^4} \cdot 2(\Delta^3+1) \cdot \mathrm{e} < 4\mathrm{e}\left(\frac{1}{\Delta} + \frac{1}{\Delta^4}\right) < 1 \ ,$$

for large enough $\Delta$ for any flaw $f \in \{B_v, Z_v\}$ and $\log_2 |\Omega| + m \log_2 \left(\frac{1+\psi}{\psi}\right) = O(n \log n + m \log_2 n)$.

## 4.3 A Lower Bound on the Number of Possible Outputs

The bound regarding the number of list colorings the algorithm can output with positive probability follows almost immediately from the two following lemmata.

▶ **Lemma 13.** *Algorithm $\mathcal{A}_1$ can output at least $\exp\left(n\left(\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}\right)\right)$ flawless partial list-colorings with positive probability.*

**Proof.** It is not hard to verify that $\mathcal{A}_1 = (F, \sim, \rho)$ is commutative. Applying Theorem 10 we get that $\mathcal{A}_1$ can output at least

$$\exp\left(\ln \frac{|\Omega|}{\lambda_{\text{init}}} - \sum_{f \in F} \psi_f\right) > \exp\left(n\left(\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}\right)\right) \ , \tag{6}$$

flawless partial colorings. ◀

▶ **Lemma 14.** *Suppose $\mathcal{A}_1$ can output $N$ flawless partial list-colorings with positive probability. Suppose further that among these partial list-colorings, the ones with the lowest number of colored vertices have exactly $\alpha n$ vertices colored, $\alpha \in (0,1)$. Then, $\mathcal{A}_2$ can output at least $\max\left(N2^{-(1-\alpha)n}, \left(\frac{8L}{11}\right)^{(1-\alpha)n}\right)$ list-colorings with positive probability.*

The proof of Lemma 14 can be found in the full version of the paper.

**Proof of Theorem 2.** Combining Lemmata 13, 14 we get that our algorithm outputs at least

$$\exp\left(n \min_{\alpha} \max\left(f(\Delta) - (1-\alpha)\ln 2, (1-\alpha)\ln\left(\frac{8L}{11}\right)\right)\right) = \mathrm{e}^{cn} \ ,$$

list-colorings with positive probability, where $f(\Delta) = \frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}$ and $c = f(\Delta)\frac{\ln \frac{8L}{11}}{\ln \frac{16L}{11}}$, concluding the proof. ◀

## References

1 Dimitris Achlioptas and Fotis Iliopoulos. Focused stochastic local search and the Lovász local lemma. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 2024–2038, 2016. doi:10.1137/1.9781611974331.ch141.

2 Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *J. ACM*, 63(3):22:1–22:29, 2016. `doi:10.1145/2818352`.

3 Dimitris Achlioptas, Fotis Iliopoulos, and Alistair Sinclair. A new perspective on stochastic local search and the lovasz local lemma. *CoRR*, abs/1805.02026, 2018. `arXiv:1805.02026`.

4 Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991. `doi:10.1002/rsa.3240020403`.

5 József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures Algorithms*, 2(4):343–365, 1991. `doi:10.1002/rsa.3240020402`.

6 Anton Bernshteyn. The johansson–molloy theorem for dp-coloring. *arXiv preprint arXiv:1708.03843*, 2017.

7 Rodrigo Bissacot, Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combinatorics, Probability & Computing*, 20(5):709–719, 2011. `doi:10.1017/S0963548311000253`.

8 Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM J. Comput.*, 42(6):2132–2155, 2013. `doi:10.1137/100799642`.

9 Antares Chen, David G. Harris, and Aravind Srinivasan. Partial resampling to approximate covering integer programs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1984–2003. SIAM, 2016. `doi:10.1137/1.9781611974331.ch139`.

10 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. `doi:10.1137/0217015`.

11 Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 134–143. ACM, 2014. `doi:10.1145/2611462.2611465`.

12 Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000)*, pages 30–39, 2000.

13 Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.

14 Bernhard Haeupler and David G Harris. Parallel algorithms and concentration bounds for the Lovász local lemma via witness-DAGs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1170–1187. SIAM, 2017.

15 Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):Art. 28, 28, 2011. `doi:10.1145/2049697.2049702`.

16 David G. Harris. New bounds for the Moser-Tardos distribution: Beyond the Lovasz local lemma. *CoRR*, abs/1610.09653, 2016. `arXiv:1610.09653`.

17 David G. Harris. Oblivious resampling oracles and parallel algorithms for the lopsided Lovász local lemma. *CoRR*, abs/1702.02547, 2017. `arXiv:1702.02547`.

18 David G. Harris and Aravind Srinivasan. The Moser-Tardos framework with partial resampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 469–478. IEEE Computer Society, 2013. `doi:10.1109/FOCS.2013.57`.

19 David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 907–925. SIAM, 2014. `doi:10.1137/1.9781611973402.68`.

20 David G. Harris and Aravind Srinivasan. Algorithmic and enumerative aspects of the Moser-Tardos distribution. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 2004–2023, 2016. `doi:10.1137/1.9781611974331.ch140`.

21 Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1327–1346. IEEE Computer Society, 2015. `doi:10.1109/FOCS.2015.85`.

22 Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244. ACM, 2011. `doi:10.1145/1993636.1993669`.

23 Kashyap Babu Rao Kolipaka, Mario Szegedy, and Yixin Xu. A sharper local lemma with improved applications. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 603–614. Springer, 2012. `doi:10.1007/978-3-642-32512-0_51`.

24 Vladimir Kolmogorov. Commutativity in the algorithmic Lovász local lemma. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 780–787. IEEE Computer Society, 2016. `doi:10.1109/FOCS.2016.88`.

25 Linyuan Lu and Laszlo A Szekely. A new asymptotic enumeration technique: the Lovász local lemma. *arXiv preprint arXiv:0905.3983*, 2009.

26 Michael Molloy. The list chromatic number of graphs with small clique number. *arXiv preprint arXiv:1701.09133*, 2017.

27 Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *STOC '98 (Dallas, TX)*, pages 524–529. ACM, New York, 1999.

28 Robin A. Moser. A constructive proof of the Lovász local lemma. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 343–350. ACM, New York, 2009.

29 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010. `doi:10.1145/1667053.1667060`.

30 Christos H. Papadimitriou. On selecting a satisfying truth assignment. In *FOCS*, pages 163–169. IEEE Computer Society, 1991. `doi:10.1109/SFCS.1991.185365`.

31 Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM J. Discrete Math.*, 28(2):911–917, 2014. `doi:10.1137/110828290`.

32 J.B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985. `doi:10.1007/BF02579368`.

33 Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In Shang-Hua Teng, editor, *SODA*, pages 611–620. SIAM, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347150`.

## A Proof of Main Results

In this section we state and prove the witness tree lemma for our setting. We then use it to prove Theorem 7. Some of the proofs are omitted for the sake of brevity, but all of them can be found in the full version of the paper.

### A.1 The Witness Tree Lemma

Given a trajectory $\Sigma = \sigma_1 \xrightarrow{w_1} \dots \sigma_t \xrightarrow{w_t} \sigma_{t+1}$ we denote by $W(\Sigma) = (w_1, \dots, w_t)$ the *witness sequence* of $\Sigma$. (Recall that according to our notation, $w_i$ denotes the index of the flaw that was addressed at the $i$-th step).

To state the witness tree lemma, we will first need to recall the definition of witness trees from [29], slightly reformulated to fit our setting. A witness tree $\tau = (T, \ell_T)$ is a finite rooted, unordered, tree $T$ along with a labelling $\ell_T : V(T) \to [m]$ of its vertices with indices of flaws such that the children of a vertex $v \in V(T)$ receives labels from $\Gamma(\ell(v))$. To lighten the notation, we will sometimes write $[v]$ to denote $\ell(v)$ and $V(\tau)$ instead of $V(T)$. Given a witness sequence $W = (w_1, w_2, \dots, w_t)$ we associate with each $i \in [t]$ a witness tree $\tau_W(i)$ that is constructed as follows: Let $\tau_W^{(i)}(i)$ be an isolated vertex labelled by $w_i$. Then, going backwards for each $j = i-1, i-2, \dots, 1$: if there is a vertex $v \in \tau_W^{j+1}(i)$ such that $[v] \sim w_j$ then we choose among those vertices the one having the maximum distance (breaking ties arbitrarily) from the root and attach a new child vertex $u$ to $v$ that we label $w_j$ to get $\tau_W^{(j)}(i)$. If there is no such vertex $v$ then $\tau_W^{(j+1)}(i) = \tau_W^{(j)}(i)$. Finally, let $\tau_W(i) = \tau_W^{(1)}(i)$.

We will say that a witness tree $\tau$ occurs in a trajectory $\Sigma$ if $W(\Sigma) = (w_1, w_2, \dots, w_t)$ and there is $k \in [t]$ such that $\tau_W(k) = \tau$.

▶ **Theorem 15** (The witness tree lemma). *Assume that $\mathcal{A} = (F, \sim, \rho)$ is commutative. Then, for every witness tree $\tau$ we have that:*

$$\Pr[\tau] \le \lambda_{\text{init}} \prod_{v \in V(\tau)} \gamma(f_{[v]}) \ .$$

We show the proof of Theorem 15 in Section A.4.

### A.2 Witness Trees and Stable Witness Sequences

Here we show some properties of witness trees (which are induced by witness sequences of the algorithm) that will be useful to us later. We also draw a connection between witness trees and *stable witness sequences*, which we will need in the proof of Theorem 7. Stable witness sequences were first introduced in [22] to make the Shearer's criterion constructive in the variable setting.

#### A.2.1 Properties of Witness Trees

The following propositions capture the main properties of witness trees we will need.

▶ **Proposition 16.** *For a witness tree $\tau = (T, \ell_T)$ let $L_i = L_i(\tau)$ denote the set of labels of the nodes at distance $i$ from the root. For each $i \ge 0$, $L_i \in \mathrm{Ind}([m])$.*

▶ **Proposition 17.** *For a witness sequence $W$ of length $t$ and any two distinct $i, j \in [t]$ we have that $\tau_W(i) \ne \tau_W(j)$.*

### A.2.2 Stable Witness Sequences

We will now recall the definition of *stable sequences*. Variations of this notion have also been used by [21, 24] to make the Shearer's criterion constructive in the more general settings of the algorithmic Local Lemma. Here we will use the following definition:

▶ **Definition 18.** A sequence of subsets $(I_1, \ldots, I_k)$ of $[m]$ with $k \geq 1$ is called *stable* if
1. $I_r \in \mathrm{Ind}([m]) \setminus \{\emptyset\}$ for each $r \in [k]$
2. $I_{r+1} \subseteq \Gamma(I_r)$ for each $r \in [k-1]$.

▶ **Definition 19.** A witness sequence $W = (w_1, \ldots, w_t)$ is called *stable* if it can be partitioned into non-empty sequences as $W = (W_1, \ldots, W_k)$ such that the elements of each sequence $W_r$ are distinct, and the sequence $\phi_W := (I_1, \ldots, I_k)$ is stable, where $I_r$ is the set of indices of flaws in $W_r$ (for $r \in [k]$).

For any arbitrary ordering $\pi$ among indices of flaws, if in addition each sequence $W_r = (w_i, \ldots, w_j)$ satisfies $w_i \prec_\pi \ldots \prec_\pi w_j$ then $W$ is called $\pi$*- stable.*

▶ **Proposition 20.** *([24]) For a stable witness sequence the partitioning in Definition 19 is unique.*

For a witness sequence $W = (w_1, \ldots, w_t)$ let $\mathrm{Rev}[W] = (w_t, \ldots, w_1)$ denote the reverse sequence. Let also $R_W$ denote the first set (the "root") of the stable sequence $\phi_W := (I_1, \ldots, I_k)$, i.e., $R_W = I_1$. Finally, let $\mathcal{R}_i^\pi$ be the set of witness sequences $W$ such that $\mathrm{Rev}[W]$ is $\pi$-stable and $R_{\mathrm{Rev}[W]} = \{i\}$.

There is a connection between stable sequences and witness trees that we will need for the proofs of Theorem 7 and which we will describe below.

Let $\mathcal{W}_i$ denote the set of witness trees with root labelled by $i$. For each $\tau \in \mathcal{W}_i$, let $\chi_\pi(\tau)$ be the *ordered* witness tree that is induced by ordering the children of each node in $\tau$ from left to right, increasingly according to $\pi$. Define $\mathcal{W}_i^\pi := \chi_\pi(\mathcal{W}_i)$ and observe that $\chi_\pi$ is a bijection. Finally, recall that for a witness tree $\tau$ we denote by $L_j(\tau)$ the set of labels of the nodes at distance $j$ from the root.

▶ **Lemma 21.** *There is a bijection $\chi_i^\pi$ mapping $\mathcal{R}_i^\pi$ to $\mathcal{W}_i^\pi$ with the following property: Let $W \in \mathcal{R}_i^\pi$ and let $(I_1 = \{i\}, I_2, \ldots, I_k)$ be the unique partitioning of* $\mathrm{Rev}[W]$ *guaranteed by Proposition 20. Then, $I_j = L_{j-1}\left(\chi_i^\pi(W)\right)$ for each $j \in [k]$ .*

### A.2.3 Counting Witness Trees

In our proofs we will need to bound the sum over all trees $\tau \in \mathcal{W}_i$ of the product of charges of the (labels of) the nodes of each tree $\tau$. Fortunately, the method for doing that is well trodden by now (see for example [29, 31]). Recall that $\mathcal{W}_i$ denotes the set of all possible witness trees with root that is labelled by $i$.

▶ **Lemma 22.** *If the Cluster Expansion condition is satisfied then:*

$$\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \gamma\left(f_{[v]}\right) \leq \psi_i \ .$$

We also show the following lemma that can be used whenever the Shearer's condition applies.

▶ **Lemma 23.** *If the Shearer's condition is satisfied then:*

$$\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \gamma\left(f_{[v]}\right) \leq \frac{q_{\{i\}}(\gamma)}{q_{\emptyset}(\gamma)} \ .$$

▶ Remark. We note that if we have assumed a stronger "cluster expansion condition" (namely, in (4) we have $\Gamma(i) \cup \{i\}$) instead of $\Gamma(i)$ then Corollary 22 could have also been shown as an immediate application of Lemma 23, since it is known ([7, 21, 24]) that, in this case, for every $i \in [m]$ we have that $\frac{q_{\{i\}}(\gamma)}{q_{\emptyset}(\gamma)} \le \psi_i$.

## A.3 Proof of Theorem 7

We first prove Theorem 7. The first part follows by Theorem 15, Lemma 22 and Proposition 17 that suggest:

$$\mathbb{E}[N_i] \le \lambda_{\text{init}} \sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \gamma(f_{[v]}) \le \lambda_{\text{init}} \psi_i \quad .$$

To see the second part of Theorem 7, consider the new set of flaws $F' = F \cup \{f_{m+1}\}$, where $f_{m+1} = E$, as well as a "truncated" commutative extension $\mathcal{A}'$ of $\mathcal{A}$ with the following properties:

**(i)** For each state $\sigma \notin f_{m+1}$ algorithm $\mathcal{A}'$ invokes $\mathcal{A}$ to choose the next state.

**(ii)** $\gamma(E) := \gamma_{\mathcal{A}'}(f_{m+1})$.

**(iii)** $f_{m+1}$ is always of the highest priority: when at a state $\sigma \in f_{m+1}$, $\mathcal{A}'$ chooses to address $f_{m+1}$.

**(iv)** $\mathcal{A}'$ stops after it addresses $f_{m+1}$ for the first time.

By coupling $\mathcal{A}$ and $\mathcal{A}'$ we see that $\Pr_{\mathcal{A}}[E] = \Pr_{\mathcal{A}'}[f_{m+1}]$. Let $\mathcal{W}_E$ be the set of witness trees that might occur in an execution of $\mathcal{A}'$ and whose root is labelled by $m+1$. Notice that due to property (iv) of $\mathcal{A}'$ every tree $\tau \in \mathcal{W}_E$ contains exactly one node (the root) labelled by $m+1$, while every other node is labelled by elements in $[m]$. Furthermore, the set of labels of the children of the root of $\tau$ is an element of $\text{Ind}(\Gamma(E))$. Finally, if $v$ is a node that corresponds to a child of the root in $\tau$, then the subtree $\tau_v$ that is rooted at $v$ is an element of $\mathcal{W}_{[v]}$. Using Theorem 15 and the fact that $\mathcal{A}'$ is commutative we can bound $\Pr_{\mathcal{A}}[E]$ as

$$\sum_{\tau \in \mathcal{W}_E} \Pr_{\mathcal{A}'}[\tau] \le \lambda_{\text{init}} \gamma(E) \sum_{S \in \text{Ind}(\Gamma(E))} \left( \prod_{j \in S} \sum_{\tau \in \mathcal{W}_j} \prod_{v \in \tau} \gamma([v]) \right) \le \lambda_{\text{init}} \gamma(E) \sum_{S \in \text{Ind}(\Gamma(E))} \prod_{j \in S} \psi_j,$$

where the last equality follows from Corollary 22. The proof of Theorem 7 in the Shearer's condition regime is identical, where instead of Lemma 22 we use Lemma 23.

## A.4 Proof of Lemma 15

Throughout the proof, we will use ideas and definitions from [24]. We also note that we will assume w.l.o.g. that algorithm $\mathcal{A}$ follows a deterministic flaw choice strategy. This is because randomized flaw choice strategies can equivalently be interpreted as convex combination of deterministic ones (and therefore, randomized strategies can be seen as taking expectation over deterministic ones).

For a trajectory $\Sigma$ of length $t$ we define

$$p(\Sigma) = \lambda_{\text{init}} \prod_{i=1}^{t} \rho_{w_i}(\sigma_i, \sigma_{i+1})$$

and notice that $\Pr[\Sigma] \le p(\Sigma)$. Furthermore, we say that a trajectory $\Sigma'$ is a proper prefix of $\Sigma$ if $\Sigma'$ is a prefix of $\Sigma$ and $\Sigma \neq \Sigma'$.

▶ **Definition 24** ([24]). A set $\mathcal{X}$ of trajectories of the algorithm will be called *valid* if (i) all trajectories in $\mathcal{X}$ follow the same deterministic flaw choice strategy (not necessarily the same used by $\mathcal{A}$), and (ii) for any $\Sigma, \Sigma' \in \mathcal{X}$ trajectory $\Sigma$ is not a proper prefix of $\Sigma'$.

▶ **Lemma 25** ([24]). *Consider a witness sequence $W = (w_1, \ldots, w_t)$ and a valid set of trajectories $\mathcal{X}$ such that $W$ is a prefix of $W(\Sigma)$ for every $\Sigma \in \mathcal{X}$. Then:*

$$\sum_{\Sigma \in \mathcal{X}} p(\Sigma) \le \lambda_{\text{init}} \prod_{i=1}^{t} \gamma(f_{w_i}) \ ,$$

A *swap* is the operation of transforming a trajectory $\Sigma = \ldots \sigma_1 \xrightarrow{i} \sigma_2 \xrightarrow{j} \sigma_3 \ldots$, with $i \nsim j$, to a trajectory $\Sigma' = \ldots \sigma_1 \xrightarrow{j} \sigma_2' \xrightarrow{i} \sigma_3 \ldots$, where $\sigma_1 \xrightarrow{j} \sigma_2' \xrightarrow{i} \sigma_3 = \text{Swap}(\sigma_1 \xrightarrow{i} \sigma_2 \xrightarrow{j} \sigma_3)$. A mapping $\Phi$ on a set of trajectories will be called a *swapping mapping* if it operates by applying a sequence of swaps.

The main idea now will be to construct a swapping mapping whose goal will be to transform trajectories of the algorithm to a form that satisfies certain properties by applying swaps .

For a trajectory $\Sigma$ in which a tree $\tau \in \mathcal{W}_i$ occurs, we denote by $W_\Sigma^\tau$ the prefix of $W(\Sigma)$ up to the step that corresponds to the root of $\tau$ (observe that Proposition 17 mandates that there exists a unique such step). Notice that, since $\tau \in \mathcal{W}_i$, the algorithm addresses flaw $f_i$ at this step, and thus the final element of $W_\Sigma^\tau$ is $\{i\}$. Finally, recall the definitions of $\mathcal{R}_i^\pi$, $\chi_\pi$ and $\chi_i^\pi$.

▶ **Lemma 26.** *Fix a witness tree $\tau \in \mathcal{W}_i$ and let $\mathcal{X}^\tau$ be a valid set of trajectories in which $\tau$ occurs. If $\mathcal{A} = (F, \sim, \rho)$ is commutative then there exists a set of trajectories $\mathcal{X}_\pi^\tau$ and a swapping mapping $\Phi^\tau : \mathcal{X}^\tau \to \mathcal{X}_\pi^\tau$ which is a bijection such that*
*(a) for any $\Sigma \in \mathcal{X}_\pi^\tau$ we have that $W_\Sigma^\tau$ is the unique witness sequence in $\mathcal{R}_i^\pi$ such that $\chi_i^\pi(W_\Sigma^\tau) = \chi_\pi(\tau)$;*
*(b) for any witness sequence $W$ the set $\{\Sigma \in \mathcal{X}_\pi^\tau \mid \text{Rev}[W_\Sigma^\tau] = W\}$ is valid.*

We prove Lemma 26 in Section A.5. To see Theorem 15, consider a witness tree $\tau \in \mathcal{W}_i$, and let $\mathcal{Y}^\tau$ be the set of all trajectories that $\mathcal{A}$ may follow in which $\tau$ occurs. Now remove from $\mathcal{Y}^\tau$ any trajectory $\Sigma$ for which there exists a trajectory $\Sigma'$ such that $\Sigma$ is a proper prefix of $\Sigma'$ to get $\mathcal{X}^\tau$. Clearly, this is a valid set and so recalling that $\chi_\pi$ is a bijection and applying Lemma 26 we have that:

$$\Pr[\tau] = \sum_{\Sigma \in \mathcal{X}^\tau} \Pr[\Sigma] \le \sum_{\Sigma \in \mathcal{X}^\tau} p(\Sigma) = \sum_{\Sigma \in \mathcal{X}_\pi^\tau} p(\Sigma) \ , \tag{7}$$

where to get the second equality we use the second requirement of Definition 3. Lemma 26 further implies that for every trajectory $\Sigma \in \mathcal{X}_\pi^\tau$ we have that $W_\Sigma^\tau$ is the (unique) witness sequence in $\mathcal{R}_i^\pi$ such that $\chi_i^\pi(W_\Sigma^\tau) = \chi_\pi(\tau)$, i.e., $W_\Sigma^\tau = (\chi_i^\pi)^{-1}(\chi_\pi(\tau))$ . This means that the witnesses of the trajectories in $\mathcal{X}_\pi^\tau$ have $W := (\chi_i^\pi)^{-1}(\chi_\pi(\tau))$ as a common prefix. Since part $(b)$ of Lemma 26 implies that $\mathcal{X}_\pi^\tau$ is valid, applying Lemma 25 we get:

$$\sum_{\Sigma \in \mathcal{X}_\pi^\tau} p(\Sigma) \le \lambda_{\text{init}} \prod_{w \in W} \gamma(f_w) = \lambda_{\text{init}} \prod_{v \in V(\tau)} \gamma(f_{[v]}) \quad , \tag{8}$$

where the second inequality follows from the fact that $\chi_i^\pi(W) = \chi_\pi(\tau)$ and $V(\tau) = V(\chi_\pi(\tau))$, concluding the proof.

## A.5 Proof of Lemma 26

Our proof builds on the proof of Theorem 19 in [24]. We will be denoting witness sequences $W = (w_1, w_2, \dots, w_t)$ as a sequence of *named indices of flaws* $W = (\mathbf{w}_1, \dots, \mathbf{w}_t)$ where $\mathbf{w}_j = (w_j, n_j)$ and $n_j = |\{k \in [j] \mid w_k = w_j\}| \geq 1$ is the number of occurrences of $w_j$ in the length-$j$ prefix of $W$. Note that a named index $\mathbf{w}$ cannot appear twice in a sequence $W$. Finally, if $\mathbf{w}$ is a named index of flaw we denote by $w$ (that is, without bold font) the flaw index that is associated with it.

For a trajectory $\Sigma$ such that $W(\Sigma) = (\mathbf{w}_1, \dots, \mathbf{w}_t)$ we define a directed acyclic graph $\mathbf{G}(\Sigma) = (\mathbf{V}(\Sigma), \mathbf{E}(\Sigma))$ where $\mathbf{V}(\Sigma) = \{\mathbf{w}_1, \dots, \mathbf{w}_t\}$ and $\mathbf{E}(\Sigma) = \{(\mathbf{w}_j, \mathbf{w}_k) \text{ s.t. } w_j \sim w_k \text{ and } j < k\ \}$. This means that we have an edge from a named flaw $\mathbf{w}_i$ to another flaw $\mathbf{w}_j$ whenever their corresponding flaw indices are related according to $\sim$ and $\mathbf{w}_j$ occurs in $\Sigma$ before $\mathbf{w}_k$.

By Proposition 17, for any trajectory $\Sigma$ in which $\tau$ occurs there is a unique step $t^* = t^*(\Sigma)$ such that $\tau_{W(\Sigma)}(t^*) = \tau$. For such a trajectory $\Sigma$, let $\mathbf{Q}(\Sigma) \subseteq \mathbf{V}(\Sigma)$ be the set of flaws from which the node $\mathbf{w}_{t^*}$ can be reached in $\mathbf{G}(\Sigma)$, where $\mathbf{w}_{t^*}$ is the named flaw index that corresponds to the step $t^*$. Notice that $w_{t^*} = i$ (since $\tau \in \mathcal{W}_i$). For $\mathbf{w} \in \mathbf{Q}(\Sigma)$ let $d(\mathbf{w})$ be the length of the longest path from $\mathbf{w}$ to $\mathbf{w}_{t^*}$ in $\mathbf{G}(\Sigma)$ plus one. For example, $d(\mathbf{w}_{t^*}) = 1$.

Let $Q(\Sigma)$ denote the sequence consisting of the named flaws in $\mathbf{Q}(\Sigma)$ listed in the order they appear in $\Sigma$. The idea is to repeatedly apply the operation Swap to $\Sigma$ so that we reach a trajectory $\Sigma'$ that has a permutation $Q_\pi(\Sigma)$ of $Q(\Sigma)$ as a prefix. In particular, we will show that $Q_\pi(\Sigma) \in \mathcal{R}_i^\pi$ and $\chi_i^\pi(Q_\pi(\Sigma)) = \chi_\pi(\tau)$.

To that end, for an integer $r \geq 1$ define $\mathbf{I}_r = \{\mathbf{w} \in \mathbf{Q}(\Sigma) \mid d(\mathbf{w}) = r\}$, and let $Q_r$ be the sequence consisting of the named flaw indices in $\mathbf{I}_r$ sorted in decreasing order with respect to $\pi$. Then, we define $Q_\pi(\Sigma) = (Q_s, \dots, Q_1)$ where $s = \max\{d(\mathbf{w}) \mid \mathbf{w} \in \mathbf{Q}(\Sigma)\}$.

▶ **Lemma 27.** *$Q_\pi(\Sigma) \in \mathcal{R}_i^\pi$ and $\chi_i^\pi(Q_\pi(\Sigma)) = \chi_\pi(\tau)$.*

**Proof.** Let $Y = Y(\Sigma) = \mathrm{Rev}[Q_\pi(\Sigma)] = (Q_1, Q_2, \dots, Q_s)$ be the reverse sequence of $\Sigma$. By definition, $R_Y = Q_1 = \{i\}$. To show that $Q \in \mathcal{R}_i^\pi$ it suffices to show that $Q_{i+1} \subseteq \Gamma(Q_i)$ for each $i \in [s-1]$. To see this, recall the definitions of $\mathbf{I}_{r+1}$ and $Q_{r+1}$ and observe that, for each $i_{r+1} \in Q_{r+1}$, there must be a path of $r$ indices of flaws $i_r, i_{r-1}, \dots, i_1$ such that for every $j \in [r-1]$ we have that $i_j \in Q_j$ and $i_j \sim i_{j+1}$.

Let $k$ be the number of elements in witness sequence $Q(\Sigma)$. Recall that $\chi_\pi^i(Q_\pi(\Sigma)) := \chi_\pi(\tau_{Q_\pi(\Sigma)}(k))$ (proof of Lemma 21). The proof is concluded by also recalling the algorithm for constructing witness trees and observing that $\tau_{Q_\pi(\Sigma)}(k) = \tau_{W(\Sigma)}(t^*) = \tau$. ◀

Note that applying Swap to $\Sigma$ does not affect graph $\mathbf{G}(\Sigma)$ and set $\mathbf{Q}(\Sigma)$ and, thus, neither the sequence $Q_\pi(\Sigma)$. With that in mind, we show next how we could apply Swap repeatedly to $\Sigma \in \mathcal{X}^\tau$ to reach a $\Sigma'$ such $Q_\pi(\Sigma)$ is a prefix of its witness sequence (that is, $W(\Sigma') = (Q_\pi(\Sigma), U)$). We will do this by applying swaps to *swappable pairs* in $\Sigma$.

▶ **Definition 28.** Consider a trajectory $\Sigma \in \mathcal{X}^\tau$. A pair $(\mathbf{w}, \mathbf{y})$ of named indices of flaws is called *a swappable* pair in $\Sigma$ if it can be swapped in $\Sigma$ (i.e., $W(\Sigma) = (\dots \mathbf{w}, \mathbf{y} \dots)$ and $\mathbf{w} \nsim \mathbf{y}$) and either

1. $(\mathbf{w}, \mathbf{y}) \in (\mathbf{V}(\Sigma) \setminus \mathbf{Q}(\Sigma)) \times \mathbf{Q}(\Sigma)$, or
2. $(\mathbf{w}, \mathbf{y}) \in \mathbf{Q}(\Sigma) \times \mathbf{Q}(\Sigma)$ and their order in $Q_\pi(\Sigma)$ is different: $Q_\pi(\Sigma) = (\dots, \mathbf{y}, \mathbf{w}, \dots)$

The position of the rightmost swappable pair in $\Sigma$ will be denoted as $k(\Sigma)$ , where the position of $(\mathbf{w}, \mathbf{y})$ in $\Sigma$ is the number of named indices of flaws that precede $\mathbf{y}$ in $W(\Sigma)$. If $\Sigma$ does not contain a swappable pair then $k(\Sigma) = 0$. Thus, $k(\Sigma) \in [0, |\Sigma| - 1]$.

We can only apply finite many swaps to swappable pairs in $\Sigma$. This is because swapping pairs of the first form moves a named index in $\mathbf{Q}(\Sigma)$ to the left, while swapping pairs of the second one decrease the number of pairs whose relative order in $Q(\Sigma)$ is not consistent with the one in $Q_\pi(\Sigma)$. Clearly, both of these actions can be performed only a finite number of times.

The following lemma shows how we can obtain a mapping $\Phi^\tau$ such that $\mathcal{X}_\pi^\tau := \Phi^\tau(\mathcal{X}^\tau)$ satisfies the first condition of Lemma 26. The proof is identical (up to minor changes) to the one of Lemma 27 of [24].

▶ **Lemma 29.** *Consider a trajectory $\Sigma \in \mathcal{X}^\tau$ such that $W(\Sigma) = (A, U)$ where $A$ and $U$ are some sequences of indices of flaws, and there are no swappable pairs inside $U$. Then $U = (B, C)$ where sequence $B$ is a subsequence of $Q_\pi(\Sigma)$ and $C$ does not contain named indices of flaws from $\mathbf{Q}(\Sigma)$.*

*In particular, if $|A| = 0$ and $W(\Sigma) = U$ does not contain a swappable pair then $W(\Sigma) = (Q_\pi(\Sigma), C)$.*

It remains to show that $\Phi^\tau$ can be constructed so that is also a bijection and that it satisfies the second condition of Lemma 26. To do so, consider the following algorithm.

- Let $\mathcal{X}_0 = \mathcal{X}^\tau$.
- While $k = \max_{\Sigma \in \mathcal{X}_p} k(\Sigma) \neq 0$
  - For each $\Sigma \in \mathcal{X}_p$: if $k(\Sigma) = k$ then swap the pair $(\mathbf{w}, \mathbf{y})$ at position $k$ in $\Sigma$, otherwise leave $\Sigma$ unchanged.
  - Let $\mathcal{X}_{p+1}$ the new set of trajectories.

For a witnesses sequence $W$ define $\mathcal{X}_p[W] = \{\Sigma \in \mathcal{X}_p \mid Q_\pi(\Sigma) = W\}$ for an index $p \geq 0$. Now the following lemma concludes the proof since $\mathcal{X}_0[W] \subseteq \mathcal{X}^\tau$ is valid. Its proof is identical (up to minor changes) to the proof of Lemma 28 in [24].

▶ **Lemma 30.** *If set $\mathcal{X}_p[W]$ is valid then so is $\mathcal{X}_{p+1}[W]$, and the mapping from $\mathcal{X}_p[W]$ to $\mathcal{X}_{p+1}[W]$ defined by the algorithm above is injective.*

## B Proof of Lemma 12

For the sake of brevity, we extend the notion of "addressing a flaw" to an arbitrary state $\sigma \in \Omega$, meaning that we recolor the vertices associated with the operation of addressing a flaw in the same way we would do it if the constraint corresponding to the flaw was indeed violated. Consider the following random experiments.

- Address $B_v$ at an arbitrary state $\sigma \in \Omega$ to get a state $\sigma'$. Let $\Pr_\sigma[B_v]$ denote the probability that $\sigma' \in B_v$.
- Address $Z_v$ at an arbitrary state $\sigma \in \Omega$ to get a state $\sigma'$. Let $\Pr_\sigma[Z_v]$ denote the probability that $\sigma' \in Z_v$.

Our claim now is that

$$\gamma(B_v) \leq \max_{\sigma' \in \Omega} \Pr_{\sigma'}[B_v] \; ; \tag{9}$$

$$\gamma(Z_v) \leq \max_{\sigma' \in \Omega} \Pr_{\sigma'}[Z_v] \; . \tag{10}$$

To see this, let $f_v \in \{B_v, Z_v\}$ and observe that

$$\gamma(f_v) = \max_{\sigma' \in \Omega} \sum_{\sigma \in f_v} \frac{\mu(\sigma)}{\mu(\sigma')} \rho_v(\sigma, \sigma') = \max_{\sigma' \in \Omega} \sum_{\sigma \in \mathrm{In}_{f_v}(\sigma')} \frac{1}{|\Lambda(\sigma)|} \; ,$$

where $\Lambda(\sigma) := \prod_{u \in N_v} L_u(\sigma)$ is the cartesian product of the lists of available colors of each vertex $u \in N_v$ at state $\sigma$ and $\mathrm{In}_{f_v}(\sigma')$ is the set of states $\sigma \in f_v$ such that $\sigma' \in A(f_v, \sigma)$.

The key observation now is that $\Lambda(\sigma') = \Lambda(\sigma)$ for each state $\sigma \in \mathrm{In}_{f_v}(\sigma)$. This is because any transition of the form $\sigma \xrightarrow{f_v} \sigma'$ does not alter the lists of available colors of vertices $u \in N_v$, since the graph is triangle-free. Thus,

$$\gamma(f_v) = \max_{\sigma' \in \Omega} \frac{|\mathrm{In}_{f_v}(\sigma')|}{\Lambda(\sigma')} = \max_{\sigma' \in \Omega} \Pr_{\sigma'}[f_v] \ ,$$

where the second equality follows from the fact that there is a bijection between $\mathrm{In}_{f_v}(\sigma')$ and the set of color assignments from $\Lambda(\sigma')$ to the vertices of $N_v$ that violate the constraint related to flaw $f_v$.

The following lemma concludes the proof.

▶ **Lemma 31** ([26]). *For every state $\sigma \in \Omega$ it holds that*

**(a)** $\Pr_\sigma[B_v] < \Delta^{-4}$;

**(b)** $\Pr_\sigma[Z_v] < \Delta^{-4}$.