Advanced Algorithms
Lovász Local Lemma
**k-SAT**

- Conjunctive Normal Form (CNF):
  \[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \]

  Boolean variables: \( x_1, x_2, \ldots, x_n \in \{\text{True, False}\} \)

- \(k\)-CNF: each clause contains exactly \(k\) variables

**Problem (k-SAT)**

**Input:** \(k\)-CNF formula \(\Phi\).

**Output:** determine whether \(\Phi\) is satisfiable.

- [Cook-Levin] \textbf{NP-hard} if \(k \geq 3\)
Problem \( (k\text{-SAT}) \)
Input: \( k\text{-CNF} \) formula \( \Phi \).
Output: determine whether \( \Phi \) is satisfiable.

- Clauses are \textit{disjoint}:

\[
\Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_4 \lor x_5 \lor x_6) \land (x_7 \lor \neg x_8 \lor \neg x_9)
\]

resolve each clause \textit{independently} \( (\Phi \text{ is always satisfiable!}) \)

- \( m < 2^k \implies \Phi \text{ is always satisfiable.} \)

\( m \): \# of clauses
The Probabilistic Method

- \(k\)-CNF:  \(\Phi = C_1 \land C_2 \land \cdots \land C_m\)

\[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \]

- Draw uniform random \(x_1, x_2, \ldots, x_n \in \{\text{True, False}\}\)

- **Bad event** \(A_i\): clause \(C_i\) is violated

  \[ \forall 1 \leq i \leq m, \quad \Pr[A_i] = 2^{-k} \]

- **Union bound:**  \(\Pr \left[ \bigvee_{i=1}^{m} A_j \right] \leq \sum_{i=1}^{m} \Pr[A_i] = m2^{-k} \)

\[ m < 2^k \implies \Pr \left[ \bigwedge_{i=1}^{m} \overline{A}_i \right] > 0 \implies \Phi \text{ is satisfiable!} \]

**disjoint clauses \(\implies\)** \(\Pr \left[ \bigwedge_{i=1}^{m} \overline{A}_i \right] = \prod_{i=1}^{m} (1 - \Pr[A_i]) > 0\)

(independent bad events)
The Probabilistic Method

Problem (\(k\)-SAT)

Input: \(k\)-CNF formula \(\Phi\).
Output: determine whether \(\Phi\) is satisfiable.

- uniform random \(x_1, \ldots, x_n \in \{\text{True}, \text{False}\}\)

\[\text{disjoint clauses \ or } \quad m < 2^k\]

\[\implies \Pr[\Phi(x) = \text{True}] > 0\]

\[\exists x \in \{T, F\}^n \quad \Phi(x) = \text{True}\]

The Probabilistic Method:

Draw \(x\) from prob. space \(\Omega\): for property \(\mathcal{P}\),

\[\Pr[\mathcal{P}(x)] > 0 \implies \exists x \in \Omega, \mathcal{P}(x)\]
Limited Dependency

- **$k$-CNF**: $\Phi = C_1 \land C_2 \land \cdots \land C_m$

  $\Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5)$

**Dependency degree $d$**: each clause intersects $\leq d$ other clauses

- uniform random $x_1, \ldots, x_n \in \{T, F\}$, each clause is violated w.p. $2^{-k}$

  (union bound) $m2^{-k} < 1$

  $d2^{-k} < 1$ ("local" union bound?)

  $\Rightarrow \Pr[ \text{no clause is violated} ] > 0$

  (LLL) $e(d + 1)2^{-k} \leq 1$

  $4d2^{-k} \leq 1$
Lovász Local Lemma (LLL)

- “Bad” events $A_1, \ldots, A_m$, where all $\Pr[A_j] \leq p$

**Dependency degree** $d$:

Each $A_i$ is “dependent” of $\leq d$ other events

**Lovász Local Lemma** [Lovász and Erdős 1973; Lovász 1977]:

\[ ep(d + 1) \leq 1 \implies \Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right] > 0 \]

- The “LLL” condition:
  - Bad events are not too likely to occur individually.
  - Bad events are not too dependent with each other.
Dependency Graph

• “Bad” events $A_1, \ldots, A_m$,

**Dependency degree $d$:**

 each $A_i$ is **mutually independent** of all except $\leq d$ other events

**Definition (independence):**

$A$ is **independent** of $B$ if $\Pr[A | B] = \Pr[A]$ or $B$ is impossible.

$A_0$ is **mutually independent** of $A_1, \ldots, A_m$ if $A_0$ is independent of every event $B = B_1 \land \cdots \land B_m$, where each $B_i = A_i$ or $\overline{A}_i$. 

**Dependency Graph**

- “Bad” events $A_1, \ldots, A_m$.

**Dependency degree $d$:** (max-degree of dependency graph)
  
each $A_i$ is **mutually independent** of all except $\leq d$ other events

**Dependency graph:**

- Vertices are bad events $A_1, \ldots, A_m$.
- Each $A_i$ is mutually independent of non-adjacent events.

**Independent random variables:**

$$X_1, X_2, X_3, X_4$$

**Bad events** (defined on subsets of variables):

- $A(X_1, X_2), B(X_2, X_3), C(X_3, X_4), D(X_4), E(X_1, X_4)$
Lovász Local Lemma (LLL)

- $A_1, \ldots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$:

  $$A_i \text{ is mutually independent of all } A_j \notin \Gamma(A_i).$$

---

**Lovász Local Lemma:**

$$p \triangleq \max_i \Pr[A_i] \text{ and } d \triangleq \max_i |\Gamma(A_i)|$$

$$ep(d + 1) \leq 1 \implies \Pr \left[ \bigwedge_{i=1}^m \overline{A_i} \right] > 0$$
Constraint Satisfaction Problem (CSP)

- **Variables:** $x_1, \ldots, x_n \in [q]$

- **(local) Constraints:** $C_1, \ldots, C_m$
  - each $C_i$ is defined on a subset $\text{vbl}(C_i)$ of variables
    $$C_i : [q]^\text{vbl}(C_i) \rightarrow \{\text{True, False}\}$$

- Any $x \in [q]^n$ is a CSP solution if it satisfies all $C_1, \ldots, C_m$

- **Examples:**
  - $k$-CNF, (hyper)graph coloring, set cover, unique games…
  - vertex cover, independent set, matching, perfect matching, …
Hypergraph Coloring

- $k$-uniform hypergraph $H = (V, E)$:
  - $V$ is vertex set, $E \subseteq \binom{V}{k}$ is set of hyperedges
- degree of vertex $v \in V$: # of hyperedges $e \ni v$
- proper $q$-coloring of $H$:
  - $f: V \rightarrow [q]$ such that no hyperedge is monochromatic
    \[ \forall e \in E, \quad |f(e)| > 1 \]

**Theorem:** For any $k$-uniform hypergraph $H$ of max-degree $\Delta$,
\[ \Delta \leq \frac{q^{k-1}}{ek} \implies H \text{ is } q\text{-colorable} \]

\[ k \geq \log_q \Delta + \log_q \log_q \Delta + O(1) \]
Hypergraph Coloring

**Theorem:** For any $k$-uniform hypergraph $H$ of max-degree $\Delta$,

$$\Delta \leq \frac{q^{k-1}}{ek} \implies H \text{ is } q\text{-colorable}$$

- Uniformly and independently color each $v \in V$ a random color $\in [q]$
- Bad event $A_e$ for each hyperedge $e \in E \subseteq \binom{V}{k}$: $e$ is monochromatic
  - $\Pr[A_e] \leq p = q^{1-k}$
- Dependency degree for bad events $d \leq k(\Delta - 1)$
  - $\Delta \leq \frac{q^{k-1}}{ek} \implies ep(d + 1) \leq 1 \quad \text{Apply LLL}$
Lovász Local Lemma (LLL)

- $A_1, \ldots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$

**Lovász Local Lemma (symmetric case):**

$p \triangleq \max_i \Pr[A_i]$ and $d \triangleq \max_i |\Gamma(A_i)|$

$$ep(d + 1) \leq 1 \implies \Pr[\bigwedge_{i=1}^m \overline{A}_i] > 0$$

$$\alpha_1 = \cdots = \alpha_m = \frac{1}{d + 1}$$

**Lovász Local Lemma (asymmetric case):**

$\exists \alpha_1, \ldots, \alpha_m \in [0,1]:$

$$\forall i, \ \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \implies \Pr \left[ \bigwedge_{i=1}^m \overline{A}_i \right] \geq \prod_{i=1}^m (1 - \alpha_i)$$
• $A_1, \ldots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$

**Lovász Local Lemma (asymmetric case):**

\[ \exists \alpha_1, \ldots, \alpha_m \in [0,1) : \]

\[ \forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \implies \Pr\left[ \bigwedge_{i=1}^{m} \overline{A}_i \right] \geq \prod_{i=1}^{m} (1 - \alpha_i) \]

**chain rule**

\[ \Pr\left[ \bigwedge_{i=1}^{m} \overline{A}_i \right] = \prod_{i=1}^{m} \Pr\left[ \overline{A}_i \mid \bigwedge_{j<i} \overline{A}_j \right] = \prod_{i=1}^{m} \left(1 - \Pr\left[ A_i \mid \bigwedge_{j<i} \overline{A}_j \right]\right) \geq \prod_{i=1}^{m} (1 - \alpha_i) \]

**Induction Hypothesis (I.H.):**

\[ \forall \text{ distinct } A_i, A_{j_1}, A_{j_2}, \ldots, A_{j_k} : \quad \Pr\left[ A_i \mid \overline{A}_{j_1} \cdots \overline{A}_{j_k} \right] \leq \alpha_i \]

**Basis:** when $k = 0$, trivial
LLL cond.: \[ \exists \alpha_1, \ldots, \alpha_m \in [0,1) \text{ s.t.} \]
\[ \forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \]

I.H.: \[ \Pr \left[ A_i \mid \overline{A}_{j_1} \cdots \overline{A}_{j_k} \right] \leq \alpha_i \quad \text{holds for all smaller } k \]

Say \( A_{j_1}, \ldots, A_{j_l} \in \Gamma(A_i), \quad A_{j_{l+1}}, \ldots, A_{j_k} \notin \Gamma(A_i) \quad (l \geq 1 \text{ or else trivial}) \)

\[ \Pr \left[ A_i \mid \overline{A}_{j_1} \cdots \overline{A}_{j_l} \overline{A}_{j_{l+1}} \cdots \overline{A}_{j_k} \right] = \frac{\Pr \left[ A_i \overline{A}_{j_1} \cdots \overline{A}_{j_l} \mid \overline{A}_{j_{l+1}} \cdots \overline{A}_{j_k} \right]}{\Pr \left[ \overline{A}_{j_1} \cdots \overline{A}_{j_l} \mid \overline{A}_{j_{l+1}} \cdots \overline{A}_{j_k} \right]} \leq \alpha_i \]

\[ \leq \Pr \left[ A_i \mid \overline{A}_{j_{l+1}} \cdots \overline{A}_{j_k} \right] = \Pr \left[ A_i \right] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \quad (\text{LLL cond.}) \]

\[ = \prod_{r=1}^{l} \Pr \left[ \overline{A}_{j_r} \mid \overline{A}_{j_{r+1}} \cdots \overline{A}_{j_k} \right] = \prod_{r=1}^{l} \left( 1 - \Pr \left[ A_{j_r} \mid \overline{A}_{j_{r+1}} \cdots \overline{A}_{j_k} \right] \right) \]

\[ (\text{I.H.}) \geq \prod_{r=1}^{l} \left( 1 - \alpha_{j_r} \right) \geq \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \]
• $A_1, \ldots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$

**Lovász Local Lemma (asymmetric case):**

\[
\exists \alpha_1, \ldots, \alpha_m \in [0,1) : \\
\forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \implies \Pr\left[\bigwedge_{i=1}^{m} A_i\right] \geq \prod_{i=1}^{m} (1 - \alpha_i)
\]

**Induction Hypothesis (I.H.):**

\[
\forall \text{ distinct } A_i, A_{j_1}, A_{j_2}, \ldots, A_{j_k} : \quad \Pr\left[A_i \mid \bigwedge_{j<i} A_j \right] \leq \alpha_i
\]

**Chain rule**

\[
\Pr\left[\bigwedge_{i=1}^{m} A_i\right] = \prod_{i=1}^{m} \Pr\left[A_i \bigwedge_{j<i} \overline{A}_j\right] = \prod_{i=1}^{m} \left(1 - \Pr\left[A_i \bigwedge_{j<i} \overline{A}_j\right]\right) \geq \prod_{i=1}^{m} (1 - \alpha_i)
\]
Lovász Local Lemma (LLL)

- \( A_1, \ldots, A_m \) has a dependency graph given by neighborhoods \( \Gamma(\cdot) \)

\[ \text{Lovász Local Lemma (asymmetric case):} \]

\[
\exists \alpha_1, \ldots, \alpha_m \in [0,1): \quad \forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \quad \implies \quad \Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right] > 0
\]

\[
\alpha_1 = \cdots = \alpha_m = \frac{1}{d+1} \quad \Rightarrow \quad \alpha_1 = \cdots = \alpha_m = \frac{1}{2d}
\]

\text{symmetric case:}

\[
p \triangleq \max_i \Pr[A_i] \quad \text{or} \quad 4pd \leq 1
\]

\[
\implies \quad \Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right] > 0
\]

\[
p \triangleq \max_i \Pr[A_i] \quad \text{or} \quad 4pd \leq 1 \quad \implies \quad \Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right] > 0
\]
Lovász Local Lemma (LLL)

- $A_1, \ldots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$

symmetric case:

\[ p \triangleq \max_i \Pr[A_i] \]
\[ d \triangleq \max_i |\Gamma(i)| \]

\[
\begin{align*}
ep(d + 1) &\leq 1 \\
4pd &\leq 1
\end{align*}
\]

\[ \Rightarrow \Pr \left[ \bigwedge_{i=1}^m \overline{A_i} \right] > 0 \]

asymmetric case:

\[ \Rightarrow \Pr \left[ \bigwedge_{i=1}^m \overline{A_i} \right] > 0 \]
Summary

• LLL establishes the following implication:

\[
\text{LLL condition} \rightarrow \text{It's possible that none of } A_1, \ldots, A_m \text{ occurs}
\]

taking only the probabilities and dependency graph of \( A_1, \ldots, A_m \)

• What’s next:

  • tight(er) LLL condition: **Shearer’s bound**
  
  • tighter bounds when more (than just local dependency structure) are known: the probabilistic method beyond LLL
  
  • beyond existence: **algorithmic/constructive LLL**
Algorithmic

Lovász Local Lemma:
The Moser-Tardos Algorithm
Algorithmic LLL (abstract version)

• “Bad” events $A_1, \ldots, A_m$ in a probability space $(\Omega, \Sigma, \Pr)$

\[ \exists \alpha_1, \ldots, \alpha_m \in [0,1) : \]
\[ \forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \]
\[ \exists \sigma \in \Omega, \quad \text{avoid all } A_1, \ldots, A_m \]

• **Algorithmic** (constructive) Lovász Local Lemma:

  Give an efficient algorithm:

  find such a good sample $\sigma \in \Omega$
  avoiding all bad events $A_1, \ldots, A_m$
Algorithmic LLL (variable version)

Variable framework for LLL (CSP with independent variables):

- mutually independent random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$
- bad events $\mathcal{A} = \{A_1, \ldots, A_m\}$, where $A_i \in \mathcal{A}$ is determined by $\text{vbl}(A_i) \subseteq \mathcal{X}$

- dependency graph is given by neighborhoods $\Gamma(\cdot)$:
  \[
  \Gamma(A_i) = \left\{ A_j \neq A_i \mid \text{vbl}(A_i) \cap \text{vbl}(A_j) \neq \emptyset \right\}
  \]

- Algorithmic (constructive) Lovász Local Lemma:
  
  Give an efficient algorithm (CSP solver):

  find such a good evaluation of $X_1, \ldots, X_n$
  avoiding all bad events $A_1, \ldots, A_m$
The Moser-Tardos Algorithm

Variable framework for LLL (CSP with independent variables):
- mutually independent random variables \( \mathcal{X} = \{X_1, \ldots, X_n\} \)
- bad events \( \mathcal{A} = \{A_1, \ldots, A_m\} \), where \( A_i \in \mathcal{A} \) is determined by \( \text{vbl}(A_i) \subseteq \mathcal{X} \)

Moser-Tardos Algorithm:
draw independent samples of \( X_1, \ldots, X_n \);
while \( \exists \) a bad event \( A_i \) that occurs:
    resample all \( X_j \in \text{vbl}(A_i) \);

Assume oracles for:
- draw ind. samples of \( X_j \)
- check if \( A_i \) occurs

Theorem [Moser-Tardos 2010]:
\[ \exists \alpha_1, \ldots, \alpha_m \in [0,1) : \]
\[ \forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \]

The Moser-Tardos algorithm terminates within \( \sum_{i=1}^{m} \frac{\alpha_i}{1 - \alpha_i} \) resamples in expectation
The Moser-Tardos Algorithm

Variable framework for LLL (CSP with independent variables):
- mutually independent random variables \( \mathcal{X} = \{X_1, \ldots, X_n\} \)
- bad events \( \mathcal{A} = \{A_1, \ldots, A_m\} \), where \( A_i \in \mathcal{A} \) is determined by \( \text{vbl}(A_i) \subseteq \mathcal{X} \)

Moser-Tardos Algorithm:
draw independent samples of \( X_1, \ldots, X_n \);
while \( \exists \) a bad event \( A_i \) that occurs:
  resample all \( X_j \in \text{vbl}(A_i) \);

Assume oracles for:
- draw ind. samples of \( X_j \)
- check if \( A_i \) occurs

Theorem [Moser-Tardos 2010]:
\[
p \triangleq \max_i \Pr[A_i], \quad d \triangleq \max_i |\Gamma(A_i)| \\
ep(d + 1) \leq 1
\]
The Moser-Tardos algorithm terminates within \( m/d \) resamples in expectation
Moser-Tardos Algorithm:
draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
  resample all $X_j \in \text{vbl}(A_i)$;

Execution Log (exe-log) $\Lambda$ of the M-T algorithm:

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
random sequence of resampled bad events

Theorem [Moser-Tardos 2010]:

$\exists \alpha_1, \ldots, \alpha_m \in [0,1) :$

$\forall i,$ $\Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$

$\forall i,$ $\mathbb{E}_{\Lambda} \left[ \# \text{ of } A_i \text{ in } \Lambda \right] \leq \frac{\alpha_i}{1 - \alpha_i}$
## Resampling Table

**Moser-Tardos Algorithm:**

- draw independent samples of $X_1, \ldots, X_n$;
- while $\exists$ a bad event $A_i$ that occurs:
  - resample all $X_j \in \text{vbl}(A_i)$;

**Exe-log $\Lambda$:**

$$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$$

random sequence of resampled bad events

**Exe-log $\Lambda$:**

$D, C, E, D, B, A, C, A, D, \ldots$

**Resampling table:**

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_1^{(0)}$, $X_1^{(1)}$, $X_1^{(2)}$, $X_1^{(3)}$, $X_1^{(4)}$, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_2$</td>
<td>$X_2^{(0)}$, $X_2^{(1)}$, $X_2^{(2)}$, $X_2^{(3)}$, $X_2^{(4)}$, ...</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$X_3^{(0)}$, $X_3^{(1)}$, $X_3^{(2)}$, $X_3^{(3)}$, $X_3^{(4)}$, ...</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$X_4^{(0)}$, $X_4^{(1)}$, $X_4^{(2)}$, $X_4^{(3)}$, $X_4^{(4)}$, ...</td>
</tr>
</tbody>
</table>

$X_j^{(t)}$: $t$-th sampling of variable $X_j$
Witness Tree

Moser-Tardos Algorithm:

- draw independent samples of $X_1, \ldots, X_n$;
- while $\exists$ a bad event $A_i$ that occurs:
  - resample all $X_j \in vbl(A_i)$;

Exe-log $\Lambda$:

\[ \Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{ A_1, \ldots, A_m \} \]
random sequence of resampled bad events

Witness tree $T(\Lambda, t)$:
- each node $u$ with label $A_{[u]} \in \mathcal{A}$, siblings have distinct labels
  - initially, $T$ contains a single root $r$ with $\Lambda_t$
  - for $i = t - 1$ to 1:
    - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
      - add child $v \rightarrow$ deepest such $u$, labeled with $\Lambda_i$
  - $T(\Lambda, t)$ is the resulting $T$

Inclusive neighborhood:

\[ \Gamma^+(A) \triangleq \Gamma(A) \cup \{ A \} \]
Witness tree $T(\Lambda, t)$: each node $u$ with label $A_{[u]} \in \mathcal{A}$, siblings have distinct labels

- initially, $T$ contains a single root $r$ with $\Lambda_t$
- for $i = t - 1$ to $1$:
  - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
    - add child $v \rightarrow$ deepest such $u$, labeled with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$
dependency graph:

exe-log $\Lambda$: D, C, E, D, B, A, C, A, D, ...

$T(\Lambda, 8)$:

$T(\Lambda, 9)$:

Witness tree $T(\Lambda, t)$: each node $u$ with label $A_{[u]} \in \mathcal{A}$, siblings have distinct labels

- initially, $T$ contains a single root $r$ with $\Lambda_t$
- for $i = t - 1$ to 1:
  - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
    - add child $v \rightarrow$ deepest such $u$, labeled with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$
Moser-Tardos Algorithm:

- draw independent samples of $X_1, \ldots, X_n$;
- while $\exists$ a bad event $A_i$ that occurs:
  - resample all $X_j \in \text{vbl}(A_i)$;

Exe-log $\Lambda$:

$$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$$

random sequence of resampled bad events

Witness tree $T(\Lambda, t)$: each node $u$ with label $A_{[u]} \in \mathcal{A}$, siblings have distinct labels

- initially, $T$ contains a single root $r$ with $\Lambda_t$
- for $i = t - 1$ to $1$:
  - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
    - add child $v \rightarrow$ deepest such $u$, labeled with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$

**Proposition:** $\forall s \neq t, \ T(\Lambda, s) \neq T(\Lambda, t)$

# of $A_i$ in $\Lambda = \sum_{\tau \in \mathcal{T}_{A_i}} I \left[ \exists t, T(\Lambda, t) = \tau \right]$ \quad $\mathcal{T}_{A_i}$: set of all witness trees with root-label $A_i$

**linearity of expectation:**

$$\mathbb{E}_{\Lambda} \left[ \text{# of } A_i \text{ in } \Lambda \right] = \sum_{\tau \in \mathcal{T}_{A_i}} \mathbb{P}_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right]$$
Moser-Tardos Algorithm:
draw independent samples of \(X_1, \ldots, X_n\);
while \(\exists\) a bad event \(A_i\) that occurs:
resample all \(X_j \in \text{vbl}(A_i)\);

Exe-log \(\Lambda\):
\[\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}\]
random sequence of resampled bad events

**Witness tree** \(T(\Lambda, t)\): each node \(u\) with label \(A_{[u]} \in \mathcal{A}\), siblings have distinct labels
- initially, \(T\) contains a single root \(r\) with \(\Lambda_t\)
- for \(i = t - 1\) to 1:
  - if \(\Lambda_i \in \Gamma^+(A_{[u]})\) for some node \(u \in T\)
    - add child \(v \rightarrow\) deepest such \(u\), labeled with \(\Lambda_i\)
- \(T(\Lambda, t)\) is the resulting \(T\)

**Lemma 1** (coupling). For any particular witness tree \(\tau\):
\[
\Pr_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right] \leq \prod_{u \in \tau} \Pr(A_{[u]})
\]
**Moser-Tardos Algorithm:**

draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
    resample all $X_j \in \text{vbl}(A_i)$;

**Exe-log $\Lambda$:**

$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
random sequence of resampled bad events

---

**Lemma 1 (coupling).** For any particular witness tree $\tau$:

$$
\Pr_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right] \leq \prod_{u \in \tau} \Pr(A_{[u]})
$$

$$
\mathbb{E}_{\Lambda} \left[ \text{# of } A_i \text{ in } \Lambda \right] = \sum_{\tau \in \mathcal{T}_{A_i}} \Pr_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right]
$$

(Lemma 1) $\leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \Pr(A_{[u]})$

$\mathcal{T}_{A_i}$: set of all witness trees with root-label $A_i$
Moser-Tardos Algorithm:

draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
    resample all $X_j \in vbl(A_i)$;

Exe-log $\Lambda$:

$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
random sequence of resampled bad events

LLL condition: $\exists \alpha_1, \ldots, \alpha_m \in [0,1): \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$

$\mathbb{E} \left[ \text{# of } A_i \text{ in } \Lambda \right] = \sum_{\tau \in \mathcal{T}_{A_i}} \Pr \left[ \exists t, T(\Lambda, t) = \tau \right]$ \hspace{1cm} $\mathcal{T}_{A_i}$: set of all witness trees with root-label $A_i$

(Lemma 1) $\quad \leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \Pr(A_{[u]})$

(LLL condition) $\quad \leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right]$

Convergence: $\quad \leq \frac{\alpha_i}{1 - \alpha_i}$
Moser-Tardos Algorithm:
- draw independent samples of $X_1, \ldots, X_n$;
- while $\exists$ a bad event $A_i$ that occurs:
  - resample all $X_j \in \text{vbl}(A_i)$;

Exe-log $\Lambda$:
- $\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
- random sequence of resampled bad events

Lemma 1 (coupling). For any particular witness tree $\tau$:
- $\Pr_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right] \leq \prod_{u \in \tau} \Pr(A_{[u]})$

For some coupling: $\exists t, T(\Lambda, t) = \tau \implies$ simulation of $\tau$ succeeds

Simulation of witness tree $\tau$:
- Each node $u \in \tau$ independently does an experiment of its label-event $A_{[u]}$.
- The process succeeds if all $A_{[u]}$ occurs.

Pr[ simulation of $\tau$ succeeds ]
- $= \Pr[D] \cdot \Pr[C] \cdot \Pr[B] \cdot \Pr[E] \cdot \Pr[A] \cdot \Pr[A]$
Moser-Tardos Algorithm:

- draw independent samples of $X_1, \ldots, X_n$;
- while $\exists$ a bad event $A_i$ that occurs:
  - resample all $X_j \in \text{vbl}(A_i)$;

Exe-log $\Lambda$:

$$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$$

random sequence of resampled bad events

**Lemma 1 (coupling).** For any particular witness tree $\tau$:

$$\Pr_{\Lambda} \left[ \exists t, T(\Lambda, t) = \tau \right] \leq \prod_{u \in \tau} \Pr(A_{[u]})$$

For some coupling: $\exists t, T(\Lambda, t) = \tau \implies$ simulation of $\tau$ succeeds

**Resampling table:**

- $X_1$: $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \ldots$
- $X_2$: $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \ldots$
- $X_3$: $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \ldots$
- $X_4$: $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \ldots$
- $X_j^{(t)}$: $t$-th sampling of variable $X_j$

**exe-log $\Lambda$:** $D, C, E, D, B, A, C, A, D, \ldots$
Moser-Tardos Algorithm:
draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
    resample all $X_j \in \text{vbl}(A_i)$;

Exe-log $\Lambda$:  
$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
random sequence of resampled bad events

LLL condition:  
$\exists \alpha_1, \ldots, \alpha_m \in [0,1): \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$

$$
\mathbb{E} \left[\text{# of } A_i \text{ in } \Lambda\right] = \sum_{\tau \in \mathcal{T}_{A_i} \Lambda} \Pr \left[ \exists t, T(\Lambda, t) = \tau \right] \\
\mathcal{T}_{A_i} : \text{ set of all witness trees with root-label } A_i
$$

(Lemma 1)  
$\leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \Pr(A[u])$

(LLL condition)  
$\leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A[u])} (1 - \alpha_j) \right]$

Convergence:  
$\leq \frac{\alpha_i}{1 - \alpha_i}$
Random Tree (Galton-Watson process)

- Grow a random witness tree $T_A$ with root-label $A$
  
  - initially, $T_A$ is a single root with label $A$
  - for $i = 1, 2, \ldots$:
    - for every vertex $u$ at depth $i$ (root has depth 1) in $T_A$
    - for every $A_j \in \Gamma^+(A_{[u]})$:
      - add a new child $v$ to $u$ ind. with prob. $\alpha_j$ and label it with $A_j$;
    - stop if no new child added for an entire level
  
- Can generate all possible witness trees $\in \mathcal{T}_A$
Random Tree (Galton-Watson process)

- Grow a random witness tree $T_A$ with root-label $A$
  - initially, $T_A$ is a single root with label $A$
  - for $i = 1, 2, \ldots$:
    - for every vertex $u$ at depth $i$ (root has depth 1) in $T_A$
    - for every $A_j \in \Gamma^+(A_{[u]})$:
      - add a new child $v$ to $u$ ind. with prob. $\alpha_j$ and label it with $A_j$;
    - stop if no new child added for an entire level

Lemma 2. For any particular witness tree $\tau \in \mathcal{T}_{A_i}$:

$$
\Pr \left[ T_{A_i} = \tau \right] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right]
$$
Lemma 2. For any particular witness tree \( \tau \in \mathcal{T}_{A_i} \):

\[
\Pr \left[ T_{A_i} = \tau \right] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right]
\]

\[
\Pr \left[ T_{A_i} = \tau \right] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[ \frac{\alpha_{[u]}}{1 - \alpha_{[u]}} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right]
\]

\[
= \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[ \alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right]
\]
Moser-Tardos Algorithm:

draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
resample all $X_j \in \text{vbl}(A_i)$;

Exe-log $\Lambda$:

$\Lambda_1, \Lambda_2, \ldots \in \mathcal{A} = \{A_1, \ldots, A_m\}$
	random sequence of resampled bad events

LLL condition:

$\exists \alpha_1, \ldots, \alpha_m \in [0,1) : \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j) \leq \frac{\alpha_i}{1 - \alpha_i}$
The Moser-Tardos Algorithm

Variable framework for LLL (CSP with independent variables):

- mutually independent random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$
- bad events $\mathcal{A} = \{A_1, \ldots, A_m\}$, where $A_i \in \mathcal{A}$ is determined by $\text{vbl}(A_i) \subseteq \mathcal{X}$

Moser-Tardos Algorithm:

draw independent samples of $X_1, \ldots, X_n$;
while $\exists$ a bad event $A_i$ that occurs:
resample all $X_j \in \text{vbl}(A_i)$;

Theorem [Moser-Tardos 2010]:

$\exists \alpha_1, \ldots, \alpha_m \in [0,1)$:
$\forall i, \quad \Pr[A_i] \leq \alpha_i \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$

The Moser-Tardos algorithm terminates within $\sum_{i=1}^m \frac{\alpha_i}{1 - \alpha_i}$ resamples in expectation.
Algorithmic
Lovász Local Lemma:
Moser's Algorithm and Entropic Proof
**k-SAT**

- **k-CNF:** \( \Phi = C_1 \land C_2 \land \cdots \land C_m \)
  \[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \]

- **dependency degree** \( d \): each clause \( C_i \) intersects \( \leq d \) other clauses

- **uniform independent**
  \( X_1, \ldots, X_n \in \{T, F\} \)

- **bad event** \( A_i \):
  \[ p = 2^{-k} \]
  \( C_i \) is violated

**Moser-Tardos Algorithm:**

- draw uniform independent \( X_1, \ldots, X_n \in \{T, F\} \);
- while \( \exists \) a violated clause \( C_i \):
  - resample all \( X_j \in \text{vbl}(C_i) \);

**Theorem [Moser-Tardos 2010]:**

\[ d \leq 2^{k-2} \Leftrightarrow 4pd \leq 1 \]

The Moser-Tardos algorithm terminates after \( O(m/d) \) iterations in expectation.
Moser’s Fix-It Algorithm

- **k-CNF**: \( \Phi = C_1 \land C_2 \land \cdots \land C_m \)

\[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \]

**Moser’s Algorithm:**
draw uniform \( X_1, \ldots, X_n \in \{T, F\} \);
while \( \exists \) a violated clause \( C_i \):
\[
\text{Fix}(C_i);
\]

**Fix}(C_i):**
resample all variables in \( \text{vbl}(C_i) \);
while \( \exists \) violated \( C_j \in \Gamma^+(C_i) \):
\[
\text{Fix}(C_j);
\]

- **Inclusive neighborhood:**
\[
\Gamma^+(C_i) \triangleq \Gamma(C_i) \cup \{C_i\} = \left\{ C_j \mid \text{vbl}(C_j) \cap \text{vbl}(C_i) \neq \emptyset \right\}
\]

- **Correctness**: any clause is fixed at most once in top-level
  top-level \( \text{Fix}(C_i) \) returned \( \iff \) \( C_i \) remains satisfied
Moser’s *Fix-It* Algorithm

- **k-CNF:** \( \Phi = C_1 \land C_2 \land \cdots \land C_m \)

\[
\Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5)
\]

**Moser’s Algorithm:**

draw uniform \( X_1, \ldots, X_n \in \{ \text{T}, \text{F} \} \);
while \( \exists \) a violated clause \( C_i \):
\[
\text{Fix}(C_i);
\]

**Fix(\(C_i\)):**
resample all variables in \( \text{vbl}(C_i) \);
while \( \exists \) violated \( C_j \in \Gamma^+(C_i) \):
\[
\text{Fix}(C_j);
\]

**Theorem** [Moser 2009]:

\[
d < 2^{k-3} \implies \text{total \# of calls to Fix()} \text{ is } O(m \log m + \log n)
\]

with high probability

with probability \( 1-o(1/n) \)
Incompressibility Principle

“Lossless compression of random data is impossible.”

Incompressibility Principle:

For any injective function $\text{Enc} : \{0,1\}^N \rightarrow \{0,1\}^*$, for uniform random $s \in \{0,1\}^N$, for any integer $l > 0$,

$$\Pr \left[ \text{length of } \text{Enc}(s) \leq N - l \right] < 2^{1-l}$$

$$\left| \{0,1\}^N \right| = 2^N$$

# of strings of $\leq N - l$ bits

$$= \sum_{i \leq N-l} 2^i < 2^{N-l+1}$$

Andrey Kolmogorov (1903–1987)
Entropic Proof

• $k$-CNF $\Phi = C_1 \land C_2 \land \cdots \land C_m$ with dependency degree $d < 2^{k-3}$

Moser’s Algorithm:
- draw uniform $X_1, \ldots, X_n \in \{T, F\}$;
- while $\exists$ a violated clause $C_i$:
  - $\text{Fix}(C_i)$;

Fix($C_i$):
- resample all variables in $\text{vbl}(C_i)$;
- while $\exists$ violated $C_j \in \Gamma^+(C_i)$:
  - $\text{Fix}(C_j)$;

• Random bits: $n$ initial bits + $t$ calls to $\text{Fix}() \times k$ bits per each call

• $t$ calls to $\text{Fix}() \implies$ we can compress random bits:

\[ n + tk \text{ random bits} \xrightarrow{\text{Enc}_\Phi} \leq n + O(m \log m) + t(\lceil \log_2(d + 1) \rceil + 2) \text{ bits} \]
Simulate($\Phi, t$):
Run Moser's Algorithm on $k$-CNF $\Phi$ for up to $t$ calls to Fix();
printf("$X_1 \ldots X_n$"); //the current $X_1, \ldots, X_n$ in Moser's algorithm

Moser's Algorithm:
draw uniform $X_1, \ldots, X_n \in \{T, F\}$;
while $\exists$ a violated clause $C_i$:
  printf("(i"));
  Fix($C_i$);

Fix($C_i$):
resample all variables in vbl($C_i$);
while $\exists$ violated $C_j \in \Gamma^+(C_i)$:
  let $r = \text{rank}$ of $C_j$ in $\Gamma^+(C_i)$;
  printf("(r") and Fix($C_j$);
  printf(")");

- Output string: 
  $$(98(2(1(1)(3))(3))(4(2))) (126(3(2\ldots\ldots(110100101010110111 \ldots))))$$
  \(t\) calls to Fix()

- Recursion trees + final assignment

\[ |\Gamma^+(C_i)| \leq d + 1 \]
Moser’s Algorithm:

draw uniform $X_1, \ldots, X_n \in \{\top, \bot\}$;
while $\exists$ a violated clause $C_i$ :
$$\text{Fix}(C_i);$$

Fix($C_i$):
resample all variables in $\text{vbl}(C_i)$;
while $\exists$ violated $C_j \in \Gamma^+(C_i)$ :
$$\text{Fix}(C_j);$$

**Observation:** $\text{Fix}(C_i)$ is called $\implies C_i$ is violated at the moment $\implies$ current values of all $X_j \in \text{vbl}(C_i)$ is uniquely determined

- **Output string:**
  $$(98(2(1(1)(3))(3))(4(2)))(126(3(2 \cdots \cdots)(110100101010110111
  t \text{ calls to } \text{Fix}()$$
  $$X_1, \ldots, X_n$$

- **Recursion trees + final assignment**

- **1-1 mapping**
  $n + tk$ random bits used by the algorithm
Simulate($\Phi$, $t$):
Run Moser’s Algorithm on $k$-CNF $\Phi$ for up to $t$ calls to Fix();
printf(“$X_1 \ldots X_n$”);

Moser’s Algorithm:
draw uniform $X_1, \ldots, X_n \in \{T, F\}$;
while $\exists$ a violated clause $C_i$ :
    printf(“(i”);
    Fix($C_i$);

Fix($C_i$):
resample all variables in $\text{vbl}(C_i)$;
while $\exists$ violated $C_j \in \Gamma^+(C_i)$ :
    let $r = \text{rank}$ of $C_j$ in $\Gamma^+(C_i)$;
    printf(“(r”) and Fix($C_j$);
    printf(“)”)

• Output string:
  • $\leq m \times \text{“(i”, where $i \in [m]$ at top-level}$
  • $\leq t \times \text{“(r”, where $r \in [d + 1]$}$
  • $\leq t \times \text{“)”}$
  • “(“)” form prefix of legal parenthesization
  • $n$ bits $X_1, \ldots, X_n$ in the end

\[
\leq n + m\left(\lceil \log_2 m \rceil + 2\right) + t\left(\lceil \log_2 (d + 1) \rceil + 2\right) \text{ bits}
\]
Incompressibility Principle:
For any injective function $\text{Enc} : \{0,1\}^N \overset{1-1}{\longrightarrow} \{0,1\}^*$, for uniform random $s \in \{0,1\}^N$, for any integer $l > 0$,

$$\Pr \left[ \text{length of Enc}(s) \leq N - l \right] < 2^{1-l}$$

- Assume $\geq t$ calls made to $\text{Fix}()$:

  $n + tk$ random bits used by the algorithm

  \[ \text{Enc}_\Phi \leq n + m([\log_2 m] + 2) + t([\log_2(d + 1)] + 2) \text{ bits} \]

  (LLL condition $d < 2^{k-3}$)

- For any $t \geq m(\log_2 m + 3) + \log_2 n + 1$:

  this can only occur with probability $< \frac{1}{n}$
Moser’s *Fix-It* Algorithm

- **k-CNF:** \( \Phi = C_1 \land C_2 \land \cdots \land C_m \)
  \[ \Phi = (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_4) \land (x_3 \lor \neg x_4 \lor \neg x_5) \]

**Moser’s Algorithm:**
- Draw uniform \( X_1, \ldots, X_n \in \{T, F\} \);
- While there exists a violated clause \( C_i \):
  - \( \text{Fix}(C_i) \);

**Fix(\( C_i \)):**
- Resample all variables in \( \text{vbl}(C_i) \);
- While there exists a violated \( C_j \in \Gamma^+(C_i) \):
  - \( \text{Fix}(C_j) \);

**Theorem [Moser 2009]:**
\[ d < 2^{k-3} \implies \text{total \# of calls to Fix() is } O(m \log m + \log n) \]
with high probability

with probability \( 1-O(1/n) \)
Lovász Local Lemma: The Probabilistic Method

Algorithmic
Lovász Local Lemma:
Moser-Tardos Algorithm
Moser's Algorithm and Entropic Proof
Summary

• The Moser-Tardos algorithm:

LLL condition

Efficiently find $X_1, \ldots, X_n$ that avoids all $A_1, \ldots, A_m$

$A_1, \ldots, A_m$ are defined on independent random variables $X_1, \ldots, X_n$

• The proof based on resampling table and witness trees

• What’s next:

  • tighter LLL condition for algorithms
  • beyond independent variables: non-variable framework
  • more than construction: sampling LLL