

Advanced Algorithms

Lovász Local Lemma

尹一通 Nanjing University, 2026 Fall

k -SAT

- Conjunctive Normal Form (**CNF**):

$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge \underbrace{(x_1 \vee x_2 \vee x_4)}_{\text{clause}} \wedge (\underbrace{x_3}_{\text{literal}} \vee \underbrace{\neg x_4}_{\text{literal}} \vee \neg x_5)$$

Boolean variables: $x_1, x_2, \dots, x_n \in \{\text{True}, \text{False}\}$

- k -CNF: each clause contains **exactly** k variables

Problem (k -SAT)

Input: k -CNF formula Φ .

Output: determine whether Φ is satisfiable.

- [Cook-Levin] **NP-hard** if $k \geq 3$

Trivial Cases

Problem (k -SAT)

Input: k -CNF formula Φ .

Output: determine whether Φ is satisfiable.

- Clauses are *disjoint*:

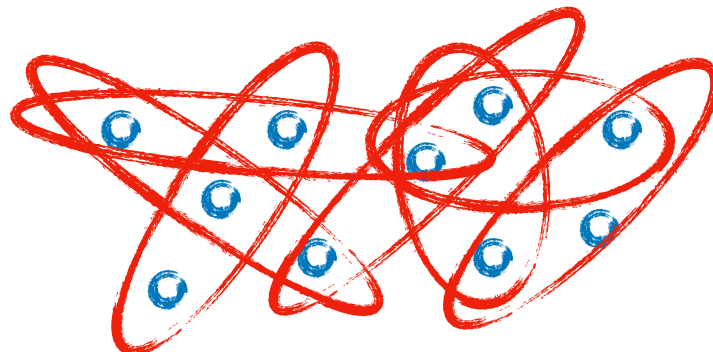
$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_7 \vee \neg x_8 \vee \neg x_9)$$



resolve each clause *independently* (Φ is always satisfiable!)

- $m < 2^k \Rightarrow \Phi$ is always satisfiable.

m : # of clauses



The Probabilistic Method

- k -CNF: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$

$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_3 \vee \neg x_4 \vee \neg x_5)$$

$\underbrace{\hspace{10em}}_{k \text{ variables}}$

- Draw uniform random $x_1, x_2, \dots, x_n \in \{\text{True}, \text{False}\}$
- **Bad event** A_i : clause C_i is violated

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

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

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

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

(independent bad events)

↗

The Probabilistic Method

Problem (k -SAT)

Input: k -CNF formula Φ .

Output: determine whether Φ is satisfiable.

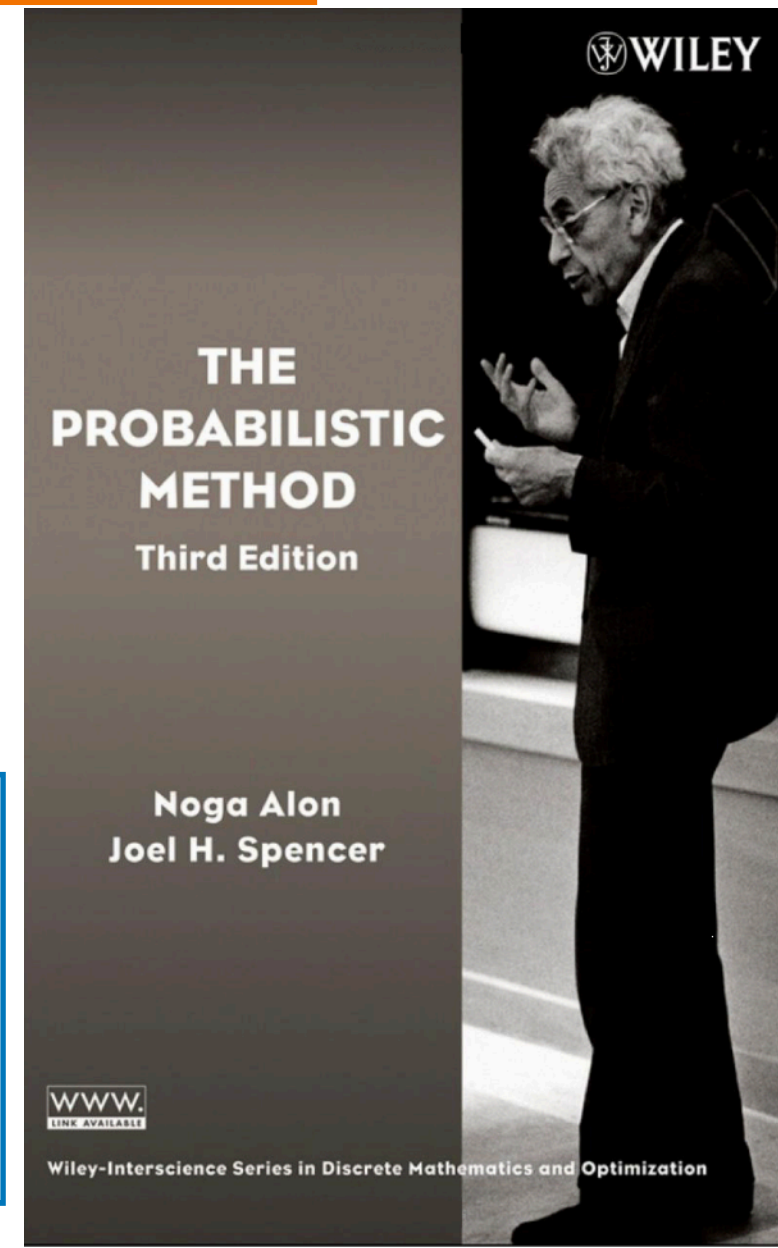
- uniform random $x_1, \dots, x_n \in \{\text{True}, \text{False}\}$

$$\left. \begin{array}{l} \text{disjoint clauses} \\ \text{or} \\ m < 2^k \end{array} \right\} \implies \Pr[\Phi(x) = \text{True}] > 0$$
$$\implies \begin{array}{l} \exists x \in \{\text{T}, \text{F}\}^n \\ \Phi(x) = \text{True} \end{array}$$

The Probabilistic Method:

Draw x from prob. space Ω : for property $\mathcal{P}$,

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



Limited Dependency

- k -CNF: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$

$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overbrace{x_3 \vee \neg x_4 \vee \neg x_5}^{k \text{ variables}})$$

Dependency degree d :

each clause intersects $\leq d$ other clauses

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

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

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

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

$4d2^{-k} \leq 1$

$$\left. \begin{array}{l} m2^{-k} < 1 \\ \text{("local" union bound?) } d2^{-k} < 1 \\ e(d+1)2^{-k} \leq 1 \\ 4d2^{-k} \leq 1 \end{array} \right\} \Rightarrow \Pr[\text{no clause is violated}] > 0$$

Lovász Local Lemma (**LLL**)

- “Bad” events $A_1, \dots, A_m$, 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]:

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

- 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 $A_1, \dots, A_m$,

Dependency degree d :

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

Definition (independence):

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

A_0 is mutually independent of $A_1, \dots, A_m$ if A_0 is independent of every event $B = B_1 \wedge \dots \wedge B_m$, where each $B_i = A_i$ or $\bar{A}_i$.

Dependency Graph

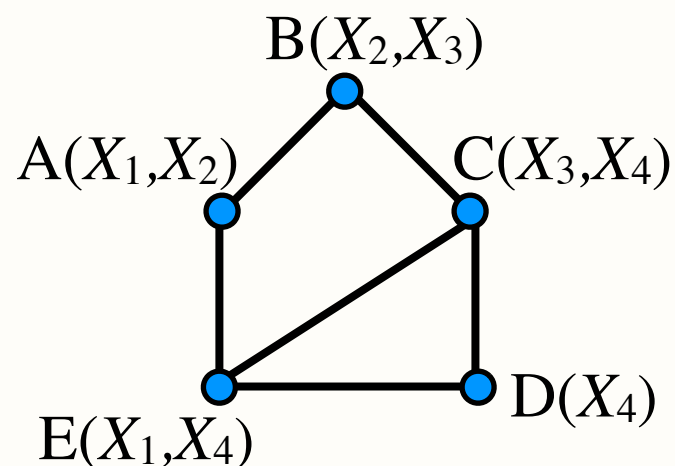
- “Bad” events $A_1, \dots, A_m$,

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

Dependency graph: $\Gamma(A_i)$: neighborhood of A_i .

Vertices are bad events $A_1, \dots, A_m$.

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)

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

A_i is mutually independent of all $A_j \notin \Gamma(A_i)$

Lovász Local Lemma:

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

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

Constraint Satisfaction Problem (**CSP**)

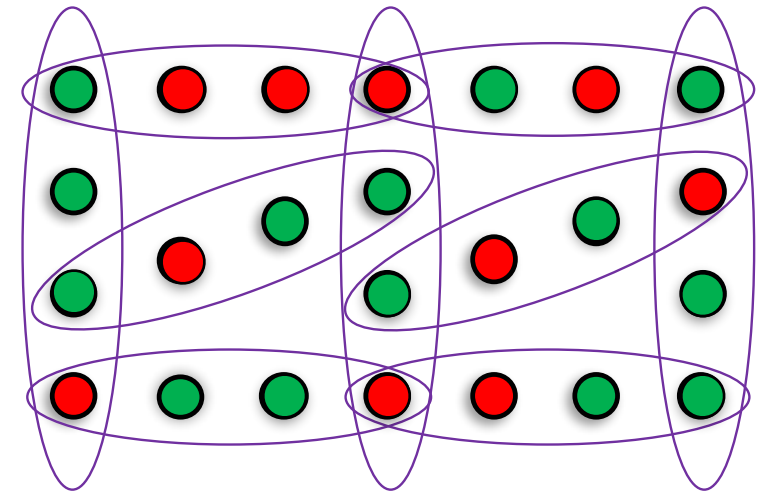
- Variables: $x_1, \dots, x_n \in [q]$
- (**local**) Constraints: $C_1, \dots, C_m$
 - each C_i is defined on a subset **vbl**(C_i) of variables

$$C_i : [q]^{\text{vbl}(C_i)} \rightarrow \{\text{True}, \text{False}\}$$

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

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

$$k \geq \log_q \Delta + \log_q \log_q \Delta + O(1)$$

Hypergraph Coloring

Theorem: For any k -uniform hypergraph H of max-degree Δ ,

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

- 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 $d \leq k(\Delta - 1)$
 - $\Delta \leq \frac{q^{k-1}}{ek} \implies ep(d + 1) \leq 1$ Apply LLL

Lovász Local Lemma (LLL)

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

Lovász Local Lemma (symmetric case):

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

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



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

Lovász Local Lemma (asymmetric case):

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

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

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

Lovász Local Lemma (asymmetric case):

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

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

chain rule

$$\Pr \left[\bigwedge_{i=1}^m \bar{A}_i \right] \stackrel{\text{chain rule}}{=} \prod_{i=1}^m \Pr \left[\bar{A}_i \mid \bigwedge_{j < i} \bar{A}_j \right] = \prod_{i=1}^m \left(1 - \Pr \left[A_i \mid \bigwedge_{j < i} \bar{A}_j \right] \right) \geq \prod_{i=1}^m (1 - \alpha_i)$$

Induction Hypothesis (I.H.):

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

Basis: when $k = 0$, trivial

LLL
cond.:

$$\exists \alpha_1, \dots, \alpha_m \in [0,1) \text{ s.t.}$$

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

I.H.: $\Pr[A_i \mid \bar{A}_{j_1} \cdots \bar{A}_{j_k}] \leq \alpha_i$ holds for all smaller k

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

$$\Pr[A_i \mid \underbrace{\bar{A}_{j_1} \cdots \bar{A}_{j_l}}_{\text{neighbors}} \mid \underbrace{\bar{A}_{j_{l+1}} \cdots \bar{A}_{j_k}}_{\text{non-neighbors}}] = \frac{\Pr[A_i \bar{A}_{j_1} \cdots \bar{A}_{j_l} \mid \bar{A}_{j_{l+1}} \cdots \bar{A}_{j_k}]}{\Pr[\bar{A}_{j_1} \cdots \bar{A}_{j_l} \mid \bar{A}_{j_{l+1}} \cdots \bar{A}_{j_k}]} \leq \alpha_i$$

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

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

$$(\text{I.H.}) \geq \prod_{r=1}^l (1 - \alpha_{j_r}) \geq \prod_{A_j \in \Gamma(A_i)} (1 - \alpha_j)$$

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

Lovász Local Lemma (asymmetric case):

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

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

chain rule

$$\Pr \left[\bigwedge_{i=1}^m \bar{A}_i \right] \stackrel{\text{chain rule}}{=} \prod_{i=1}^m \Pr \left[\bar{A}_i \mid \bigwedge_{j < i} \bar{A}_j \right] = \prod_{i=1}^m \left(1 - \Pr \left[A_i \mid \bigwedge_{j < i} \bar{A}_j \right] \right) \geq \prod_{i=1}^m (1 - \alpha_i)$$

Induction Hypothesis (I.H.):

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

Lovász Local Lemma (LLL)

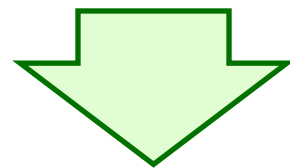
- $A_1, \dots, A_m$ has a dependency graph given by neighborhoods $\Gamma(\cdot)$

Lovász Local Lemma (asymmetric case):

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

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

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



$$\alpha_1 = \dots = \alpha_m = \frac{1}{2d}$$

symmetric case:

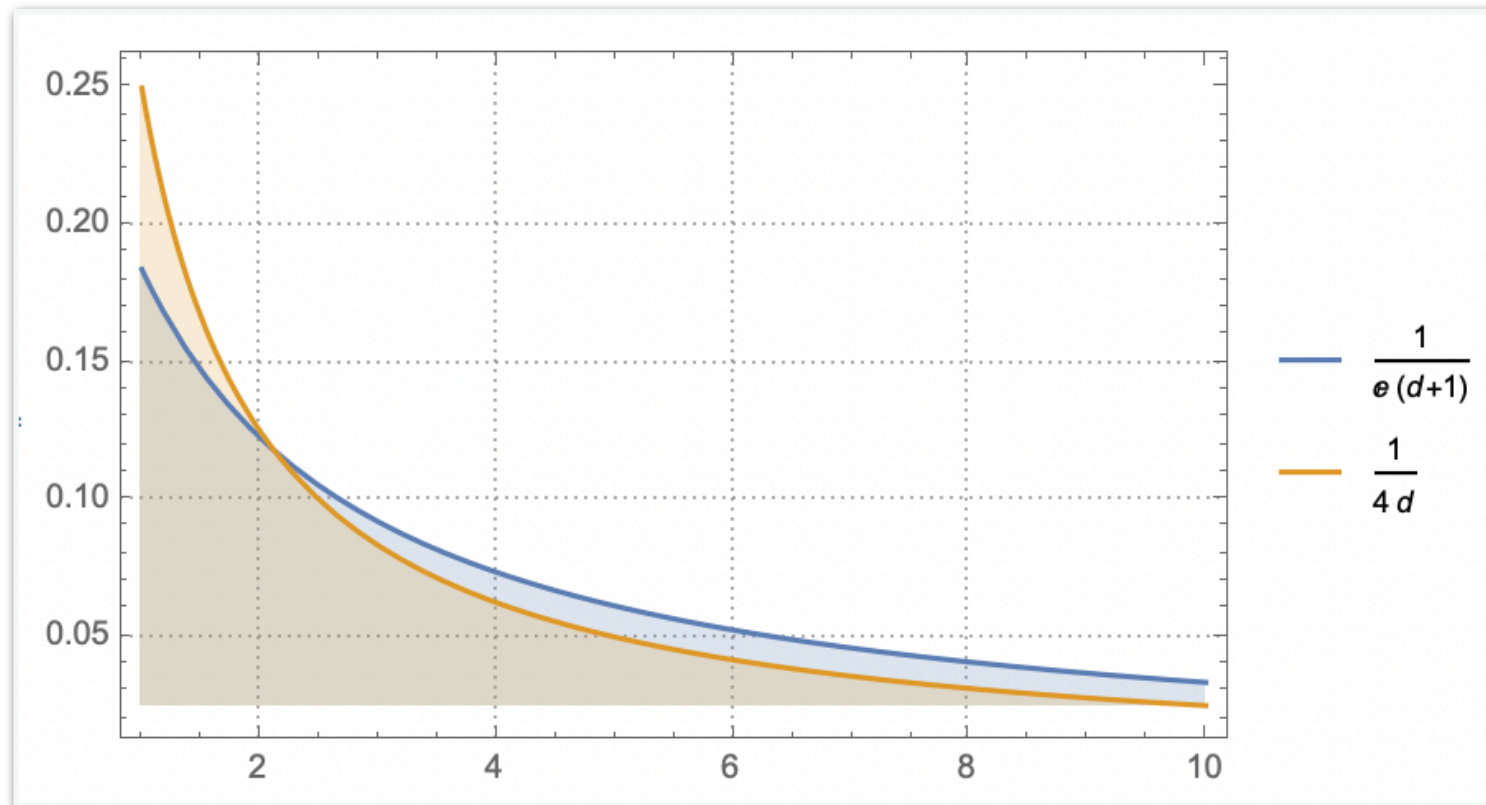
$$p \triangleq \max_i \Pr[A_i]$$

$$d \triangleq \max_i |\Gamma(i)|$$

$$\left. \begin{array}{l} ep(d+1) \leq 1 \\ \text{or} \\ 4pd \leq 1 \end{array} \right\} \implies \Pr \left[\bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

Lovász Local Lemma (LLL)

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



case):

$$\Pr \left[\bigwedge_{i=1}^m \bar{A}_i \right] \geq \prod_{i=1}^m (1 - \alpha_i)$$

$$\alpha_1 = \dots = \alpha_m = \frac{1}{2d}$$

symmetric case:

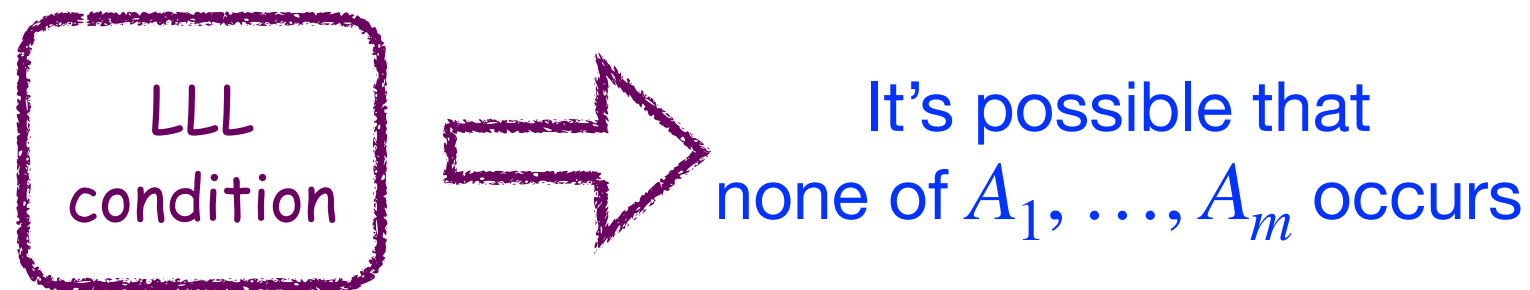
$$p \triangleq \max_i \Pr[A_i]$$

$$d \triangleq \max_i |\Gamma(i)|$$

$$\left. \begin{array}{l} ep(d+1) \leq 1 \\ \text{or} \\ 4pd \leq 1 \end{array} \right\} \Rightarrow \Pr \left[\bigwedge_{i=1}^m \bar{A}_i \right] > 0$$

Summary

- LLL establishes the following implication:



knowing only the probabilities and dependency graph of $A_1, \dots, A_m$

- 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

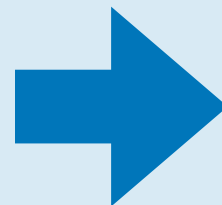
Algorithmic LLL (abstract version)

- “Bad” events $A_1, \dots, A_m$ in a probability space $(\Omega, \Sigma, \text{Pr})$

Lovász Local Lemma:

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

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



$$\exists \sigma \in \Omega,$$

avoid all $A_1, \dots, A_m$

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

Give an efficient algorithm:

find such a good sample $\sigma \in \Omega$
avoiding all bad events $A_1, \dots, A_m$

Algorithmic LLL (variable version)

Variable framework for LLL (CSP with independent variables):

- **mutually independent** random variables $\mathcal{X} = \{X_1, \dots, X_n\}$
- bad events $\mathcal{A} = \{A_1, \dots, A_m\}$, 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 $X_1, \dots, X_n$
avoiding all bad events $A_1, \dots, A_m$

The Moser-Tardos Algorithm

Variable framework for LLL (CSP with independent variables):

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

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

while $\exists$ a bad event A_i that occurs:

resample all $X_j \in \text{vbl}(A_i)$;

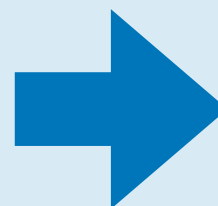
Assume **oracles** for:

- draw ind. samples of X_j
- check if A_i occurs

Theorem [Moser-Tardos 2010]:

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

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



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 $\mathcal{X} = \{X_1, \dots, X_n\}$
- bad events $\mathcal{A} = \{A_1, \dots, A_m\}$, where $A_i \in \mathcal{A}$ is determined by **vbl**(A_i) $\subseteq \mathcal{X}$

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

while $\exists$ a bad event A_i that occurs:

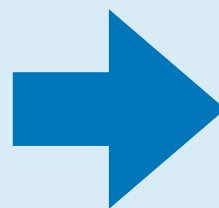
resample all $X_j \in \text{vbl}(A_i)$;

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

Execution Log

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

while $\exists$ a bad event A_i that occurs:

resample all $X_j \in \text{vbl}(A_i)$;

Execution Log (exe-log) Λ of the M-T algorithm:

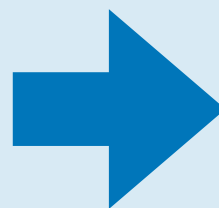
$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A} = \{A_1, \dots, A_m\}$$

random sequence of **resampled** bad events

Theorem [Moser-Tardos 2010]:

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

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



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

Execution Log

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

while $\exists$ a bad event A_i that occurs:

resample all $X_j \in \text{vbl}(A_i)$;

Execution Log (exe-log) Λ of the M-T algorithm:

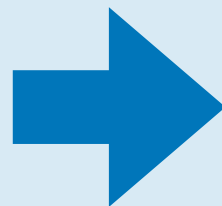
$$\Lambda_1, \Lambda_2, \Lambda_3, \dots \in \mathcal{A} = \{A_1, \dots, A_m\}$$

random sequence of **resampled** bad events

Theorem [Moser-Tardos 2010]:

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

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



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

Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,D,B,A,C,A,D, ...

Resampling table:

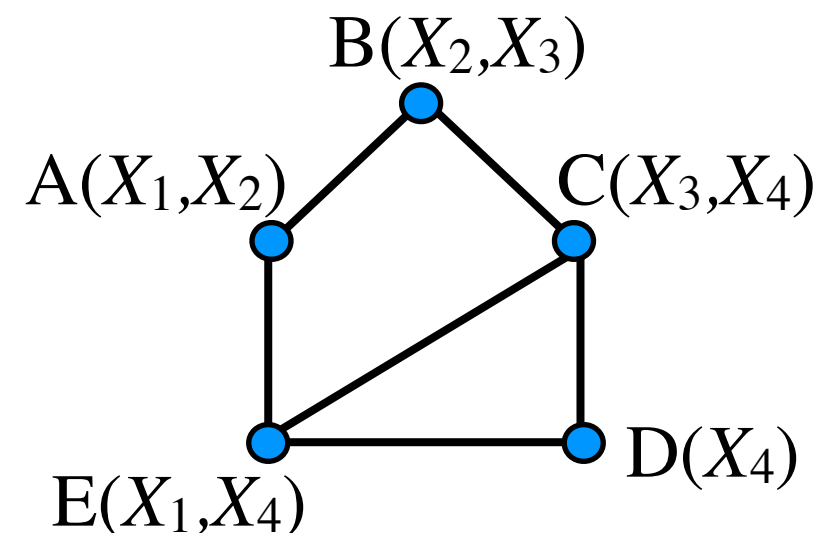
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

while $\exists$ a bad event A_i that occurs:

resample all $X_j \in \text{vbl}(A_i)$;

Exe-log Λ :

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

random sequence of resampled bad events

Exe-log Λ : D,C,E,D,B,A,C,A,D, ...

Resampling table:

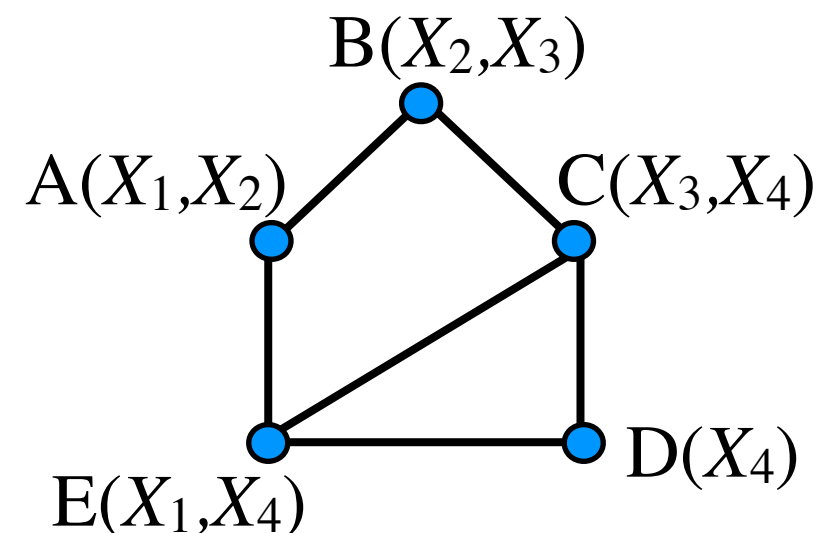
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : **D**,C,E,D,B,A,C,A,D, ...

Resampling table:

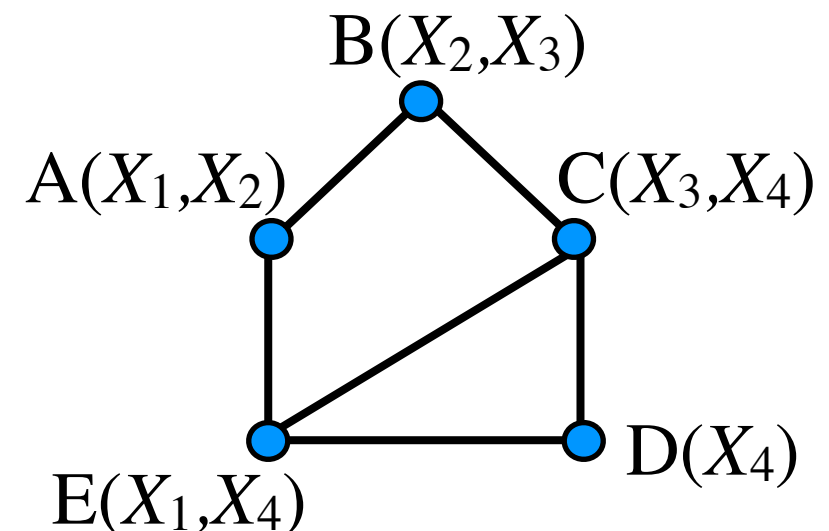
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : **$X_4^{(0)}$** , $X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D, **C**, E, D, B, A, C, A, D, ...

Resampling table:

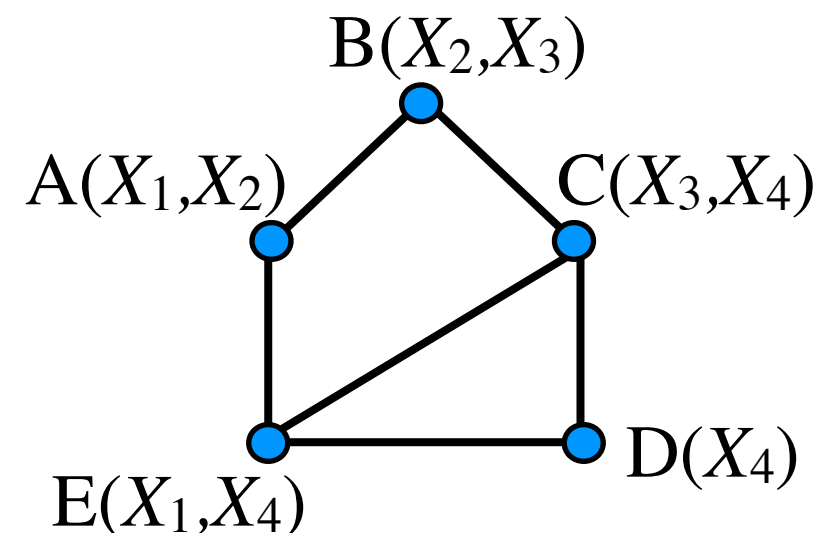
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : **$X_3^{(0)}$** , $X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}$, **$X_4^{(1)}$** , $X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,**E**,D,B,A,C,A,D, ...

Resampling table:

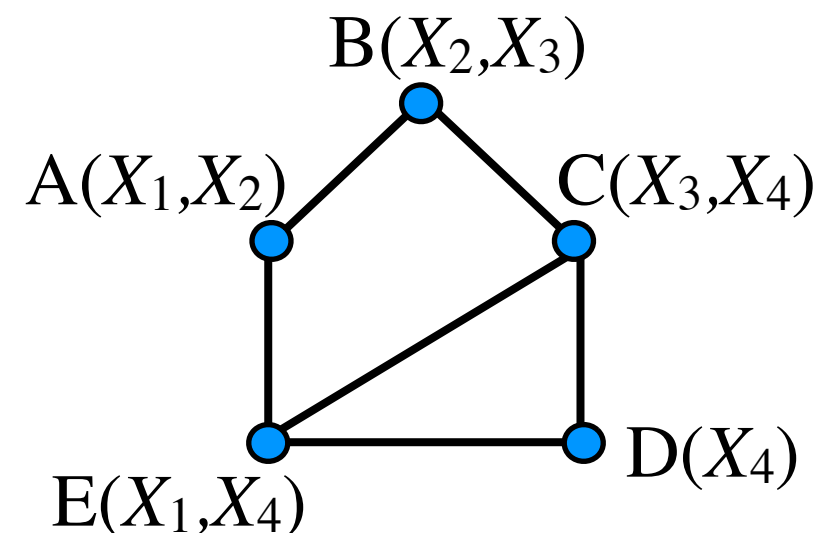
X_1 : **$X_1^{(0)}$** , $X_1^{(1)}$, $X_1^{(2)}$, $X_1^{(3)}$, $X_1^{(4)}$, ...

X_2 : $X_2^{(0)}$, $X_2^{(1)}$, $X_2^{(2)}$, $X_2^{(3)}$, $X_2^{(4)}$, ...

X_3 : **$X_3^{(0)}$** , $X_3^{(1)}$, $X_3^{(2)}$, $X_3^{(3)}$, $X_3^{(4)}$, ...

X_4 : $X_4^{(0)}$, $X_4^{(1)}$, **$X_4^{(2)}$** , $X_4^{(3)}$, $X_4^{(4)}$, ...

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,**D**,B,A,C,A,D, ...

Resampling table:

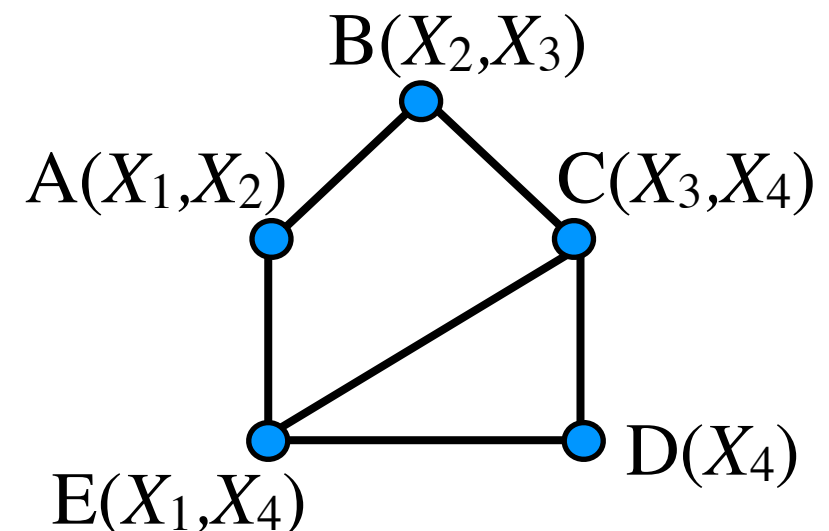
X_1 : **$X_1^{(0)}$** , $X_1^{(1)}$, $X_1^{(2)}$, $X_1^{(3)}$, $X_1^{(4)}$, ...

X_2 : $X_2^{(0)}$, $X_2^{(1)}$, $X_2^{(2)}$, $X_2^{(3)}$, $X_2^{(4)}$, ...

X_3 : **$X_3^{(0)}$** , $X_3^{(1)}$, $X_3^{(2)}$, $X_3^{(3)}$, $X_3^{(4)}$, ...

X_4 : $X_4^{(0)}$, $X_4^{(1)}$, $X_4^{(2)}$, **$X_4^{(3)}$** , $X_4^{(4)}$, ...

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,D,**B**,A,C,A,D, ...

Resampling table:

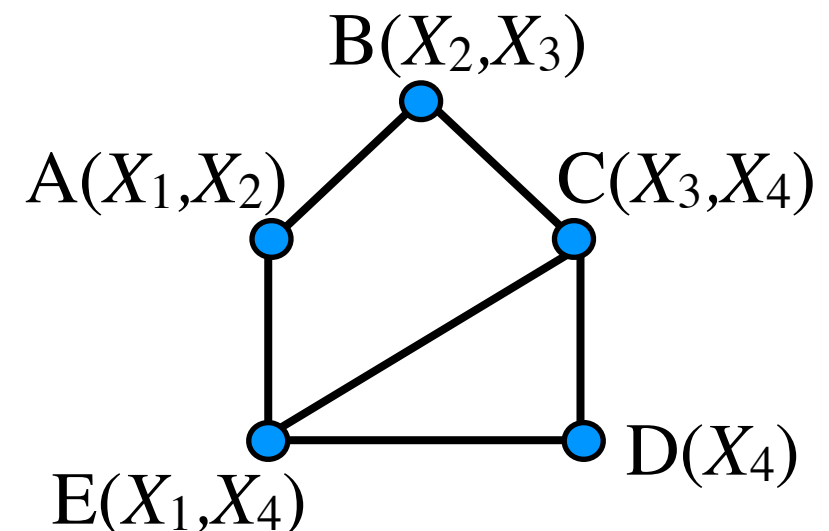
X_1 : $X_1^{(0)}$, $X_1^{(1)}$, $X_1^{(2)}$, $X_1^{(3)}$, $X_1^{(4)}$, ...

X_2 : $X_2^{(0)}$, $X_2^{(1)}$, $X_2^{(2)}$, $X_2^{(3)}$, $X_2^{(4)}$, ...

X_3 : $X_3^{(0)}$, $X_3^{(1)}$, $X_3^{(2)}$, $X_3^{(3)}$, $X_3^{(4)}$, ...

X_4 : $X_4^{(0)}$, $X_4^{(1)}$, $X_4^{(2)}$, $X_4^{(3)}$, $X_4^{(4)}$, ...

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,D,B,**A**,C,A,D, ...

Resampling table:

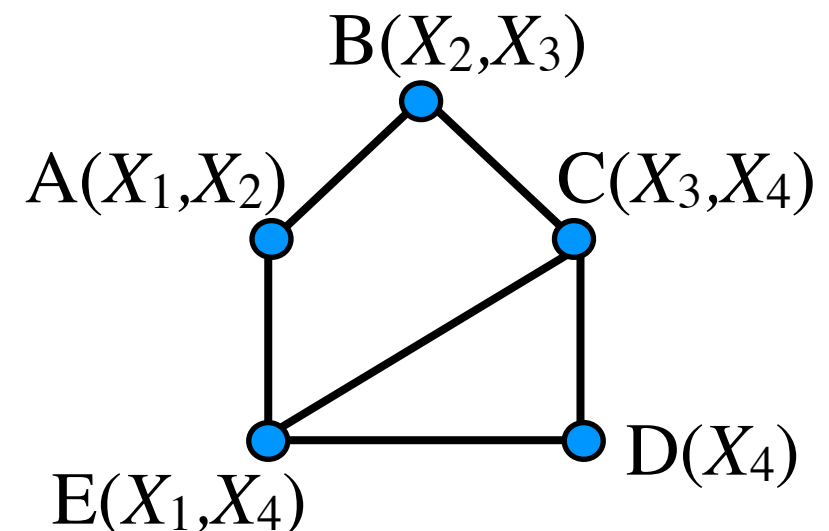
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,D,B,A,C,A,D, ...

Resampling table:

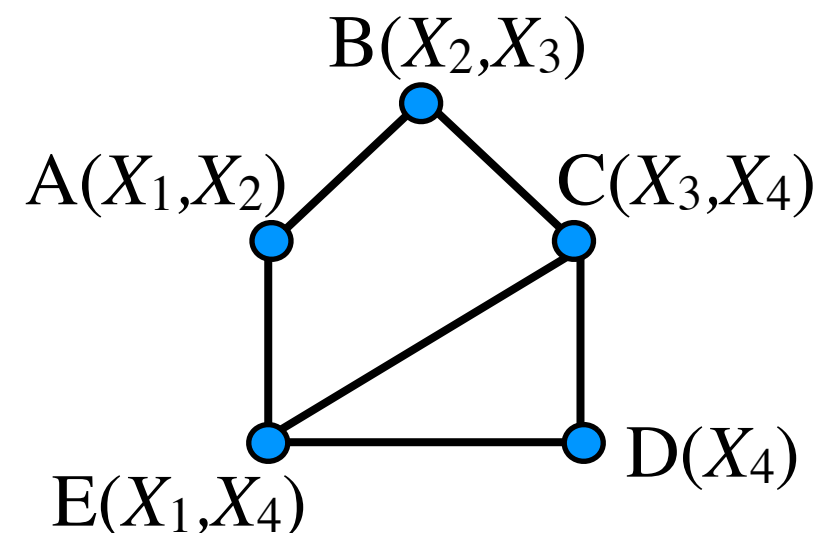
X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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



Resampling Table

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Exe-log Λ : D,C,E,D,B,A,C,**A**,D, ...

Resampling table:

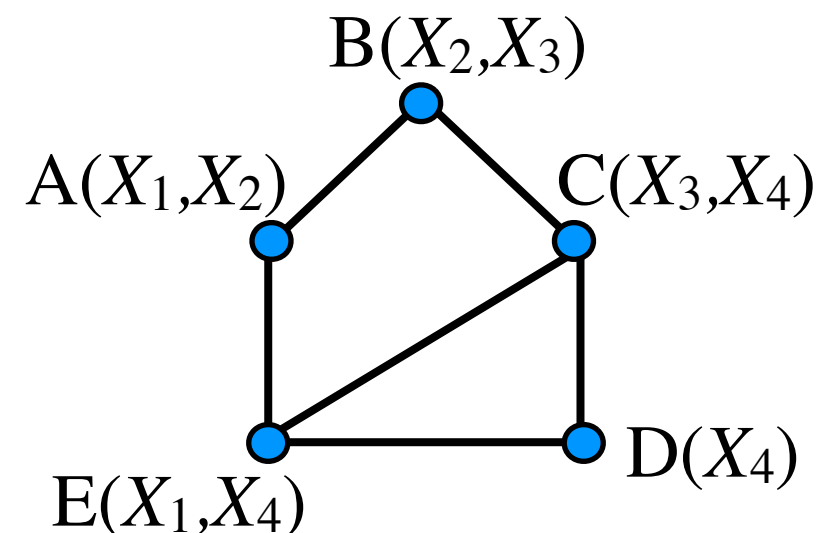
X_1 : $X_1^{(0)}, X_1^{(1)}, \text{X}_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

X_2 : $X_2^{(0)}, X_2^{(1)}, \text{X}_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, \text{X}_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, \text{X}_4^{(4)}, \dots$

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



Witness Tree

Moser-Tardos Algorithm:

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

Exe-log Λ :

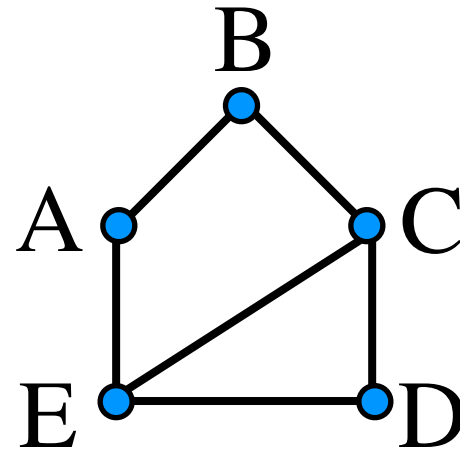
$\Lambda_1, \Lambda_2, \dots \in \mathcal{A} = \{A_1, \dots, A_m\}$
random sequence of resampled bad events

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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

Inclusive neighborhood: $\Gamma^+(A) \triangleq \Gamma(A) \cup \{A\}$

dependency graph:



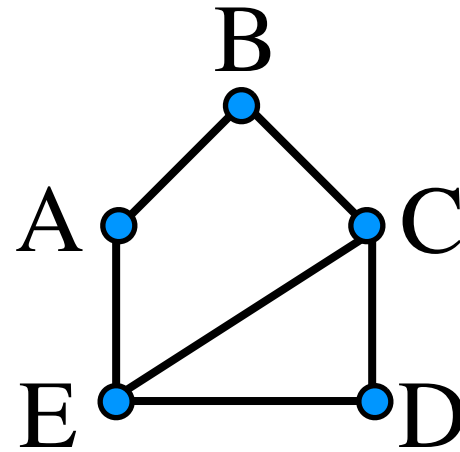
exe-log Λ : D, C, E, D, B, A, C, A, D, ...

$T(\Lambda, 8):$  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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



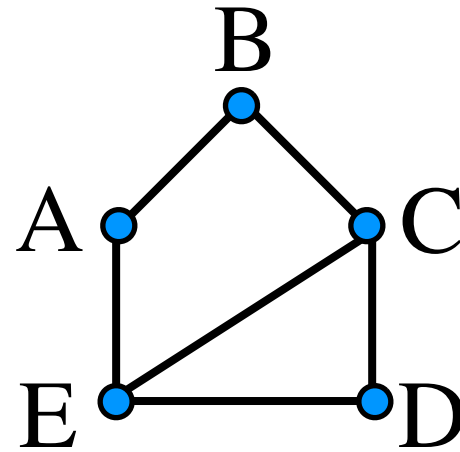
exe-log Λ : D, C, E, D, B, A, C, A, D, ...

$T(\Lambda, 8)$:  A

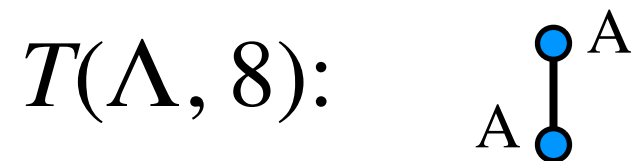
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



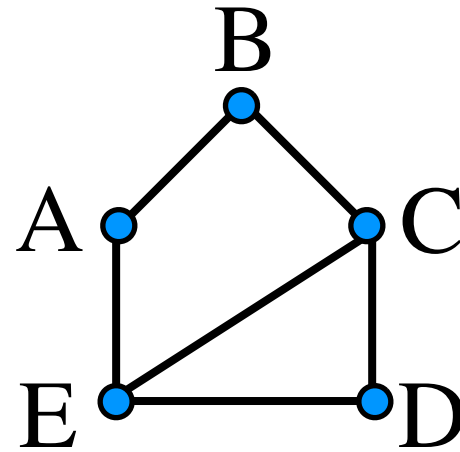
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

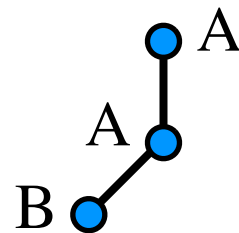
dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...



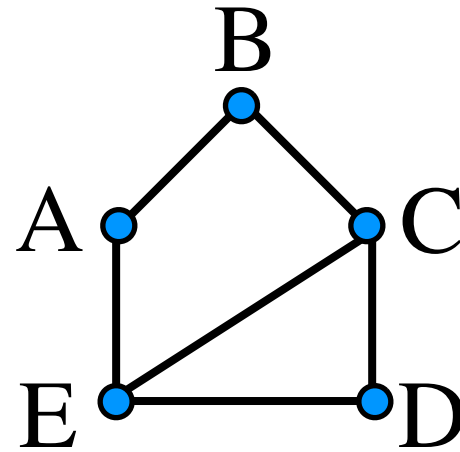
$T(\Lambda, 8)$:



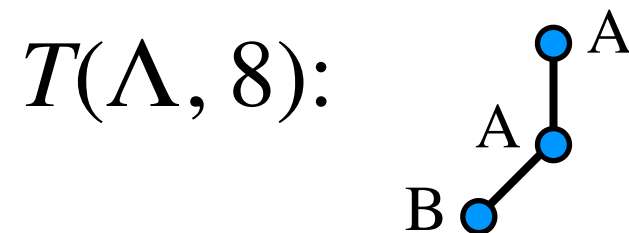
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



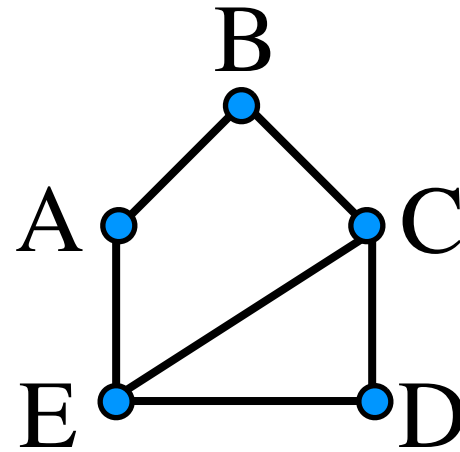
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



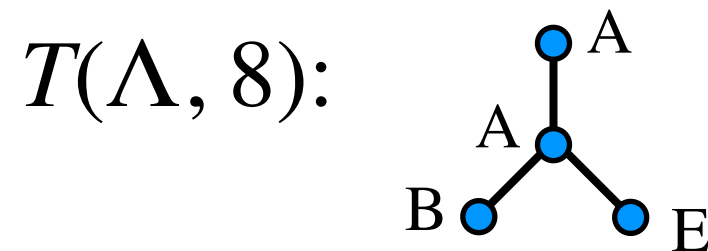
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



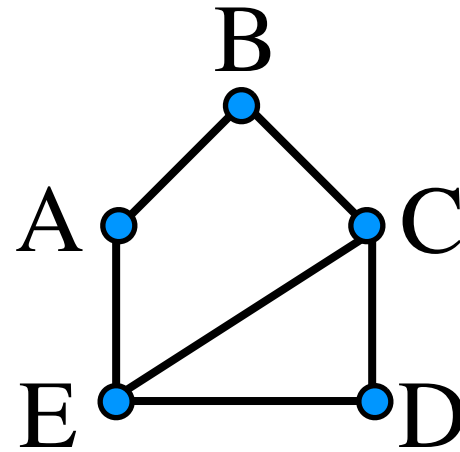
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



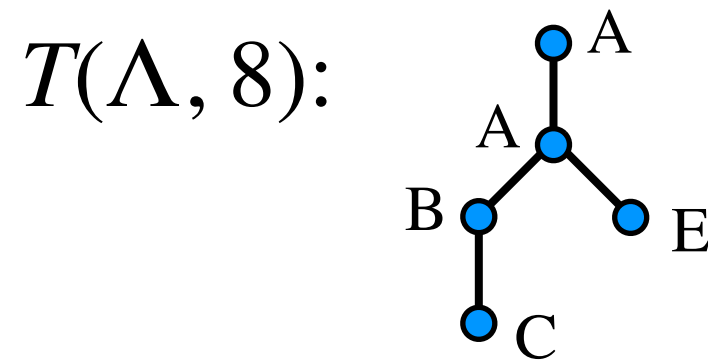
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



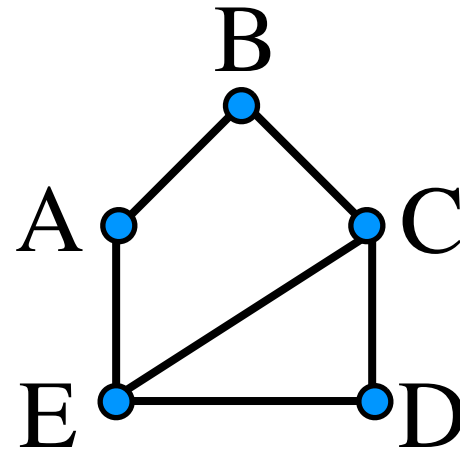
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

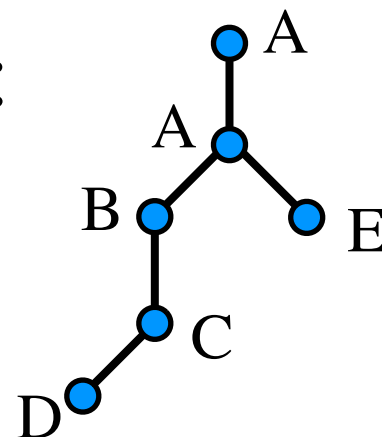
dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

△
△

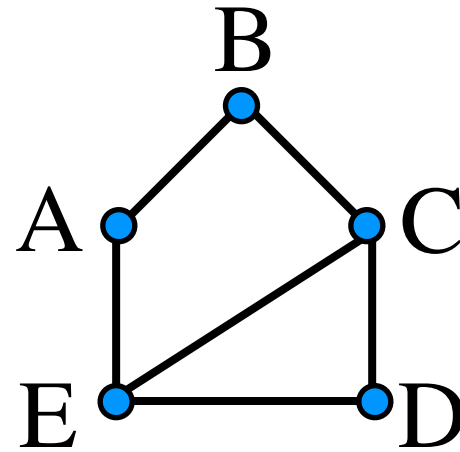
$T(\Lambda, 8)$:



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

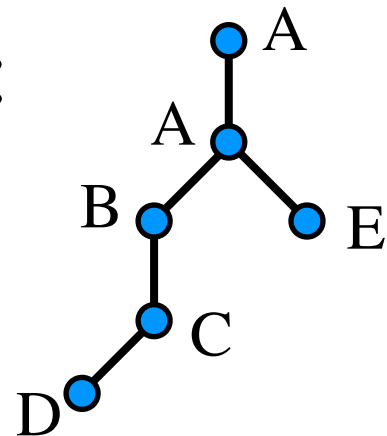
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

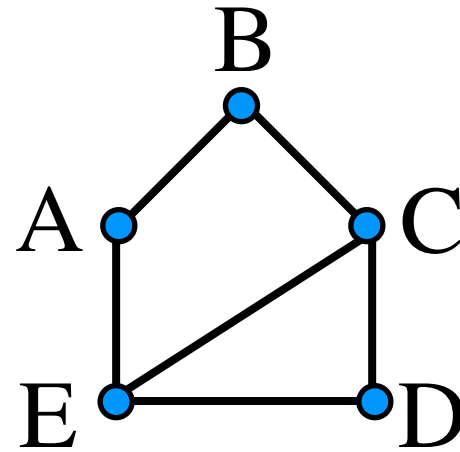
$T(\Lambda, 8)$:



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

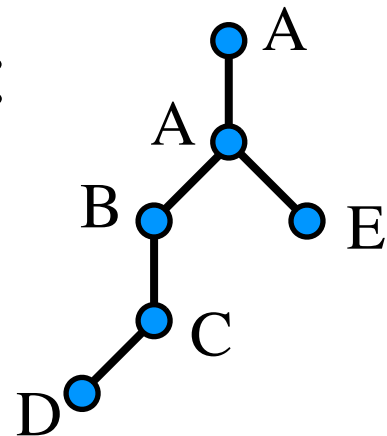
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, Δ D, ...

$T(\Lambda, 8)$:



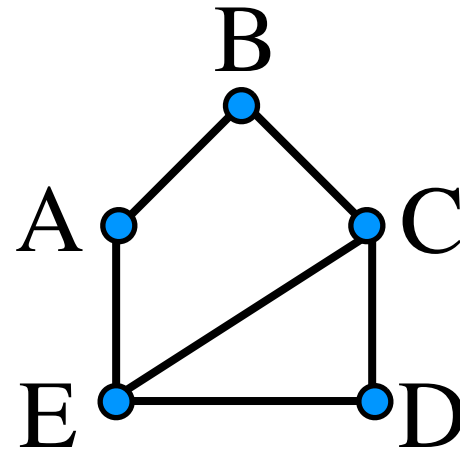
$T(\Lambda, 9)$:



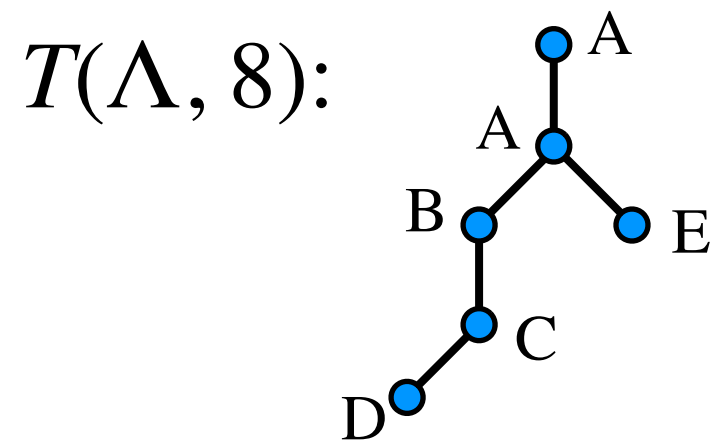
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

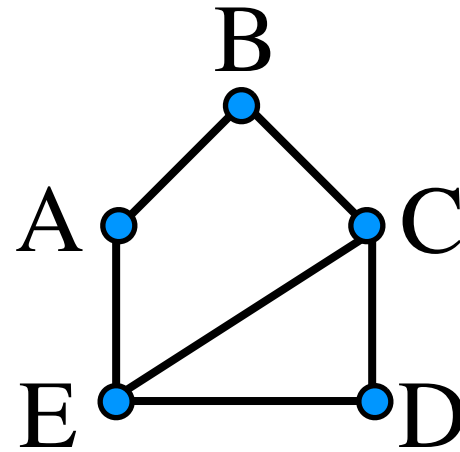


$T(\Lambda, 9)$:  D

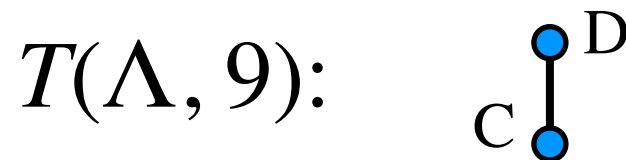
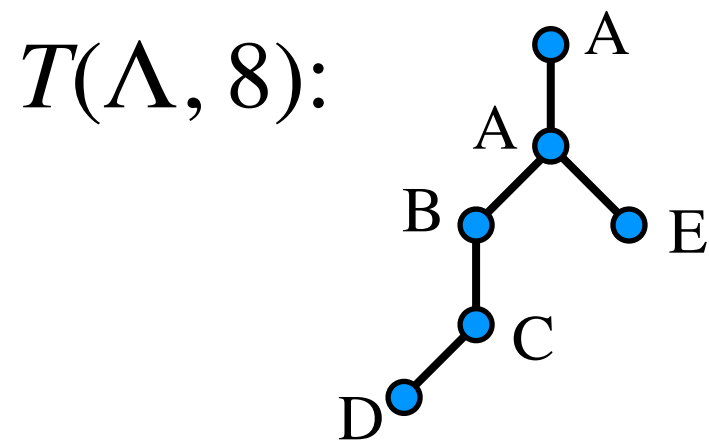
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



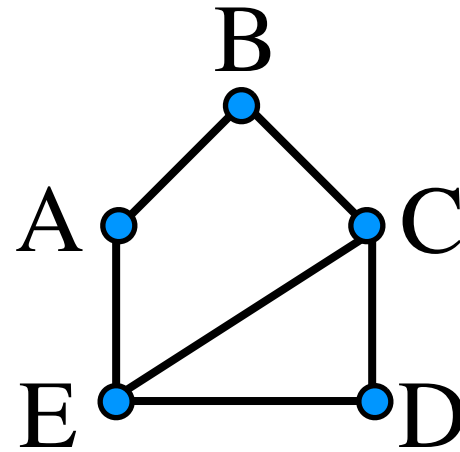
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



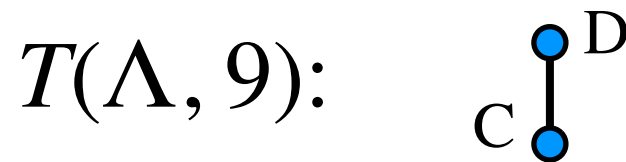
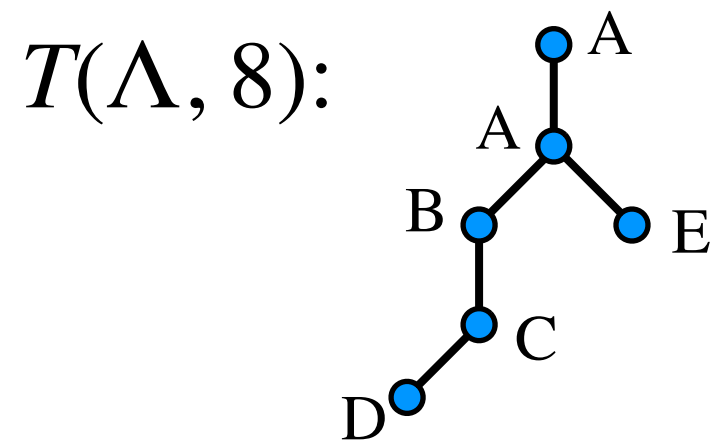
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



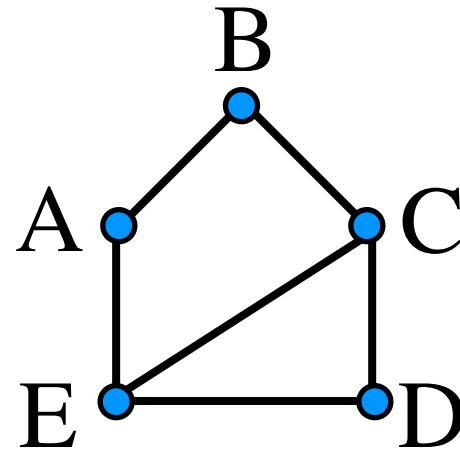
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



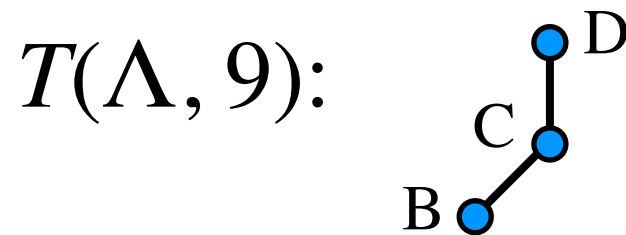
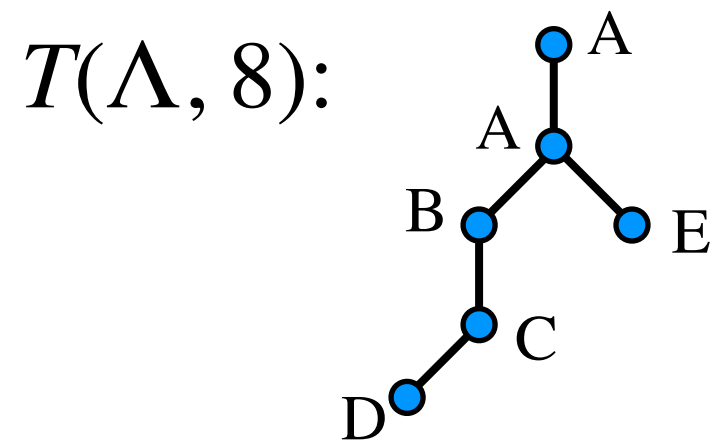
Witness tree $T(\Lambda, t)$: each node u with label $A_{[u]} \in \mathcal{A}$, siblings have *distinct* labels

- initially, T contains a single **root** r with Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



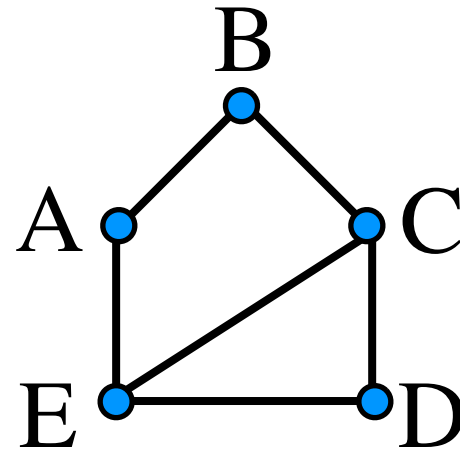
exe-log Λ : D, C, E, D, B, A, C, A, D, ...



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

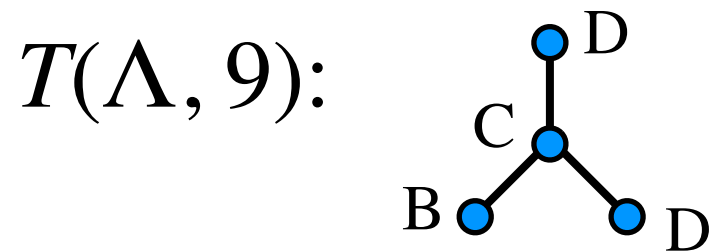
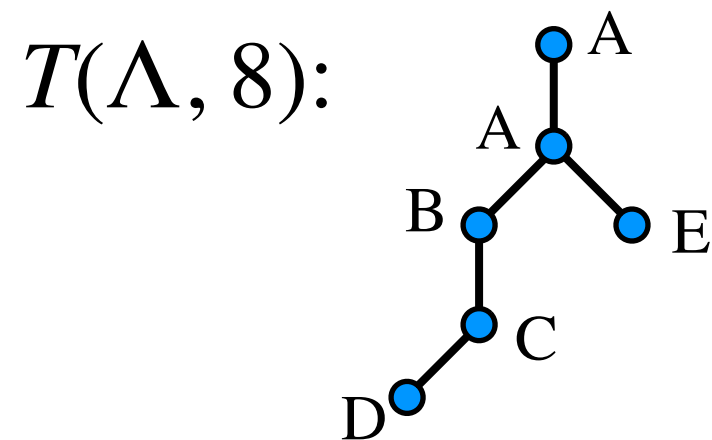
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

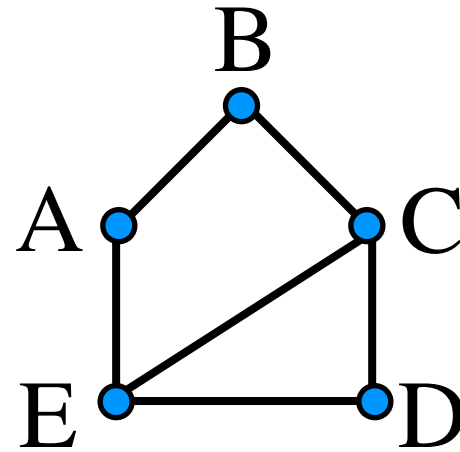
△
△



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

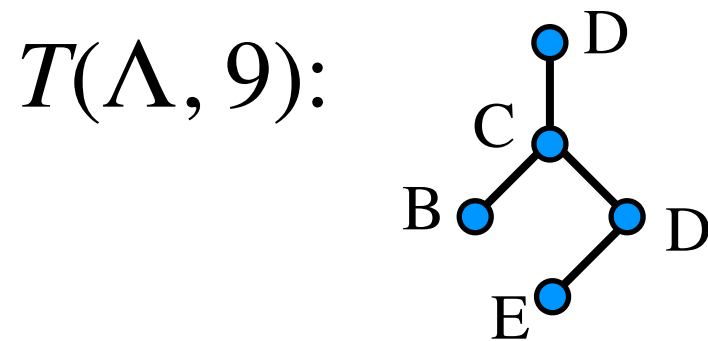
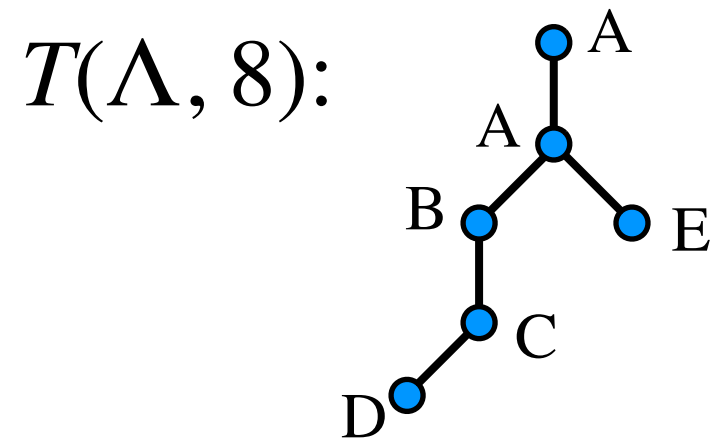
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

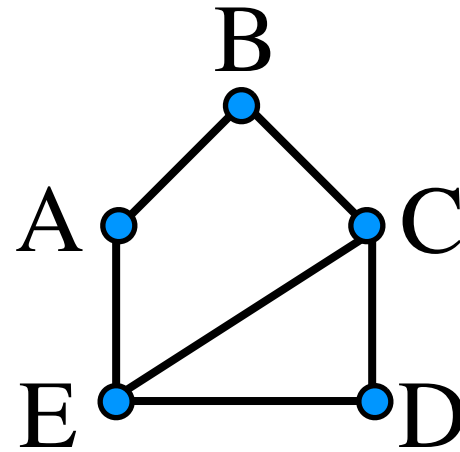
△
△



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

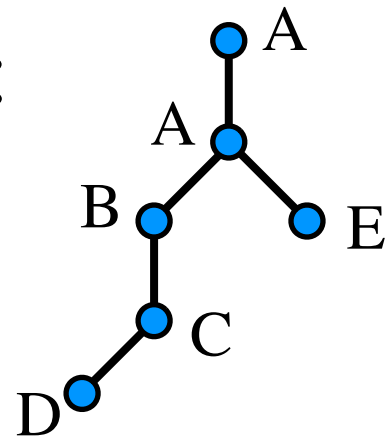
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:

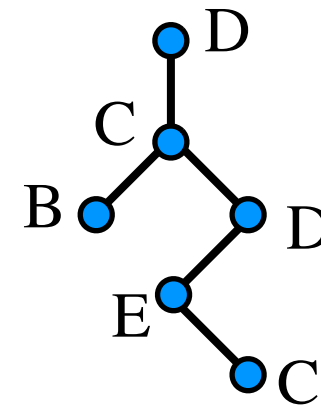


exe-log Λ : D, C, E, D, B, A, C, A, D, ...

$T(\Lambda, 8)$:



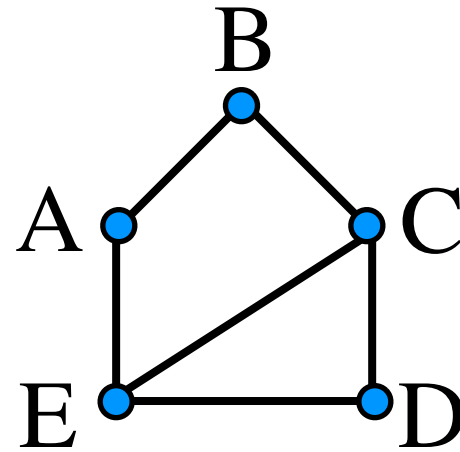
$T(\Lambda, 9)$:



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

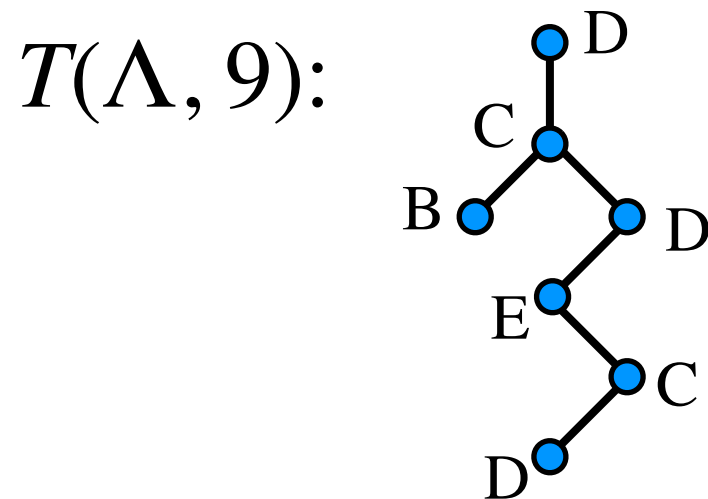
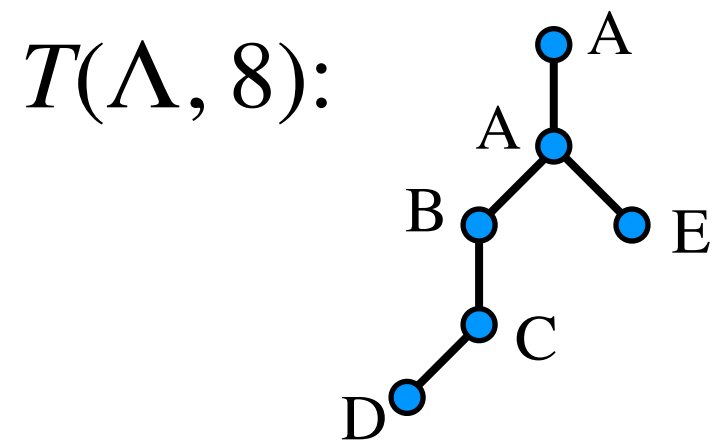
- initially, T contains a single **root** r with Λ_t
- for $i = t - 1$ to 1:
 - if $\Lambda_i \in \Gamma^+(A_{[u]})$ for some node $u \in T$
 - add child $v \rightarrow$ *deepest* such u , labeled with Λ_i
- $T(\Lambda, t)$ is the resulting T

dependency graph:



exe-log Λ : D, C, E, D, B, A, C, A, D, ...

△
△



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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

Moser-Tardos Algorithm:

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

Exe-log Λ :

$\Lambda_1, \Lambda_2, \dots \in \mathcal{A} = \{A_1, \dots, A_m\}$
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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

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

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

linearity of expectation:

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

Moser-Tardos Algorithm:

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

Exe-log Λ :

$\Lambda_1, \Lambda_2, \dots \in \mathcal{A} = \{A_1, \dots, A_m\}$
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 Λ_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 Λ_i
- $T(\Lambda, t)$ is the resulting T

Lemma 1 (coupling). For any particular witness tree τ :

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

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

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

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

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

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

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

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

$$\text{Convergence:} \leq \frac{\alpha_i}{1 - \alpha_i}$$

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

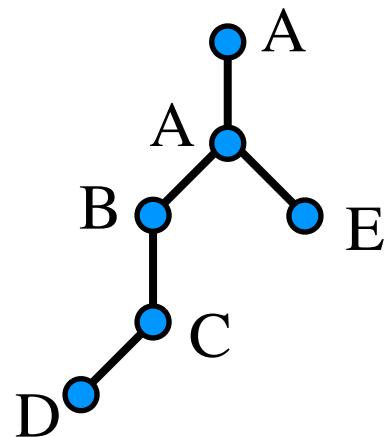
Lemma 1 (coupling). For any particular witness tree τ :

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

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

Simulation of witness tree τ :

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



$$\begin{aligned} & \Pr[\text{simulation of } \tau \text{ succeeds}] \\ &= \Pr[D] \cdot \Pr[C] \cdot \Pr[B] \cdot \Pr[E] \cdot \Pr[A] \cdot \Pr[A] \end{aligned}$$

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

$X_1 : X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

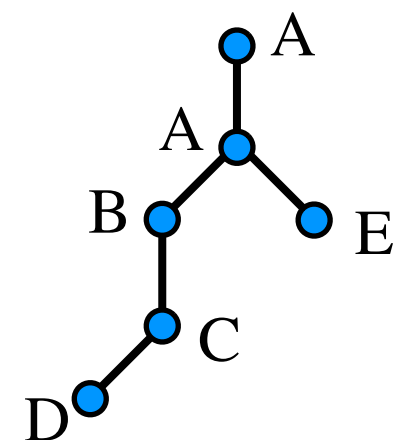
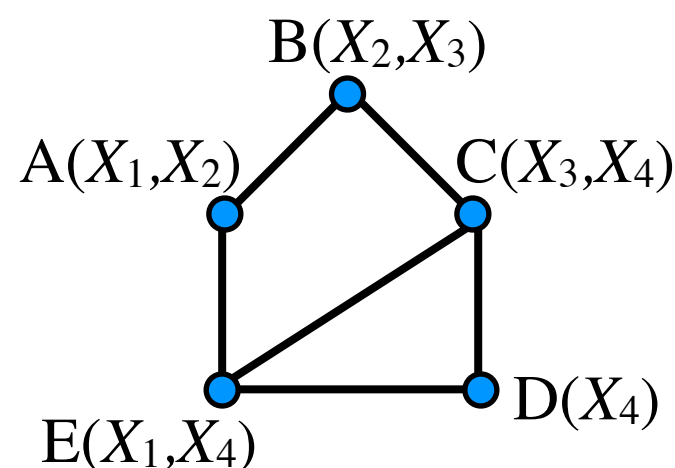
$X_2 : X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

$X_3 : X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

$X_4 : X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

$X_j^{(t)} : t\text{-th sampling of variable } X_j$

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

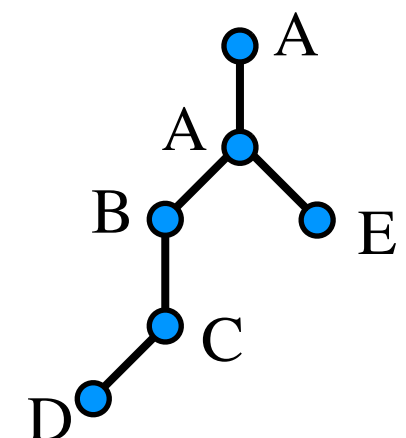
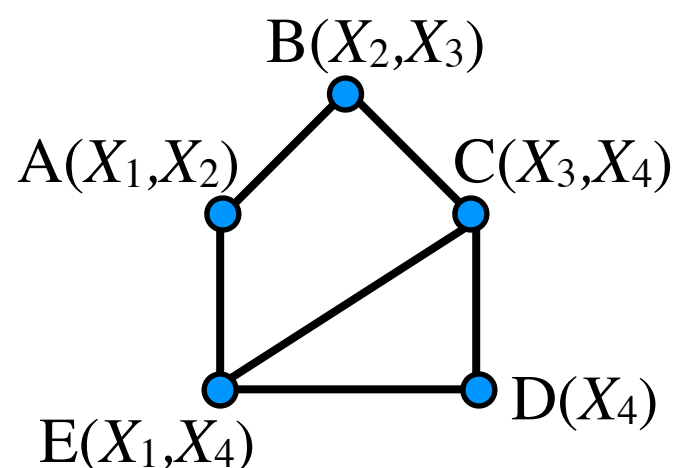
$$\Pr_{\Lambda} \left[\exists t, T(\Lambda, t) = \tau \right] \leq \prod_{u \in \tau} \Pr(A_{[u]})$$

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

Resampling table:

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

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

$X_1 : X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

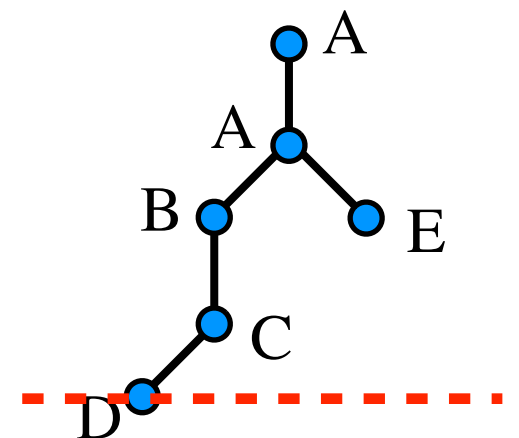
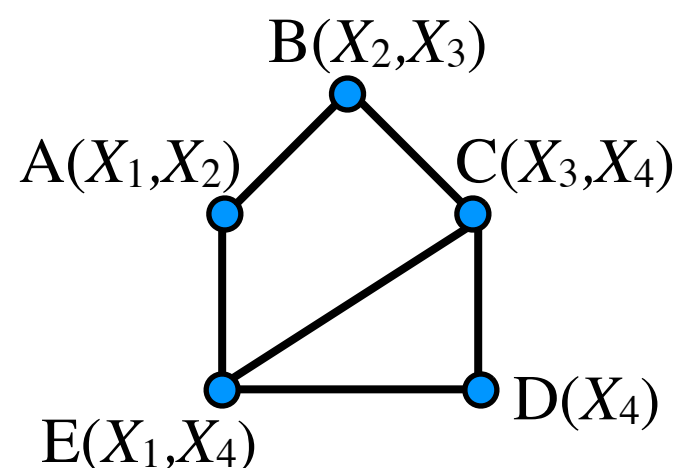
$X_2 : X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

$X_3 : X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

$X_4 : X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

$X_j^{(t)} : t\text{-th sampling of variable } X_j$

exe-log Λ : **D**, C, E, D, B, A, C, A, D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

$X_1 : X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

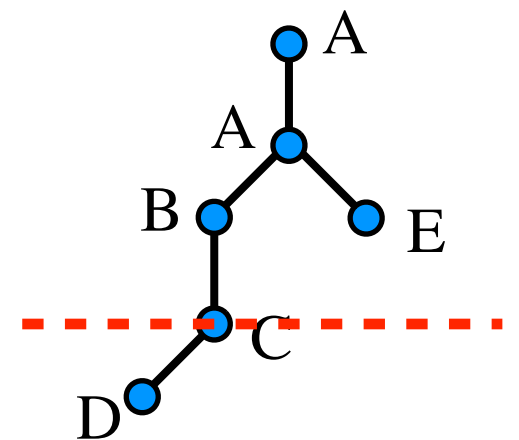
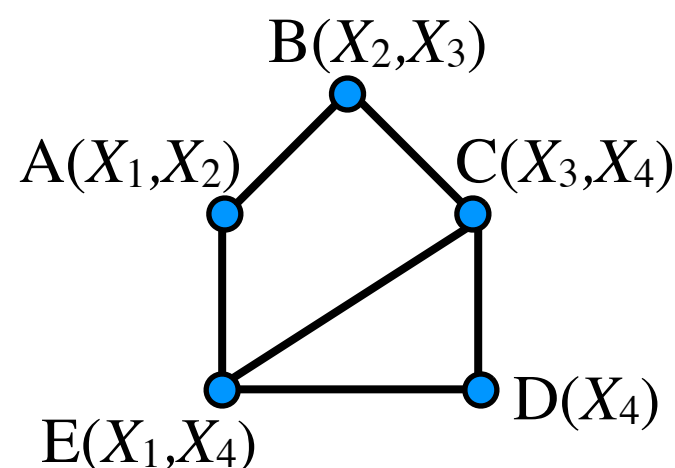
$X_2 : X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

$X_3 : X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

$X_4 : X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

$X_j^{(t)} : t\text{-th sampling of variable } X_j$

exe-log Λ : D, C, E, D, B, A, C, A, D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}$, $X_1^{(1)}$, $X_1^{(2)}$, $X_1^{(3)}$, $X_1^{(4)}$, $\dots$

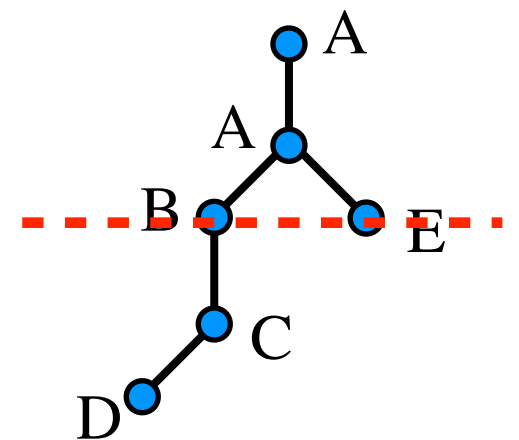
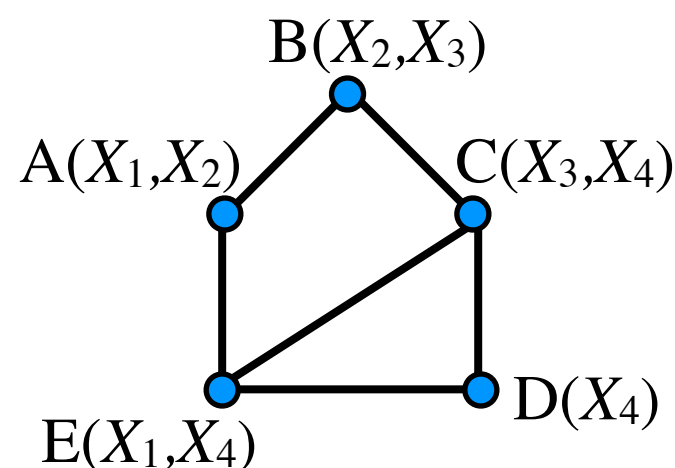
X_2 : $X_2^{(0)}$, $X_2^{(1)}$, $X_2^{(2)}$, $X_2^{(3)}$, $X_2^{(4)}$, $\dots$

X_3 : $X_3^{(0)}$, $X_3^{(1)}$, $X_3^{(2)}$, $X_3^{(3)}$, $X_3^{(4)}$, $\dots$

X_4 : $X_4^{(0)}$, $X_4^{(1)}$, $X_4^{(2)}$, $X_4^{(3)}$, $X_4^{(4)}$, $\dots$

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

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

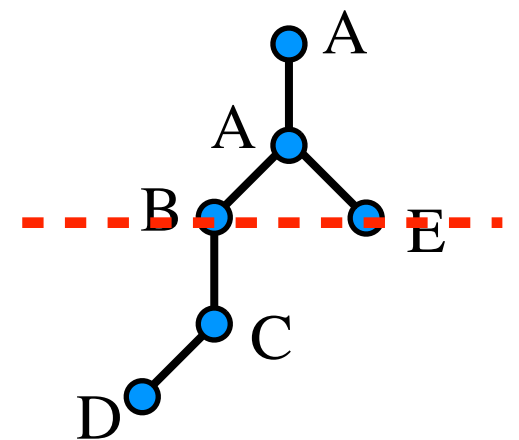
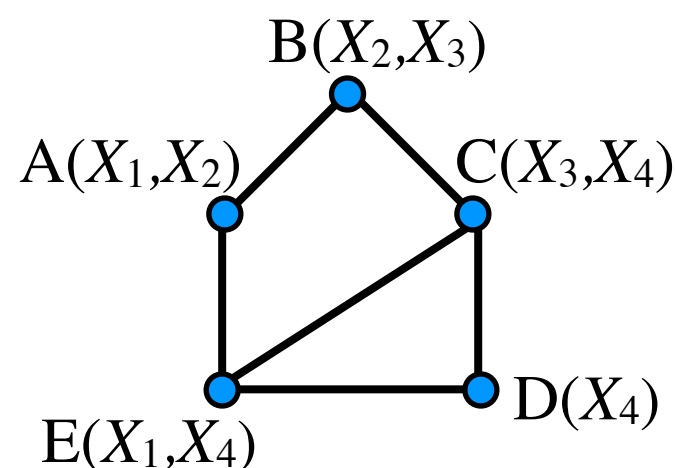
X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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

exe-log Λ : D,C,E,**D**,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

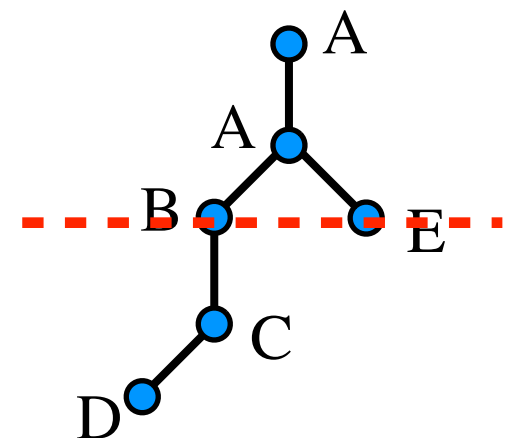
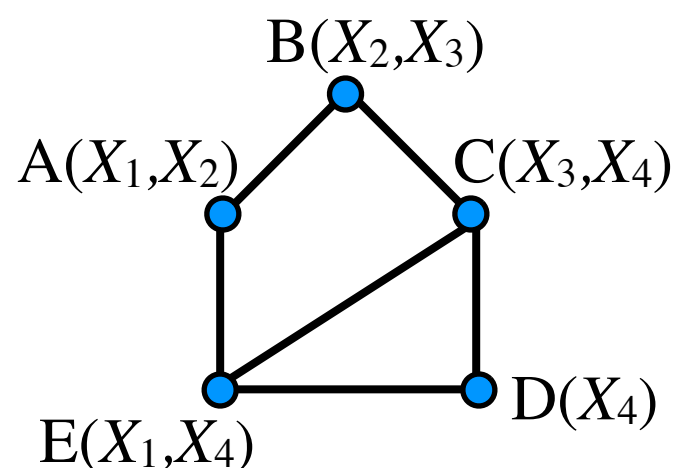
X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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

exe-log Λ : D,C,E,D,**B**,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

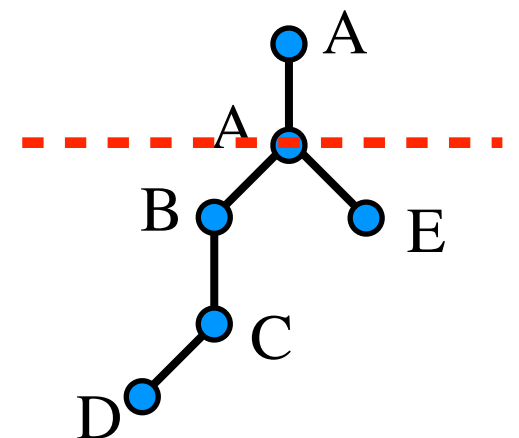
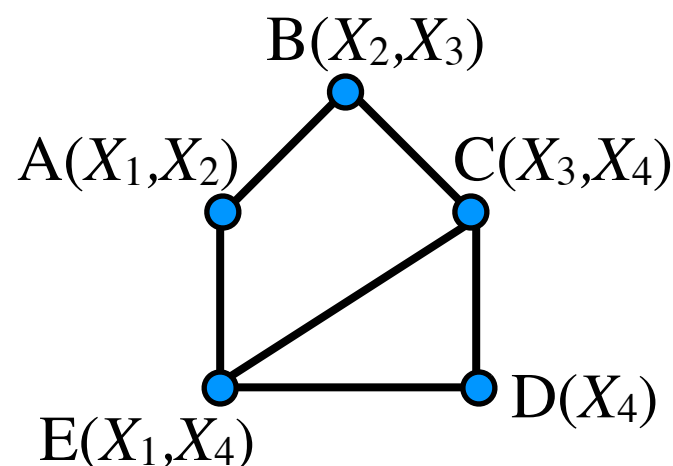
X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

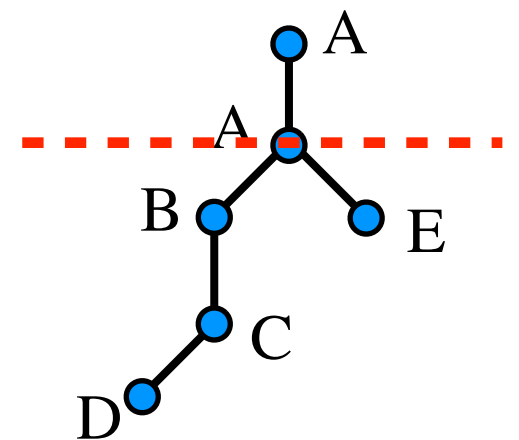
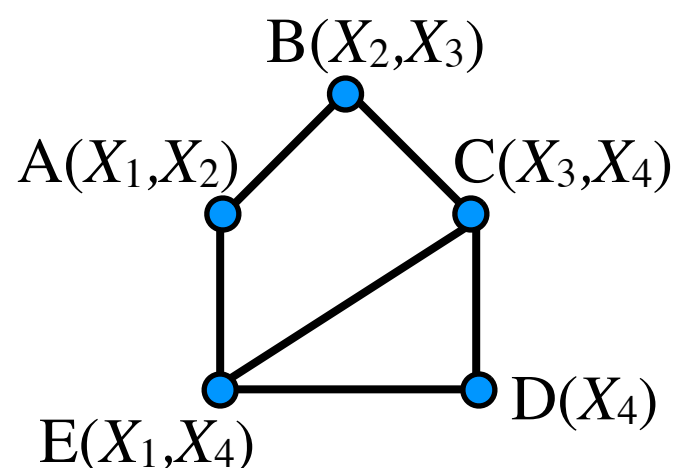
X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

Lemma 1 (coupling). For any particular witness tree τ :

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

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

Resampling table:

X_1 : $X_1^{(0)}, X_1^{(1)}, X_1^{(2)}, X_1^{(3)}, X_1^{(4)}, \dots$

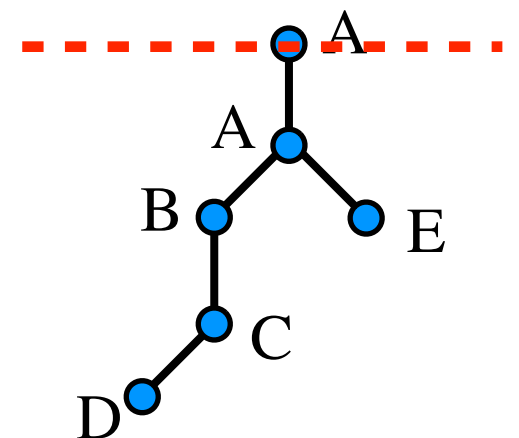
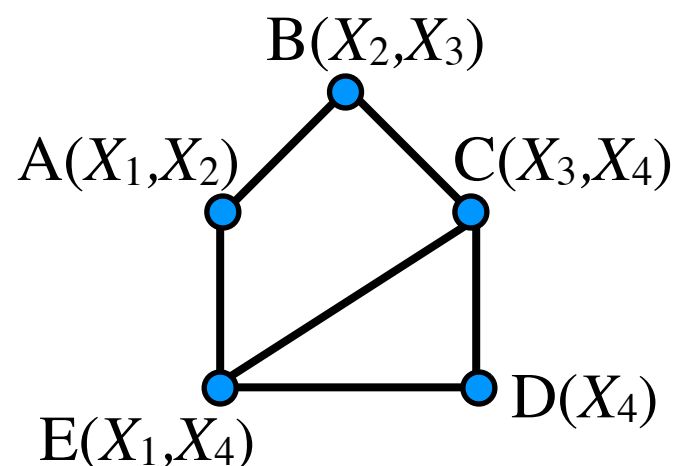
X_2 : $X_2^{(0)}, X_2^{(1)}, X_2^{(2)}, X_2^{(3)}, X_2^{(4)}, \dots$

X_3 : $X_3^{(0)}, X_3^{(1)}, X_3^{(2)}, X_3^{(3)}, X_3^{(4)}, \dots$

X_4 : $X_4^{(0)}, X_4^{(1)}, X_4^{(2)}, X_4^{(3)}, X_4^{(4)}, \dots$

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

exe-log Λ : D,C,E,D,B,A,C,A,D, ...



Moser-Tardos Algorithm:

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

Exe-log Λ :

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

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

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

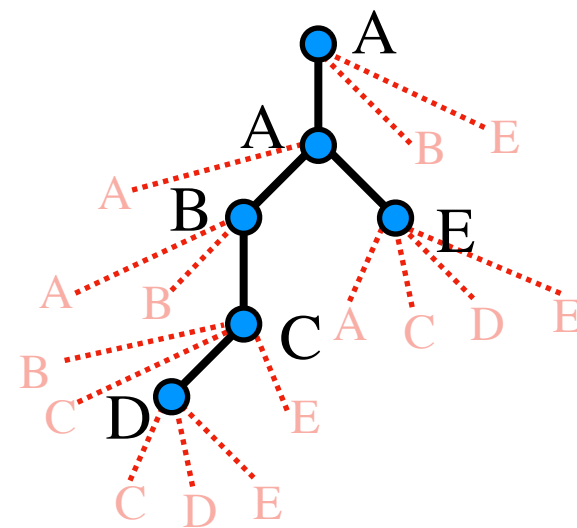
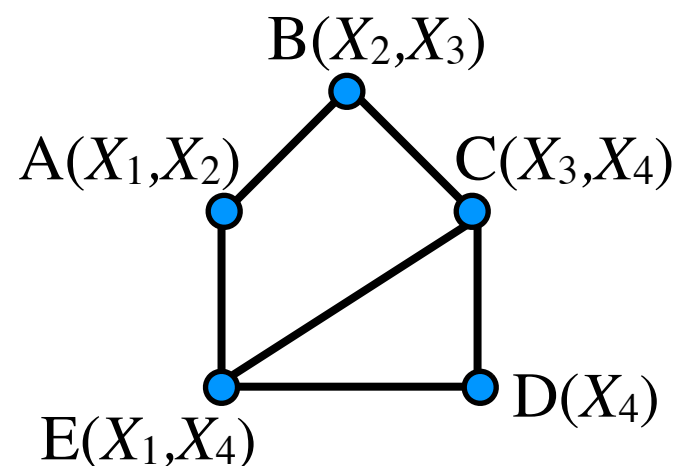
$$(\text{Lemma 1}) \leq \sum_{\tau \in \mathcal{T}_{A_i}} \prod_{u \in \tau} \Pr(A_{[u]})$$

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

$$\text{Convergence:} \leq \frac{\alpha_i}{1 - \alpha_i}$$

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 $i = 1, 2, \dots$:
 - 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. α_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)

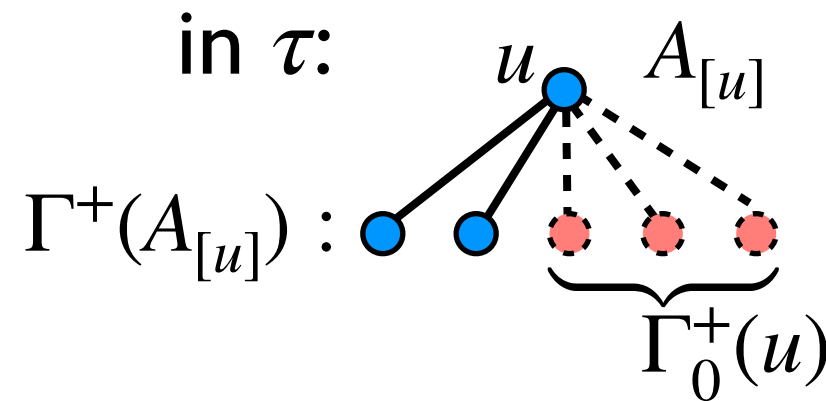
- Grow a random witness tree T_A with root-label A
 - initially, T_A is a single root with label A
 - for $i = 1, 2, \dots$:
 - for every vertex u at depth i (root has depth 1) in T_A
 - for every $A_j \in \Gamma^+(A_{[u]})$:
 - add a new child v to u ind. with prob. α_j and label it with A_j ;
 - stop 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 $\tau \in \mathcal{T}_{A_i}$:

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



$$\begin{aligned} \Pr \left[T_{A_i} = \tau \right] &= \frac{1}{\alpha_i} \prod_{u \in \tau} \left[\alpha_{[u]} \prod_{A_j \in \Gamma_0^+(u)} (1 - \alpha_j) \right] = \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[\frac{\alpha_{[u]}}{1 - \alpha_{[u]}} \prod_{A_j \in \Gamma^+(A_{[u]})} (1 - \alpha_j) \right] \\ &= \frac{1 - \alpha_i}{\alpha_i} \prod_{u \in \tau} \left[\alpha_{[u]} \prod_{A_j \in \Gamma(A_{[u]})} (1 - \alpha_j) \right] \end{aligned}$$

Moser-Tardos Algorithm:

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

Exe-log Λ :

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

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

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

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

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

$$(\text{Lemma 2}) \leq \frac{\alpha_i}{1 - \alpha_i} \sum_{\tau \in \mathcal{T}_{A_i}} \Pr [T_{A_i} = \tau] \leq \frac{\alpha_i}{1 - \alpha_i}$$

The Moser-Tardos Algorithm

Variable framework for LLL (CSP with independent variables):

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

Moser-Tardos Algorithm:

draw independent samples of $X_1, \dots, X_n$;

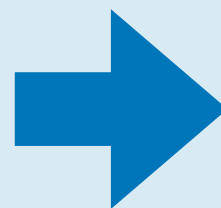
while $\exists$ a bad event A_i that occurs:

resample all $X_j \in \text{vbl}(A_i)$;

Theorem [Moser-Tardos 2010]:

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

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



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

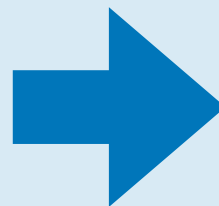
- k -CNF: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$
 $\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overbrace{x_3 \vee \neg x_4 \vee \neg x_5}^{k \text{ variables}})$
- **dependency degree d** : each clause C_i intersects $\leq d$ other clauses
- uniform independent
 $X_1, \dots, X_n \in \{T, F\}$
- bad event A_i : $p = 2^{-k}$
 C_i is violated

Moser-Tardos Algorithm:

draw uniform independent $X_1, \dots, X_n \in \{T, F\}$;
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: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$
 $\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge \overbrace{(x_3 \vee \neg x_4 \vee \neg x_5)}^{k \text{ variables}}$

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;
while $\exists$ a violated clause C_i :
 Fix(C_i);

Fix(C_i):

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

Moser's *Fix-It* Algorithm

- k -CNF: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$
 $\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge \overbrace{(x_3 \vee \neg x_4 \vee \neg x_5)}^{k \text{ variables}}$

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;
while $\exists$ a violated clause C_i :
 Fix(C_i);

Fix(C_i):

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

Theorem [Moser 2009]:

$d < 2^{k-3} \implies$ total # of calls to **Fix**() 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:

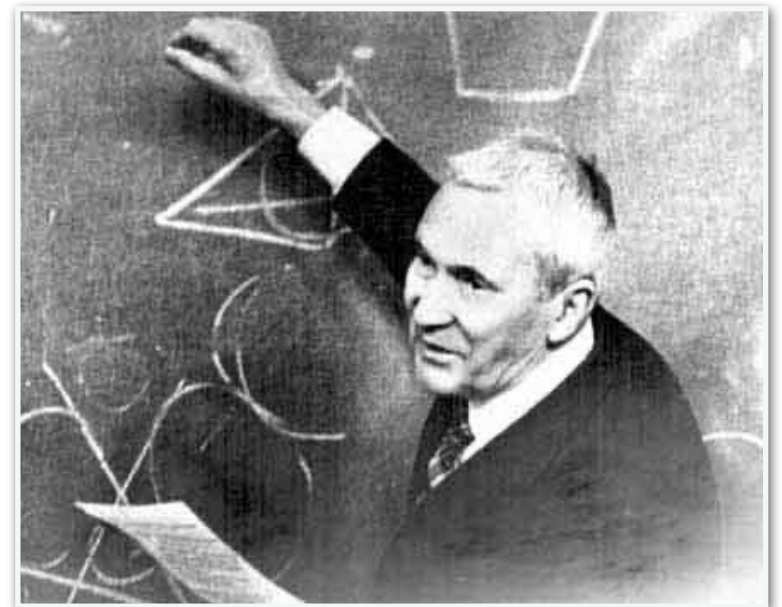
For any **injective** function $\text{Enc} : \{0,1\}^N \xrightarrow{1-1} \{0,1\}^*$, for uniform random $s \in \{0,1\}^N$, for any integer $l > 0$,

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

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

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 $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$ with dependency degree $d < 2^{k-3}$

Moser's Algorithm:

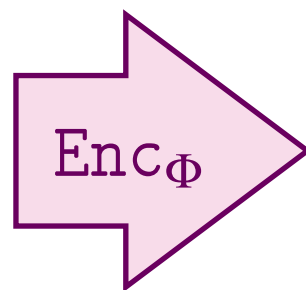
draw uniform $X_1, \dots, X_n \in \{T, F\}$;
while $\exists$ a violated clause C_i :
 Fix(C_i);

Fix(C_i):

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

- **Random bits**: n initial bits + t calls to **Fix**() $\times k$ bits per each call
- t calls to **Fix**() $\implies$ we can compress random bits:

$n + tk$ random bits



$\leq n + O(m \log m) + t(\lceil \log_2(d+1) \rceil + 2)$
bits

Simulate(Φ, t):

Run Moser's Algorithm on k -CNF Φ for up to t calls to Fix();

```
printf("X1...Xn"); //the current X1,...,Xn in Moser's algorithm
```

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;

while $\exists$ a violated clause C_i :

```
printf("("i");
```

```
Fix(Ci);
```

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(Cj);
```

```
printf(")");
```

- Output string:

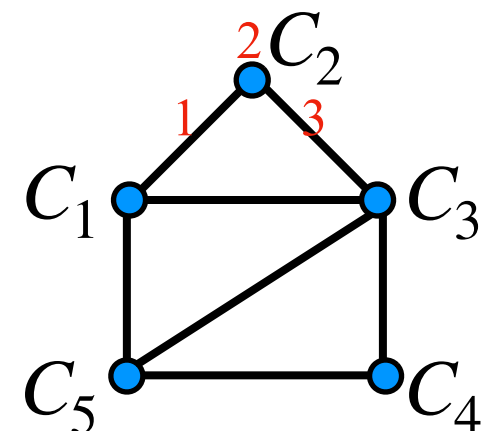
(98(2(1(1)(3))(3))(4(2)))(126(3(2...
.....(110100101010110111

t calls to Fix()

$X_1, \dots, X_n$

all $C_j \in \Gamma^+(C_i)$ are
sorted in an order

$$|\Gamma^+(C_i)| \leq d + 1$$



- Recursion trees + final assignment

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;

while $\exists$ a violated clause C_i :

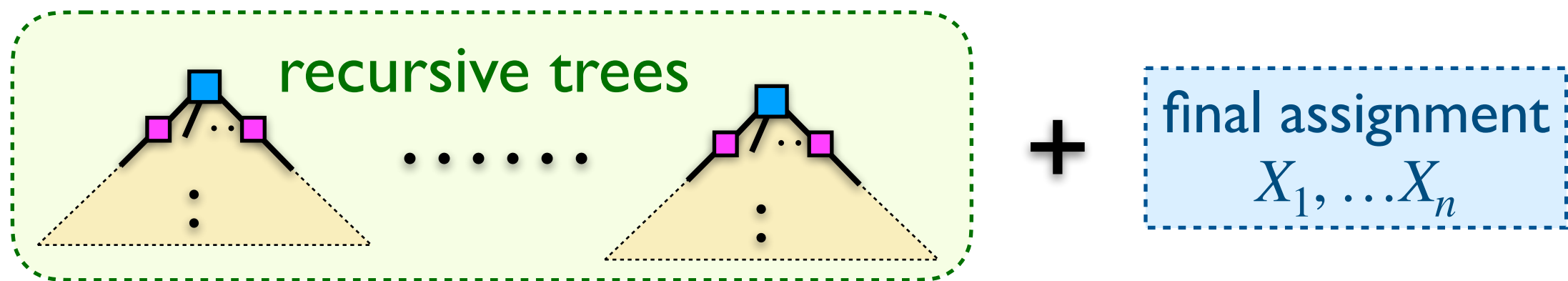
Fix(C_i);

Fix(C_i):

resample all variables in $\text{vbl}(C_i)$;

while $\exists$ violated $C_j \in \Gamma^+(C_i)$:

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...
.....(110100101010110111

t calls to **Fix**()

$X_1, \dots, X_n$



$n + tk$ random bits
used by the algorithm

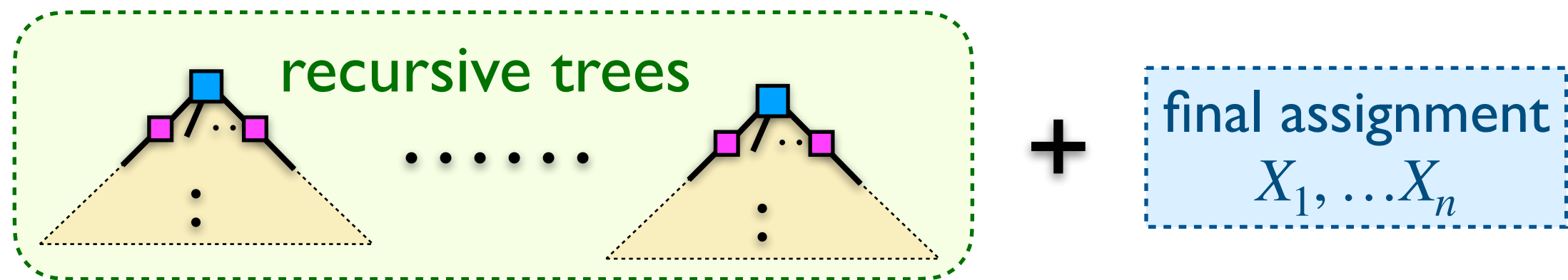
- Recursion trees** + **final assignment**

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;
while $\exists$ a violated clause C_i :
 Fix(C_i);

Fix(C_i):

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



Observation: **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

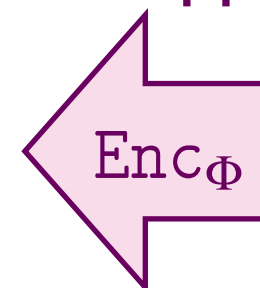
- Output string:**

(98(2(1(1)(3))(3))(4(2)))(126(3(2...
.....(110100101010110111

t calls to **Fix**()

$X_1, \dots, X_n$

1-1 mapping



$n + tk$ random bits
used by the algorithm

- Recursion trees** + **final assignment**

Simulate(Φ, t):

Run Moser's Algorithm on k -CNF Φ for up to t calls to Fix();

printf(" $X_1 \dots X_n$ ");

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;

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:**

- $\leq m \times "(i"$, where $i \in [m]$ at top-level

- $\leq t \times "(r"$, where $r \in [d + 1]$

- $\leq t \times ")"$

- $"(" "$)" form prefix of legal parenthesization

- n bits $X_1, \dots, X_n$ in the end

$\leq n + m(\lceil \log_2 m \rceil + 2)$
 $+ t(\lceil \log_2(d + 1) \rceil + 2)$
bits

Incompressibility Principle:

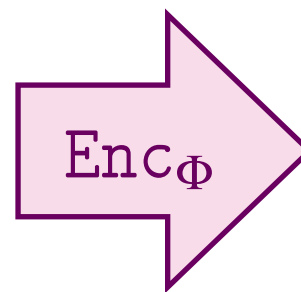
For any **injective** function $\text{Enc} : \{0,1\}^N \xrightarrow{1-1} \{0,1\}^*$, for uniform random $s \in \{0,1\}^N$, for any integer $l > 0$,

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



- Assume $\geq t$ calls made to **Fix()**:

$n + tk$ random bits
used by the algorithm



$$\leq n + m(\lceil \log_2 m \rceil + 2) + t(\lceil \log_2(d+1) \rceil + 2) \text{ bits}$$

$$(\text{LLL condition } d < 2^{k-3}) \leq n + m(\log_2 m + 3) + t(k-1)$$

- 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: $\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_m$
 $\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge \overbrace{(x_3 \vee \neg x_4 \vee \neg x_5)}^{k \text{ variables}}$

Moser's Algorithm:

draw uniform $X_1, \dots, X_n \in \{T, F\}$;
while $\exists$ a violated clause C_i :
 Fix(C_i);

Fix(C_i):

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

Theorem [Moser 2009]:

$d < 2^{k-3} \implies$ 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:

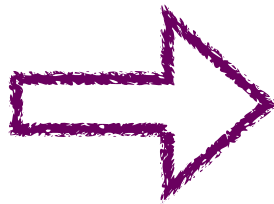
Moser-Tardos Algorithm

Moser's Algorithm and Entropic Proof

Summary

- The Moser-Tardos algorithm:

LLL
condition



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

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

- 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