In this lecture we'll present an algorithmic version of the Lovász Local Lemma (LLL).
This is from Moser '09; we'll do the version from Moser-Tardos '09.
The original LLL is from Erdos-Lovasz '75.

Notation:

"Bad" events $B_1, \dots, B_n$.

For each i, let $D_i \subseteq \{B_1, \dots, B_n\} \setminus \{B_i\}$ denote the dependencies for B_i

& $D_i^+ = D_i \cup \{B_i\}$.

Let $X_1, \dots, X_m$ be the underlying random variables.

For event B_i, let $vbl(B_i) = \{X_j : B_i \text{ depends on } X_j\}$,

If B_i occurs we say B_i is violated.

LLL: If there exists $x_1, \dots, x_n \in [0,1)$ s.t.

for all i, $$\Pr(B_i) \le x_i \prod_{j \in D_i} (1 - x_j)$$

then $$\Pr(\mathcal{G}) = \Pr\left(\bigcap_{i=1} \overline{B_i}\right) > 0.$$

Algorithmic version:

Moreover, we can find a setting of $\{x_1, \dots, x_m\}$ that violates none of the B_i in expected time $\le \sum_{i=1}^{n} x_i/(1-x_i)$.

Here's the algorithm.

1. Choose an initial assignment for $x_1, \dots, x_m$.
2. If some B_i is violated (if multiple B_i's are violated, arbitrarily choose one) then resample $vbl(B_i)$

repeat

Let the execution of the algorithm be denoted as:

$$E := E(1), \ldots, E(T)$$

where $E(t)$ is the event B_i resampled at time t.

We'll define a set of witness trees corresponding to E.
For a tree $\mathcal{T}$, let $V(\mathcal{T})$ denote its vertices &
for $v \in V(\mathcal{T})$, let $Depth(v) = d(v) =$ depth of v
$=$ distance from v to the root of $\mathcal{T}$
where $d(r) = 0$ & its children have depth 1, etc.

For each $t' \in \{1, \ldots, T\}$:
Create a witness tree $\mathcal{T}(t')$ as follows:
- make event $E(t')$ as the root
- for $t = t'-1 \to 1$:
  - add $E(t)$ as a child of the node $E(j)$ in the current tree with largest depth & where $E(t) \in D^+(E(j))$
  - if there is no such $E(j)$ then leave out $E(t)$

Note, in a witness tree,

- all children have distinct labels, and
- an event B_i occurs at most once at each depth.

Why? if adding B_i and it already occurs at depth d, then we can add B_i as a child of that node at depth d (or of a node at higher depth).

<u>Lemma 0:</u> Fix a witness tree $\mathcal{T}$.

$$\Pr(\mathcal{T} \text{ appears in } E) = \prod_{v \in V(\mathcal{T})} \Pr(B_v)$$

where B_v is the event corresponding to node v.

<u>Proof:</u>

Fix a witness tree $\mathcal{T}$.

Order the vertices $V(\mathcal{T})$ so that higher depth are before lower depth, i.e., first the leaves at the highest depth & then work up the tree.

Consider the following algorithm:

Go through $V(\mathcal{T})$ in order.

For vertex v, resample $vbl(B_v)$.

Say that $\hat{T}$ was violated if for all $v \in V(\hat{T})$, the resampling of B_v violated this event B_v.

Note, $Pr(\hat{T} \text{ was violated}) = \prod_{v \in V(\hat{T})} Pr(B_v)$,

Since each B_v only depends on $vbl(B_v)$ & these are resampled at this time.

Now return to the original algorithm & the execution E.
For each variable x_j, imagine an infinite list of resamplings of x_j.

For a vertex $v \in V(\hat{T})$, consider the resampling of $vbl(B_v)$ in the algorithm on $\hat{T}$.
Consider $x_j \in vbl(B_v)$.
Note, x_j does not occur again on the same level of $\hat{T}$.
Thus, let $n_{j,v}$ be the # of ~~& occurrences of~~ resamplings of x_j due to events B_l which occur at depths $>$ depth(v).
Note that in the original algorithm for E, x_j is resampled exactly $n_{j,v}+1$ times prior to B_v
(the $+1$ is for the initial setting of x_j's)

Decide the random choices for the variables $X_1, \dots, X_m$ using the tree algorithm & then use them for the algorithm as well (with -1) so that the first resampling of X_j in the tree gives the initial setting of X_j in E.

In this way, if B_v is violated in $\tilde{T}$ then in E at the corresponding time t the event B_v will be violated prior to this time.

Therefore, $Pr(\tilde{T} \text{ appears in } E) \leq Pr(\tilde{T} \text{ appears in } E)$.

& since $Pr(\tilde{T} \text{ appears in } E) = \prod_{v \in V(\tilde{T})} Pr(B_v)$

this proves the lemma.

For event B_i, let N_i be the # of times that B_i is resampled in the original algorithm E.

Note, the running time of the algorithm is propstional to $\sum_{i=1}^{n} N_i$

And, N_i = # of trees with root B_i in execution E.

To prove the main theorem we need to show:

<u>Lemma 1:</u> $E[N_i] \leq \frac{x_i}{1-x_i}$

Consider the following <u>Galton-Watson</u> tree (this is a random tree):

Fix the root to be B_i.

For node B_i:

for each $B_j \in D_i^+$:

- add B_j as a child of B_i with prob. x_j & leave out with prob. $1-x_j$

Repeat, if B_j is added.

Fix a tree $\tilde{T}$ with root B_i.

Let $p_{\tilde{T}} := \Pr(\text{G-W process produces } \tilde{T})$

<u>Lemma 2:</u> $p_{\tilde{T}} = \frac{1-x_i}{x_i} \prod_{v \in V(\tilde{T})} x'_v$

where $x'_i = x_i \prod_{j \in D_i} (1-x_j)$.

<u>Proof of Lemma 2:</u>

For $v \in V(\tilde{T})$ let $W_v = D^+_{B_v} \setminus N_{\tilde{T}}(v)$

= dependencies of B_v which are not children of v in $\tilde{T}$.

since root is fixed

Then, $p_{\tilde{T}} = \frac{1}{x_i} \prod_{v \in V(\tilde{T})} x_v \prod_{u \in W_v} (1-x_u)$ ← don't include B_u w.p. $1-x_u$

have to add B_v w.p. x_v

$$= \frac{1-x_i}{x_i} \prod_{v \in V(\tilde{T})} \frac{x_v}{1-x_v} \prod_{u \in D^+_v} (1-x_u)$$

$$= \frac{1-x_i}{x_i} \prod_{v \in V(\tilde{T})} x_v \prod_{u \in D_v} (1-x_u)$$

$$= \frac{1-x_i}{x_i} \prod_{v \in V(\tilde{T})} x'_v \quad \blacksquare$$

Now we can prove lemma 1 bounding $E[N_i]$.

Proof of Lemma 1:

$$E[N_i] = \sum_{\tau} \Pr(\tau \text{ appears in } E)$$

$$\leq \sum_{\tau} \prod_{v \in V(\tau)} \Pr(B_v) \quad \text{(by Lemma 0)}$$

$$\leq \sum_{\tau} \prod_{v} x_v' \quad \text{(by the hypothesis of the LLL)}$$

$$= \frac{x_i}{1-x_i} \sum_{\tau} p_{\tau} \quad \text{(by Lemma 2)}$$

$$\leq \frac{x_i}{1-x_i} \quad \text{since } \sum_{\tau} p_{\tau} = 1$$

because the G-W Process Produces 1 tree.

∎

This proves the algorithmic version of LLL.