Theoretical Paper Exposition : "A Constructive Proof of the General Lovász Local Lemma" by Robin Moser and Gábor Tardos

DD2440 - Honors Projects
Alex Martinez Rivera, alexmr@kth.se
Video available here

November 2025

1 Introduction

This article is a presentation of sections 1, 2 and 3, of the paper "A Constructive Proof of the General Lovász Local Lemma" by Robin Moser and Gábor Tardos [5].

I made a video that explains this presentation available here.

In this entire article, $\mathcal{A} = (A_i)_{i \leq n}$ is a finite collection of events in a probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

1.1 Some intuition

The events of $\mathcal{A}$ are called "bad events", and we want **none of them** to happen.
The goal is to find a condition P as weak as possible and some $\epsilon > 0$ such that:

$$(P) \implies \mathbb{P}\left(\bigwedge_{A \in \mathcal{A}} \overline{A}\right) \geq \epsilon \tag{1}$$

Remark 1. *If the events of $\mathcal{A}$ are mutually independent, and if no event is* certain*, the equation above is satisfied with $\epsilon = \prod_{A\in\mathcal{A}} (1 - \mathbb{P}(A))$. More formally we can state this as a lemma:*

Lemma 1 (Classic conditions for Equation 1)**.**

$$(P) := \mathcal{A} \text{ are mutually independent and } \forall A \in \mathcal{A}, \mathbb{P}(A) < 1$$

$$\epsilon := \prod_{A\in\mathcal{A}} (1 - \mathbb{P}(A))$$

Proof. Since the events $A_1, \ldots, A_n$ are mutually independent, the probability of their complement intersection factorizes:

$$\mathbb{P}\left(\bigwedge_{i=1}^{n} \overline{A_i}\right) = \prod_{i=1}^{n} \mathbb{P}(\overline{A_i}) = \prod_{i=1}^{n} (1 - \mathbb{P}(A_i)) .$$

By assumption, $\mathbb{P}(A_i) < 1$ for all i, so each term in the product is positive. Therefore, the probability that none of the bad events occur is strictly positive, and we can set

$$\epsilon := \prod_{i=1}^{n} (1 - \mathbb{P}(A_i)) > 0.$$

This proves the lemma. □

1.2 Erdős–Lovász Local Lemma

First, we define a *dependency graph* for the family of bad events.

Definition 1. *A graph $D = (\mathcal{A}, E)$ is called a* dependency graph *for the events $\mathcal{A} = \{A_1, \ldots, A_m\}$ if for every $i \in \{1, \ldots, m\}$, the event A_i is* ***independent of all events*** *$\{A_j : j \neq i,\ (A_i, A_j) \notin E\}$.*

Remark 2. *For a given set of events $\mathcal{A}$, there exists multiple* ***dependency graphs****.*

Moreover, given a vertex $A \in \mathcal{A}$, $\Gamma(A)$ is the set of all its neighbors, and $d(A) := |\Gamma(A)|$.

Using the setup of the previous paragraph, the General Erdős–Lovász Local Lemma can be stated as:

Lemma 2 (General Erdős–Lovász conditions for Equation 1)**.**

$$(P) := \exists x : \mathcal{A} \to (0,1), \left[\forall A \in \mathcal{A}, \left(\mathbb{P}(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)) \right) \right]$$

$$\epsilon := \prod_{i=1}^{n} (1 - x(A_i))$$

Remark 3. *This lemma generalizes the case of mutually independent events: if no dependencies exist, we simply use* $x(A) := \mathbb{P}(A)$*.*

Given $p \in (0,1)$ and $d \in \mathbb{N}$, a less general version of Erdős–Lovász Local Lemma is the following:

Lemma 3 (Symmetrical Erdős–Lovász conditions for Equation 1)**.**

$$(P) := \left[\forall A \in \mathcal{A}, \begin{cases} \mathbb{P}(A) \leq p \\ d(A) \leq d \end{cases} \quad and \quad ep(d+1) \leq 1 \right]$$

$$\epsilon := \left(\frac{d}{d+1} \right)^n$$

1.3 The goal of this article

The problem with Erdős–Lovász Local Lemmas, is that they have only been shown theoretically, not constructively. It means that the proof does not provide an event $E \in \mathcal{F}$ such that $\mathbb{P}(E) > 0$ and $\forall A \in \mathcal{A}, E \subset \overline{A}$.

The main goal of the paper "A Constructive Proof of the General Lovász Local Lemma" by Robin Moser and Gábor Tardos [5] is to find a constructive proof of another version of the General Erdős–Lovász Local Lemma, yet slightly less powerful than the original lemma (see Lemma 2). Here is the formulation:

The variable-based setting

Let $\mathcal{X}$ be a finite collection of mutually independent random variables : $\mathcal{X} := (X_1, ..., X_m)$. We then assume that:

$$\forall i \in \{1, ..., n\}, A_i := \texttt{test}(i, \mathcal{X}) \tag{2}$$

where `test` is a computable function that returns a boolean.

Definition 2 (Violation of a "bad" event)**.**
We say that

$$\textit{An evaluation } (x_1, ..., x_m) \textit{ of } \mathcal{X} \textbf{ \textit{violates}} \ A_i \in \mathcal{A}$$

if

$$\texttt{test}(i, (x_1, ..., x_m)) \textit{ is true.}$$

Definition 3 (Variables of a "bad" event)**.**
For a given $i \leq n$, A_i does not neceserally depend on all variables of $\mathcal{X}$. We call ***variables*** *of A_i, the minimal subset $\mathrm{vbl}(A_i) \subset \mathcal{X}$ that A_i depends on.*

Now, the *dependency graph* $D = (\mathcal{A}, E)$ is a little bit easier to define:

$$A_i \overset{E}{\sim} A_j \iff \mathrm{vbl}(A_i) \cap \mathrm{vbl}(A_j) \neq \emptyset \tag{3}$$

Finally, the lemma we will consider is the following:

Lemma 4 (Moser and Tardos conditions for Equation 1)**.**

$$(P) := \begin{cases} \mathcal{A} \textit{ depends on } \mathcal{X} \textit{ as defined above, and} \\ \exists x : \mathcal{A} \to (0,1), \left[\forall A \in \mathcal{A}, \left(\mathbb{P}(A) \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B))\right)\right] \end{cases}$$

$$\epsilon := \prod_{i=1}^{n} (1 - x(A_i))$$

This lemma has already been proven by Erdős and Lovász [5], since it cearly is a particular case of the original lemma (Lemma 2). But remember, our goal is to find a constructive proof, i.e. an algorithm that gives us an evaluation of $\mathcal{X}$ that doesn't violate any event of $\mathcal{A}$.

This algorithm will be presented in section 3. But first, we start with a *State of the Art* that shows how the paper from Moser and Tardos is groundbreaking.

2 State of the Art

2.1 The k-hypergraph 2-coloring setting

Definition 4. *A k-uniform hypergraph $H = (V, E)$ is defined by:*

- *The set V is a finite set of vertices,*
- *An edge $e \in E$ is a set of exactly k elements of V.*

Given a k-uniform hypergraph $H = (V, E)$, we assume that each vertex is randomly 2-colored with uniform probability $\frac{1}{2}$ either *red* or *blue*.

The set $\mathcal{A}$ of "bad events" is

$$\mathcal{A} = \{A_e := \text{"edge } e \text{ is monochromatic"} \mid e \in E\}, \tag{4}$$

For a fixed edge $e = \{v_1, \ldots, v_k\}$, the probability that it is monochromatic is

$$\mathbb{P}(A_e) \;=\; \mathbb{P}(\text{all red}) + \mathbb{P}(\text{all blue}) \;=\; 2 \cdot 2^{-k} \;=\; 2^{1-k}.$$

Thus each bad event has probability

$$p = 2^{1-k}.$$

Two bad events A_e and $A_{e'}$ are independent unless the edges e and e' share at least one vertex. Let d denote the maximum number of edges that intersect any given edge. Then every bad event is independent of all but at most d others.

Applying the symmetric Lovász Local Lemma (Lemma 3) with $p = 2^{1-k}$ yields the condition

$$e \cdot 2^{1-k} \cdot (d+1) \;\leq\; 1,$$

which is equivalent to

$$d + 1 \;\leq\; \frac{2^{k-1}}{e}.$$

Thus, whenever every edge of the hypergraph intersects at most $\frac{2^{k-1}}{e} - 1$ other edges, the probability that no edge is monochromatic is positive, and therefore:

Corollary 1 (Hypergraph 2-coloring statement)**.**
If every edge of a k-uniform hypergraph intersects fewer than $\frac{2^k}{2e}$ other edges, then there exists a 2-coloring of the vertices such that no edge is monochromatic.

We just proved this corollary using the non constructive Lovász Local Lemma when $d \leq \frac{2^k}{2e}$. But could we aim towards a constructive proof ? For now, some constructive proofs have been found, but only when $d \leq c$, where c is some constant $< \frac{2^k}{2e}$.

2.2 Timeline of the yet-discovered constructive proofs

Date	Authors	Maximal dependency degree	Paper
1991	József Beck	$d \leq O(2^{k/48})$	[2]
1991	Noga Alon	$d \leq O(2^{k/8})$	[1]
2000	Artur Czumaj & Christian Scheideler	$d \leq O(2^{k/8})$	[3]
2008	Aravind Srinivasan	$d \leq O(2^{k/4})$	[6]
2009	Robin A. Moser	$d \leq O(2^{k/2})$	[4]

These bounds are weaker than the existential bound $\sim 2^k/e$ guaranteed by the non-constructive LLL. [1]

The paper of Robin Moser and Gábor Tardos from 2010 [5], finally reached a constructive proof of the Hypergraph 2-coloring statement (see Corollary 1).

But how ?

3 Moser and Tardos's constructive proof

In this section, we consider the variable-based setting defined in subsection 1.3.

3.1 The Algorithm and the Theorem

Moser and Tardos came up with the following algorithm:

[1] The result from 2000 by Artur Czumaj and Christian Scheideler[3], is an extension for non-uniform hypergraphs

Algorithm 1 The Sequential Solver

```
1: function SEQUENTIAL_LLL(𝒳, 𝒜)
2:     for all X ∈ 𝒳 do
3:         j ← index of X in 𝒳
4:         ν_j ← a random evaluation of X
5:     end for
6:     while ∃A ∈ 𝒜, A is violated do
7:         for all X ∈ vbl(A) do
8:             j ← index of X in 𝒳
9:             ν_j ← a new random evaluation of X
10:        end for
11:    end while
12:    return (ν_j)_{j≤m}
13: end function
```

Theorem 1. *If there exists an assignment of reals $x : \mathcal{A} \to (0,1)$ such that*

$$\forall A \in \mathcal{A}: \quad \Pr[A] \leq x(A) \prod_{B \in \Gamma(A)} (1 - x(B)),$$

then there exists an assignment of values to the variables $\mathcal{X}$ not violating any of the events in $\mathcal{A}$. Moreover, the randomized algorithm described above resamples an event $A \in \mathcal{A}$ at most an expected $x(A)/(1-x(A))$ times before it finds such an evaluation. Thus, the expected total number of resampling steps is at most

$$\sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}.$$

3.2 The Proof

What's very powerful in the variable-based setting of Moser and Tardos (see 1.3), is that we have the following lemma:

Lemma 5 (Characterization of dependencies in the variable-based setting)**.** *A graph $D = (\mathcal{A}, E)$ is a valid **dependency graph** (see Definition 1) if and only if*

$$\mathrm{vbl}(A_i) \cap \mathrm{vbl}(A_j) \neq \varnothing \quad \Longrightarrow \quad (A_i, A_j) \in E.$$

3.2.1 Witness Trees and Their Construction

Definition 5 (Execution Log)**.** *During the execution of the Moser–Tardos algorithm, each time a violated event is resampled, we record the identity of that event.*

Formally, the execution log *is the infinite (or finite) sequence*

$$C(1), C(2), C(3), \dots$$

where $C(t) \in \mathcal{A}$ denotes the event resampled at time step t. We call C the log *of the resampling algorithm.*

The purpose of witness trees is to record the "causal history" explaining why a given resampling step occurred. We now formalize their structure.

Definition 6 (Witness Tree)**.** *A* witness tree *is a pair $\tau = (T, \sigma_T)$, where:*

- *T is a finite rooted tree,*
- *$\sigma_T : V(T) \to \mathcal{A}$ is a labelling of nodes by bad events,*

such that for every vertex $u \in V(T)$, every child v of u satisfies

$$\sigma_T(v) \in \Gamma^+(\sigma_T(u)),$$

that is, each child is labeled with either the same event or a neighbor of it in the dependency graph. A witness tree is called proper *if siblings always receive distinct labels.*

To simplify notation, for $v \in V(T)$ we write $[v] := \sigma_T(v)$ and $V(\tau) := V(T)$.

We now describe how to construct, from the execution log C and a fixed time t, the canonical witness tree associated with the resampling step t.

Definition 7 (Construction of the Witness Tree $\tau_C(t)$)**.** *Fix a time step $t \geq 1$ and let C be the resampling log.*

We construct a sequence of trees

$$\tau_C^{(t)}(t),\ \tau_C^{(t-1)}(t),\ \dots,\ \tau_C^{(1)}(t),$$

defined inductively backwards in time as follows.

(1) Initialization. *Define $\tau_C^{(t)}(t)$ to be the one-vertex tree whose root is labeled $C(t)$.*

(2) Backward extension. *For $i = t-1, t-2, \ldots, 1$, consider the event $C(i)$. We distinguish two cases:*

- ***Case A.*** *There exists a vertex $v \in V(\tau_C^{(i+1)}(t))$ such that*

$$C(i) \in \Gamma^+([v]).$$

Among all such vertices, choose one having maximum distance *from the root of $\tau_C^{(i+1)}(t)$. Attach a new child u to this vertex v, and label it by*

$$[u] := C(i).$$

The resulting tree is defined to be $\tau_C^{(i)}(t)$.

- ***Case B.*** *No vertex v satisfies $C(i) \in \Gamma^+([v])$. In this case, do nothing and define*

$$\tau_C^{(i)}(t) := \tau_C^{(i+1)}(t).$$

(3) Final definition. *The witness tree associated with the resampling at time t is*

$$\tau_C(t) := \tau_C^{(1)}(t).$$

Definition 8 (Occurrence of a Witness Tree)**.** *A witness tree τ is said to* occur in the log C *if there exists a time t such that*

$$\tau_C(t) = \tau.$$

You can follow the steps of the construction above, and you should get the following:

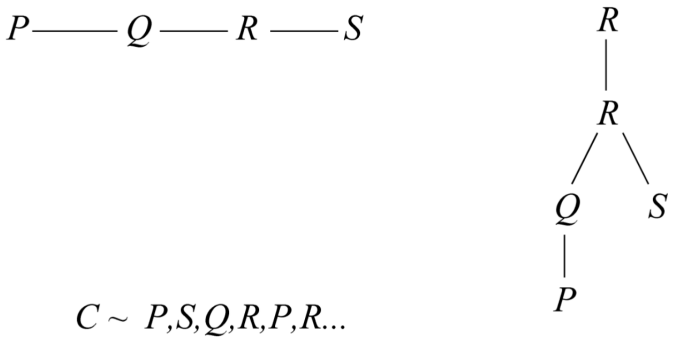

Figure 1: An example of a witness tree constructed from a resampling log.

Then, in the paper [5] they prove the following lemma (that we are not proving here):

Lemma 6 (Witness Tree Properties)**.** *Let τ be a fixed witness tree and C the (random) log produced by the algorithm. Then:*

1. *If τ occurs in C, then τ is proper.*
2. *The probability that τ appears in C satisfies*

$$\Pr[\tau \text{ occurs in } C] \leq \prod_{v \in V(\tau)} \Pr([v]).$$

The following explanation from Moser and Tardos illustrates how the resampling of an event A can be analyzed in terms of witness trees and how this leads to a bound on the expected number of resamplings.

Let C be the *log* of the execution of the algorithm of subsection 3.1.

> For any event $A \in \mathcal{A}$, let N_A denote the random variable that counts how many times the event A is resampled during the execution of the algorithm, that is, the number of time steps t with $C(t) = A$. Let t_i denote the i-th such time step.
>
> Let $\mathcal{T}_A$ denote the set of all proper witness trees having the root labelled A. As we have previously shown, $\tau_C(t_i) \in \mathcal{T}_A$ for all i. Moreover, for each i, the tree $\tau_C(t_i)$ contains exactly i vertices labelled A, and thus $\tau_C(t_i) \neq \tau_C(t_j)$ unless $i = j$.

From this we conclude that N_A not only counts the number of occurrences of A in the log, but also coincides with the number of distinct proper witness trees occurring in C that have their root labeled A, that is,

$$N_A = \sum_{\tau \in \mathcal{T}_A} 1_{\{\tau \text{ occurs in } C\}}.$$

Therefore, one can bound the expectation of N_A simply by summing the bounds in Lemma 6 on the probabilities of the occurrences of the different proper witness trees.

What we have so far

- For each event $A \in \mathcal{A}$, we defined the random variable N_A counting the number of times A is resampled.
- We showed that N_A equals the number of *distinct proper witness trees* rooted at A that occur in the resampling log:

$$N_A = \sum_{\tau \in \mathcal{T}_A} 1_{\{\tau \text{ occurs in } C\}}.$$

Finally, we now need to prove that $\mathbb{E}(N_A)$ is bounded.

3.2.2 A bound for $\mathbb{E}(N_A)$

Our goal is to bound the expected number of times a given event A is resampled: $\mathbb{E}(N_A)$. Thanks to Lemma 6, we have a *probabilistic bound* for a single witness tree:

$$\Pr[\tau \text{ occurs in } C] \leq \prod_{v \in V(\tau)} \Pr([v]).$$

That's why we need to come up with a **Random Generation of Witness Trees** [2].

First, we need to introduce a *multitype Galton–Watson branching process* for generating proper witness trees rooted at a fixed event $A \in \mathcal{A}$. This process will allow us to systematically bound the probabilities of all witness trees.

[2] This is the title of the third section of Moser and Tardos' paper [5]

Definition 9 (Galton–Watson Process for Witness Trees)**.**

1. ***Initialization.*** *Create a root vertex labeled A.*
2. ***Branching step.*** *For each vertex v produced in the previous round, and for each event $B \in \Gamma^+([v])$, independently:*
   - *Add a child vertex labeled B with probability $x(B)$.*
   - *Skip adding a child labeled B with probability $1 - x(B)$.*
3. ***Iteration.*** *Repeat the branching step for newly added vertices in the next round.*
4. ***Termination.*** *The process stops naturally if no new vertices are added in a round.*

Moser and Tardos then stated the following lemma:

Lemma 7 (Probability of Generating a Fixed Proper Witness Tree)**.** *Let τ be a fixed proper witness tree with root labeled A. Let p_τ denote the probability that the Galton–Watson process above produces exactly τ. Then*

$$p_\tau = \frac{1 - x(A)}{x(A)} \prod_{v \in V(\tau)} x'([v]).$$

where:

$$x'(B) := x(B) \prod_{C \in \Gamma(B)} (1 - x(C)).$$

The proof of this lemma is admitted[3].

3.2.3 Conclusion

Using Lemma 7, we can bound the expected number of resamplings of A:

$$\mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\tau \text{ occurs in } C] \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'([v]) = \frac{x(A)}{1 - x(A)} \sum_{\tau \in \mathcal{T}_A} p_\tau \leq \frac{x(A)}{1 - x(A)}.$$

This provides a concrete upper bound on the expected number of resamplings of A, and summing over all $A \in \mathcal{A}$ gives an expected bound on the total number of resampling steps, that proves Theorem 1.

[3]Again, I don't write the proof of the lemma here, since I don't want this paper to be too long. The proof is available in the third section of Moser and Tardos' paper [5], and I try to explain it as well as possible in the video available here.

References

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

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

[3] Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proceedings of the 11th ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 30–39, 2000.

[4] Robin A. Moser. A constructive proof of the Lovász local lemma. *Journal of the ACM*, 57(2):11:1–11:15, 2008. Earlier conference version appeared in STOC 2009. doi:10.1145/1614320.1614321.

[5] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11:1–11:15, jan 2010. doi:10.1145/1667053.1667060.

[6] Aravind Srinivasan. Improved constructive bounds for the Lovász local lemma. *Random Structures & Algorithms*, 2008. Achieves approximately $d = O(2^{k/4})$ in certain settings.