Advanced Algorithms

南京大学

尹一通
\textbf{\textit{k-SAT}}

- \textit{n Boolean variables: } $x_1, x_2, \ldots, x_n \in \{\text{true, false}\}$
- \textit{conjunctive normal form:}

  \[ \textit{k-CNF} \quad \phi = C_1 \land C_2 \land \cdots \land C_m \]

  \text{“Is } \phi \text{ satisfiable?”}

- \textit{m clauses: } $C_1, C_2, \ldots, C_m$
- \textit{each clause } $C_i = \ell_{i1} \lor \ell_{i2} \lor \cdots \lor \ell_{ik}$
  \text{is a disjunction of exact } k \text{ literals}
- \textit{each literal: } $\ell_j \in \{x_r, \neg x_r\}$ \text{ for some } r
- \textit{degree } d : \text{ each clause shares variables with at most } d \text{ other clauses}
Theorem

\[ d \leq 2^{k-2} \quad \Rightarrow \quad \phi \text{ is always satisfiable} \]

uniform random assignment \( X_1, X_2, \ldots, X_n \)

for clause \( C_i \), bad event \( A_i : C_i \) is not satisfied

\[ d \leq 2^{k-2} \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] > 0 \]
Lovász Sieve

- **Bad events:** \( A_1, A_2, \ldots, A_n \)

- None of the bad events occurs:
  \[
  \Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right]
  
  > 0
  
- The probabilistic method: being good is possible
  \[
  \Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] > 0
  
  
\]
events: $A_1, A_2, \ldots, A_n$

dependency graph: $D(V,E)$

$$V = \{1, 2, \ldots, n\}$$

$ij \in E \iff A_i$ and $A_j$ are dependent

$d: \text{max degree of dependency graph}$

$X_1, \ldots, X_4$ mutually independent
events: \( A_1, A_2, \ldots, A_n \)

\( d \): max degree of dependency graph

**Lovász Local Lemma**

- \( \forall i, \ Pr[A_i] \leq p \)
- \( ep(d + 1) \leq 1 \)

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

**General Lovász Local Lemma**

\[ \exists x_1, \ldots, x_n \in [0, 1) \]
\[ \forall i, \ Pr[A_i] \leq x_i \prod_{j \sim i} (1 - x_j) \]

\[ \Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] \geq \prod_{i=1}^{n} (1 - x_i) \]
LLL for $k$-SAT

$\phi : k$-CNF of max degree $d$

**Theorem**

\[ d \leq 2^{k-2} \quad \Rightarrow \quad \exists \text{satisfying assignment for } \phi \]

uniform random assignment $X_1, X_2, \ldots, X_n$

for clause $C_i$, **bad event** $A_i : C_i$ is **not** satisfied

LLL:

\[ e(d + 1) \leq 2^k \quad \Rightarrow \quad \Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] > 0 \]
Algorithmic $LLL$

$\phi$: $k$-CNF of max degree $d$ with $m$ clauses on $n$ variables

\textbf{Theorem}

\[ d \leq 2^{k-2} \quad \exists \text{satisfying assignment for } \phi \]

\textbf{Theorem} (Moser, 2009)

\[ d < 2^{k-3} \quad \text{satisfying assignment can be found in } O(n + km \log m) \text{ w.h.p.} \]
\( \phi : k\text{-CNF of max degree } d \text{ with } m \text{ clauses on } n \text{ variables} \)

**Solve(\( \phi \))**
- pick a random assignment \( x_1, x_2, \ldots, x_n \); 
- while \( \exists \) unsatisfied clause \( C \)
  - **Fix(\( C \));**

**Fix(\( C \))**
- replace variables in \( C \) with random values; 
- while \( \exists \) unsatisfied clause \( D \) overlapping with \( C \)
  - **Fix(\( D \));**
\( \phi : k\text{-CNF of max degree } d \text{ with } m \text{ clauses on } n \text{ variables} \)

\begin{align*}
\text{Solve}(\phi) & \quad \text{Pick a random assignment } x_1, x_2, \ldots, x_n; \\
& \quad \text{while } \exists \text{ unsatisfied clause } C \\
& \quad \quad \text{Fix}(C); \\
\text{Fix}(C) & \quad \text{replace variables in } C \text{ with random values;} \\
& \quad \text{while } \exists \text{ unsatisfied clause } D \text{ overlapping with } C \\
& \quad \quad \text{Fix}(D); \\
\end{align*}

at top-level:

**Observation**: A clause \( C \) is satisfied and will keep satisfied once it has been fixed.

\# of top-level calls to \( \text{Fix}(C) \) : \( \leq m \) (# of clauses)

total \# of calls to \( \text{Fix}(C) \) (including recursive calls) : \( t \)
$\phi : k$-CNF of max degree $d$ with $m$ clauses on $n$ variables

**Solve($\phi$)**
- Pick a random assignment $x_1, x_2, \ldots, x_n$;
- while $\exists$ unsatisfied clause $C$
  - Fix($C$);

**Fix($C$)**
- replace variables in $C$ with random values;
- while $\exists$ unsatisfied clause $D$ overlapping with $C$
  - Fix($D$);

$\leq m$ recursion trees  \hspace{2cm} \text{total \# nodes: \hspace{0.5cm} } t$

**Observation:** Fix($C$) is called assignment of $C$ is uniquely determined
$\leq m$ recursion trees    total # nodes:  $t$

the sequence of random bits can be recovered from:

final assignment:    $n$ bits

+ recursion trees:    $\leq m[\log_2 m] + t(\log_2 d + O(1))$ bits

for each recursion tree:

root:    $[\log_2 m]$ bits

each internal node:    $\leq \log_2 d + O(1)$ bits

total # of random bits:    $n+tk$    (assigned bits)
\[ \leq m \text{ recursion trees} \quad \text{total \# nodes:} \quad t \]

The sequence of random bits is encoded to:

\[ \leq n + m \lfloor \log_2 m \rfloor + t(\log_2 d + 3) \text{ bits} \]

**Incompressibility Theorem** (Kolmogorov)

\( N \) uniform random bits cannot be encoded to substantially less than \( N \) bits.
total # of random bits: \( n + tk \) (assigned bits)

the sequence of random bits is \textit{encoded to}:

\[
\leq n + m \lceil \log_2 m \rceil + t(\log_2 d + 3) \text{ bits}
\]

\textbf{Incompressibility Theorem} (Kolmogorov)

\( N \) uniform random bits cannot be encoded to less than \( N - l \) bits with probability \( 1 - O(2^{-l}) \).
\[ \leq m \text{ recursion trees} \quad \text{total # nodes: } t \]

\[ \text{total # of random bits: } n + tk \quad \text{(assigned bits)} \]

the sequence of random bits is \textit{encoded to}:

\[ \leq n + m \lfloor \log_2 m \rfloor + t (\log_2 d + c) \quad \text{bits} \]

\[ t (k - c - \log_2 d) \leq m \lfloor \log_2 m \rfloor + \log n \quad \text{whp} \]

when \[ d < 2^{k-c} \]

\[ t \leq \frac{m \lfloor \log_2 m \rfloor + \log n}{k - c - \log_2 d} \]

\text{total running time: } n + tk = O(n + km \log m)
Algorithmic LLL

$\phi : k$-CNF of max degree $d$ with $m$ clauses on $n$ variables

$\phi = C_1 \land C_2 \land \cdots \land C_m$

**Theorem** (Moser, 2009)

$d < 2^{k-c}$ satisfying assignment can be found in $O(n + km \log m)$ whp

**Solve($\phi$)**

Pick a random assignment $x_1, x_2, \ldots, x_n$;

while $\exists$ unsatisfied clause $C$

**Fix($C$)**

replace variables in $C$ with random values;

while $\exists$ unsatisfied clause $D$ overlapping with $C$

**Fix($D$)**;
events: $A_1, A_2, \ldots, A_n$

$d$ : max degree of dependency graph

**Lovász Local Lemma**

- $\forall i, \Pr[A_i] \leq p$
- $ep(d + 1) \leq 1$

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

**General Lovász Local Lemma**

$\exists x_1, \ldots, x_n \in [0, 1)$

$\forall i, \Pr[A_i] \leq x_i \prod_{j \sim i} (1 - x_j)$

\[
\Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] \geq \prod_{i=1}^{n} (1 - x_i)
\]
Constraint Satisfaction Problem

- **variables**: $x_1, x_2, \ldots, x_n \in D$ (domain)
- **constraints**: $C_1, C_2, \ldots, C_m$
  - where $C_i(x_{i1}, x_{i2}, \ldots) \in \{\text{true}, \text{false}\}$
- **CSP solution**: an assignment of variables satisfying *all* constraints
- **examples**: SAT, graph colorability, ...
- **existence**: When does a solution exist?
- **search**: How to find a solution?
The Probabilistic Method

CSP $C_1, C_2, \ldots, C_m$ defined on $x_1, x_2, \ldots, x_n$

• sampling random values of $x_1, x_2, \ldots, x_n$

• **Bad event** $A_i$: constraint $C_i$ is **violated**

• None of the bad events occurs with prob: $\Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right]$

• **The probabilistic method**: being **good** is possible

$$\Pr \left[ \bigwedge_{i=1}^{m} \overline{A_i} \right] > 0$$
events: $A_1, A_2, \ldots, A_n$

d : max degree of dependency graph

**Lovász Local Lemma**

- $\forall i$, $\Pr[A_i] \leq p$
- $ep(d + 1) \leq 1$

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

**General Lovász Local Lemma**

$\exists x_1, \ldots, x_n \in [0, 1)$

$\forall i$, $\Pr[A_i] \leq x_i \prod_{j \sim i} (1 - x_j)$

\[
\Pr \left[ \bigwedge_{i=1}^{n} \overline{A_i} \right] \geq \prod_{i=1}^{n} (1 - x_i)
\]
mutually independent random variables: $X \in \mathcal{X}$

bad events: $A \in \mathcal{A}$ defined on variables in $\mathcal{X}$

$vbl(A) \subseteq \mathcal{X}$: set of variables on which $A$ is defined

neighborhood: $\Gamma(A) = \{ B \in \mathcal{A} \mid B \neq A \text{ and } vbl(A) \cap vbl(B) \neq \emptyset \}$

inclusive neighborhood: $\Gamma^+(A) = \Gamma(A) \cup \{ A \}$

“events that are dependent with $A$, excluding/including $A$ itself”

Lovász Local Lemma (general)

\[
\exists \alpha : \mathcal{A} \rightarrow [0, 1) \\
\forall A \in \mathcal{A} : \\
\Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B) \\
\Pr \left[ \bigwedge_{A \in \mathcal{A}} \overline{A} \right] \geq \prod_{A \in \mathcal{A}} (1 - \alpha_A) \\
> 0
\]
**mutually independent** random variables: \( X \in \mathcal{X} \\
**bad events**: \( A \in \mathcal{A} \) defined on variables in \( \mathcal{X} \\
**vbl(A) \subseteq \mathcal{X}**: set of variables on which \( A \) is defined

**neighborhood**: \( \Gamma(A) = \{B \in \mathcal{A} \mid B \neq A \text{ and } \text{vbl}(A) \cap \text{vbl}(B) \neq \emptyset\} \)

**inclusive neighborhood**: \( \Gamma^+(A) = \Gamma(A) \cup \{A\} \)

“events that are dependent with \( A \), excluding/including \( A \) itself”

---

**Lovász Local Lemma (general)**

\[
\exists \alpha : \mathcal{A} \rightarrow [0, 1) \\
\forall A \in \mathcal{A} : \\
\Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B)
\]

\( \exists \) values of variables in \( \mathcal{X} \) avoiding all bad events \( A \in \mathcal{A} \) simultaneously.
Algorithmic LLL

bad events $A \in \mathcal{A}$ defined on mutually independent random variables $X \in \mathcal{X}$

$vbl(A)$: set of variables on which $A$ is defined

neighborhood $\Gamma(A)$ and inclusive neighborhood $\Gamma^+(A)$

Assumption:
I. We can efficiently sample an independent evaluation of every random variable $X \in \mathcal{X}$.
II. We can efficiently check the violation of every event $A \in \mathcal{A}$.

RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated bad event $A \in \mathcal{A}$:
resample all $X \in vbl(A)$;
bad events $A \in \mathcal{A}$ defined on
mutually independent random variables $X \in \mathcal{X}$

$vbl(A)$: set of variables on which $A$ is defined

neighborhood $\Gamma(A)$ and inclusive neighborhood $\Gamma^+(A)$

RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated bad event $A \in \mathcal{A}$:
resample all $X \in vbl(A)$;

Moser-Tardos 2010:

$\exists \alpha : A \to [0, 1)$
$\forall A \in \mathcal{A} :$
$\Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B)$

RandomSolver finds values of all $X \in \mathcal{X}$ avoiding all $A \in \mathcal{A}$
within expected
\[
\sum_{A \in \mathcal{A}} \frac{\alpha_A}{1 - \alpha_A}
\]
bad events $A \in \mathcal{A}$ defined on mutually independent random variables $X \in \mathcal{X}$

$vbl(A)$: set of variables on which $A$ is defined

neighborhood $\Gamma(A)$ and inclusive neighborhood $\Gamma^+(A)$

RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated bad event $A \in \mathcal{A}$:
resample all $X \in vbl(A)$;

Moser-Tardos 2010:

- $\forall A \in \mathcal{A}$, $\Pr[A] \leq p$
- $ep(d + 1) \leq 1$
  where $d = \max_A |\Gamma(A)|$

RandomSolver finds values of all $X \in \mathcal{X}$ violating all $A \in \mathcal{A}$ within expected $|\mathcal{A}| / d$ resamples.
\(k\text{-SAT}\)

\(\phi: k\text{-CNF of max degree } d \text{ with } m \text{ clauses on } n \text{ variables}

\text{RandomSolver:}

pick a random assignment \(x_1, x_2, \ldots, x_n\);
while \(\exists\) an unsatisfied clause \(C\):
replace variables in \(C\) with random values;

\[d \leq 2^{k-2} \quad (e(d+1) \leq 2^k)\]

\text{RandomSolver returns a satisfying assignment within expected } O(n + km/d) \text{ time}
**bad events** $A \in \mathcal{A}$ defined on **mutually independent** random variables $X \in \mathcal{X}$

$vbl(A)$: set of variables on which $A$ is defined

neighborhood $\Gamma(A)$ and **inclusive** neighborhood $\Gamma^+(A)$

---

**RandomSolver:**

sample all $X \in \mathcal{X}$;

while $\exists$ a non-violated bad event $A \in \mathcal{A}$:

resample all $X \in vbl(A)$;

---

**Moser-Tardos 2010:**

$$\exists \alpha : \mathcal{A} \rightarrow [0, 1) \quad \forall A \in \mathcal{A} :$$

$$\Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B)$$

**RandomSolver** finds values of all $X \in \mathcal{X}$ avoiding all $A \in \mathcal{A}$ within expected resamples.

$$\sum_{A \in \mathcal{A}} \frac{\alpha_A}{1 - \alpha_A}$$
RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated $A \in \mathcal{A}$:
    resample all $X \in \text{vbl}(A)$;

execution log $\Lambda$:

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$

random sequence of resampled events

$N_A = |\{ i \mid \Lambda_i = A \}|$

total # of times that $A$ is resampled

\textbf{Moser-Tardos 2010:}

$\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

$\forall A \in \mathcal{A} :$

$\Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B)$

$\forall A \in \mathcal{A} :$

$\mathbb{E}[N_A] \leq \frac{\alpha_A}{1 - \alpha_A}$
RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated $A \in \mathcal{A}$:
    resample all $X \in \text{vbl}(A)$;

execution log $\Lambda$:

$$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$$

random sequence of resampled events

**witness tree**: A witness tree $\tau$ is a labeled tree in which every vertex $v$ is labeled by an event $A_v \in \mathcal{A}$, such that siblings have distinct labels.

$T(\Lambda, t)$ is a **witness tree** constructed from exe-log $\Lambda$:

- initially, $T$ is a single root with label $\Lambda_t$
- for $i = t-1, t-2, \ldots, 1$
  - if $\exists$ a vertex $v$ in $T$ with label $A_v \in \Gamma^+(\Lambda_i)$
  - add a new child $u$ to the deepest such $v$ and label it with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$
$T(\Lambda, t)$ is a witness tree constructed from exe-log $\Lambda$:

- initially, $T$ is a single root with label $\Lambda_t$
- for $i = t-1, t-2, \ldots, 1$
  - if $\exists$ a vertex $v$ in $T$ with label $A_v \in \Gamma^+(\Lambda_i)$
  - add a new child $u$ to the deepest such $v$ and label it with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$
dependency graph:

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

$T(\Lambda, 8)$:

$T(\Lambda, 9)$:

$T(\Lambda, t)$ is a witness tree constructed from exe-log $\Lambda$:

- initially, $T$ is a single root with label $\Lambda_t$
- for $i = t-1, t-2, ..., 1$
  - if $\exists$ a vertex $v$ in $T$ with label $A_v \in \Gamma^+(\Lambda_i)$
  - add a new child $u$ to the deepest such $v$ and label it with $\Lambda_i$
- $T(\Lambda, t)$ is the resulting $T$
RandomSolver:
sample all \( X \in \mathcal{X} \);
while \( \exists \) a non-violated \( A \in \mathcal{A} \):
resample all \( X \in \text{vbl}(A) \);

execution log \( \Lambda \):
\( \Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A} \)
random sequence of resampled events

witness tree: A witness tree \( \tau \) is a labeled tree in which every vertex \( v \) is labeled by an event \( A_v \in \mathcal{A} \), such that siblings have distinct labels.

\( T(\Lambda, t) \) is a witness tree constructed from exe-log \( \Lambda \):

- initially, \( T \) is a single root with label \( \Lambda_t \)
- for \( i = t-1, t-2, \ldots, 1 \)
  - if \( \exists \) a vertex \( v \) in \( T \) with label \( A_v \in \Gamma^+(\Lambda_i) \)
  - add a new child \( u \) to the deepest such \( v \) and label it with \( \Lambda_i \)
- \( T(\Lambda, t) \) is the resulting \( T \)

\( T(\Lambda, s) \neq T(\Lambda, t) \) if \( s \neq t \)

\[ \mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau] \]

\( \mathcal{T}_A \): set of all witness trees with root-label \( A \)
RandomSolver:
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated $A \in \mathcal{A}$:
    resample all $X \in \text{vbl}(A)$;

execution log $\Lambda$:

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$

random sequence of resampled events

$T(\Lambda, t)$ is a witness tree constructed from exe-log $\Lambda$:

- initially, $T$ is a single root with label $\Lambda_t$
- for $i = t-1, t-2, \ldots, 1$
  - if $\exists$ a vertex $v$ in $T$ with label $A_v \in \Gamma^+(\Lambda_i)$
    - add a new child $u$ to the deepest such $v$ and label it with $\Lambda_i$
  - $T(\Lambda, t)$ is the resulting $T$

Lemma 1
For any particular witness tree $\tau$:

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$
**RandomSolver:**
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated $A \in \mathcal{A}$:
resample all $X \in \text{vbl}(A)$;

**execution log** $\Lambda$:

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$

random sequence of resampled events

---

**Lemma 1**

For any particular witness tree $\tau$:

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

$N_A = \left| \{ i \mid \Lambda_i = A \} \right|$ total # of times that $A$ is resampled

$$\mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$\mathcal{T}_A$: set of all witness trees with root-label $A$

$$(\text{lemma 1}) \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$$
RandomSolver:

- sample all $X \in \mathcal{X}$;
- while $\exists$ a non-violated $A \in \mathcal{A}$:
  - resample all $X \in \text{vbl}(A)$;

LLL condition: $\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

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

$$E[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

(lemma 1) $\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$  

(LLL cond.) $\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]$

goal: $\leq \frac{\alpha_A}{1 - \alpha_A}$

execution log $\Lambda$:

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$

random sequence of resampled events

$\mathcal{T}_A$: set of all witness trees with root-label $A$
**RandomSolver:**

- sample all $X \in \mathcal{X}$;
- while $\exists$ a non-violated $A \in \mathcal{A}$:
  - resample all $X \in \text{vbl}(A)$;

**execution log $\Lambda$:**

$$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$$

random sequence of resampled events

**Lemma 1**

For any particular witness tree $\tau$:

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

**exe-log $\Lambda$:**

D,C,E,D,B,A,C,A,D, ...
**RandomSolver:**

*sample all $X \in \mathcal{X}$;*

*while $\exists$ a non-violated $A \in \mathcal{A}$:*

*resample all $X \in \text{vbl}(A)$;*

**execution log $\Lambda$:**

$$\Lambda_1, \Lambda_2, \Lambda_3, ... \in \mathcal{A}$$

random sequence of resampled events

**Lemma 1**

For any particular witness tree $\tau$:

$$\Pr[\exists t, T(\Lambda, t) = \tau] \leq \prod_{v \in \tau} \Pr[A_v]$$

$$N_A = |\{ i | \Lambda_i = A \}|$$

total # of times that $A$ is resampled

$$\mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

$\mathcal{T}_A$: set of all witness trees with root-label $A$

$(\text{lemma 1}) \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$
**RandomSolver:**
sample all $X \in \mathcal{X}$;
while $\exists$ a non-violated $A \in \mathcal{A}$:
  resample all $X \in \text{vbl}(A)$;

**LLL condition:**
$\exists \alpha : \mathcal{A} \rightarrow [0, 1)$
$\forall A \in \mathcal{A} : \Pr[A] \leq \alpha_A \prod_{B \in \Gamma(A)} (1 - \alpha_B)$

$$E[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau]$$

(lemma 1) $\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]$  

(LLL cond.) $\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]$  

**goal:** $\leq \frac{\alpha_A}{1 - \alpha_A}$

**execution log $\Lambda$:**
$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$
random sequence of resampled events

**$\mathcal{T}_A$:** set of all witness trees with root-label $A$
grow a random witness tree $T_A \in \mathcal{T}_A$:

- initially, $T_A$ is a single root with label $A$
- for $i = 1, 2, ...$
  - for every vertex $v$ at depth $i$ (root has depth 1) in $T_A$
  - for every $B \in \Gamma^+(A_v)$:
    - add a new child $u$ to $v$ independently with probability $\alpha_B$;
    - and label it with $B$;
  - stop if no new child added for an entire level

**Lemma 2** For any particular witness tree $\tau \in \mathcal{T}_A$:

$$
\Pr[T_A = \tau] = \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right]
$$
**Lemma 2** For any particular witness tree $\tau \in \mathcal{T}_A$:

$$\Pr[T_A = \tau] = \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right]$$

**Diagram:**

In $\tau$:

- $\Gamma^+(A)$:
  - Blue dots
  - $\Gamma_0^+(A)$

$$\Pr[T_A = \tau] = \frac{1}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma^+_0(A_v)} (1 - \alpha_B) \right]$$

$$= \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \frac{\alpha(A_v)}{1 - \alpha(A_v)} \prod_{B \in \Gamma^+(A_v)} (1 - \alpha_B) \right]$$

$$= \frac{1 - \alpha_A}{\alpha_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha_B) \right]$$
**RandomSolver:**

sample all $X \in \mathcal{X}$;

while $\exists$ a non-violated $A \in \mathcal{A}$:

- resample all $X \in \text{vbl}(A)$;

**execution log $\Lambda$:**

$\Lambda_1, \Lambda_2, \Lambda_3, \ldots \in \mathcal{A}$

random sequence of resampled events

**LLL condition:**

$\exists \alpha : \mathcal{A} \rightarrow [0, 1)$

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

$$
\mathbb{E}[N_A] = \sum_{\tau \in \mathcal{T}_A} \Pr[\exists t, T(\Lambda, t) = \tau] \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \Pr[A_v]
$$

(lemma 1)

$$
\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in \tau} \left[ \alpha(A_v) \prod_{B \in \Gamma(A_v)} (1 - \alpha(B)) \right]
$$

(LLL cond.)

$$
\leq \frac{\alpha_A}{1 - \alpha_A} \sum_{\tau \in \mathcal{T}_A} \Pr[T_A = \tau] \leq \frac{\alpha_A}{1 - \alpha_A}
$$

(lemma 2)
**bad events** $A \in \mathcal{A}$ defined on **mutually independent** random variables $X \in \mathcal{X}$

$vbl(A)$: set of variables on which $A$ is defined

neighborhood $\Gamma(A)$ and **inclusive** neighborhood $\Gamma^+(A)$

---

**RandomSolver**:

sample all $X \in \mathcal{X}$;

while $\exists$ a non-violated bad event $A \in \mathcal{A}$:

resample all $X \in vbl(A)$;

---

**Moser-Tardos 2010**:

\[
\exists \alpha : \mathcal{A} \rightarrow [0, 1) \\
\forall A \in \mathcal{A} : \\
\quad \Pr[A] \leq \alpha(A) \prod_{B \in \Gamma(A)} (1 - \alpha(B))
\]

**RandomSolver** finds values of all $X \in \mathcal{X}$ violating all $A \in \mathcal{A}$ within expected resamples.

\[
\sum_{A \in \mathcal{A}} \frac{\alpha(A)}{1 - \alpha(A)}
\]