Fine-Grained Complexity and Algorithm Design Boot Camp

# Lower Bounds Based on SETH

**Dániel Marx**
**(slides by Daniel Lokshtanov)**

**Simons Institute, Berkeley, CA**

**September 4, 2015**

# Tight lower bounds

Have seen that ETH can give tight lower bounds

How tight? ETH «ignores» constants in exponent

How to distinguish $1.85^n$ from $1.0001^n$?

# SAT

**Input:** Formula $\psi$ with m clauses over n boolean variables.

**Question:** Does there exist an assignment to the variables that satisfies all clauses?

Note: Input can have size superpolynomial in n!

Fastest algorithm for SAT: $2^n$poly(m)

# d-SAT

Here all clauses have size $\leq$ d

Input size $\leq n^d$

| | |
|---|---|
| Fastest algorithm for 2-SAT: | n+m |
| Fastest algorithm for 3-SAT: | $1.31^n$ |
| Fastest algorithm for 4-SAT: | $1.47^n$ |
| … | |
| Fastest algorithm for d-SAT: | $2\uparrow(1-c/d)n$ |
| Fastest algorithm for SAT: | $2\uparrow n$ |

# Strong ETH

Let $s_d = \inf\{c : \text{d-SAT has a } 2^{cn} \text{ algorithm}\}$

Let $s_\infty = \lim_{d\to\infty} s_d$

Know: $0 \leq s_d \leq s_\infty \leq 1$

ETH: $s_3 > 0$

SETH: $s_\infty = 1$

# Showing Lower Bounds under SETH

d-SAT $\longrightarrow$ Your Problem

$1.99999{\uparrow}n$

Too fast algorithm?

The number of 9's MUST
be independent of d

# Dominating Set

**Input:** $n$ vertices, integer $k$

**Question:** Is there a set $S$ of at most $k$ vertices such that $N[S] = V(G)$?

Naive: $n^{k+1}$

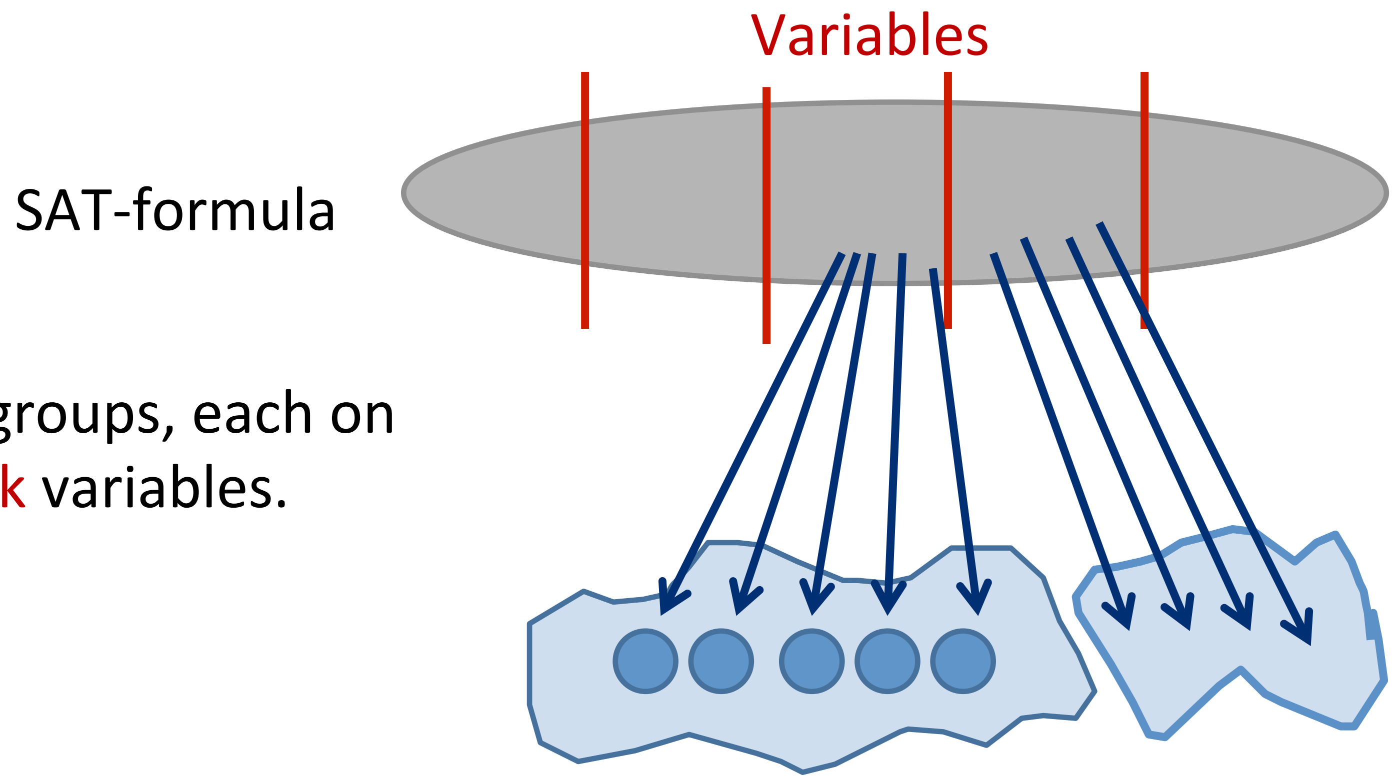
Smarter: $n^{k+o(1)}$

Assuming ETH: no $f(k)n^{o(k)}$

$n^{k/10}$?

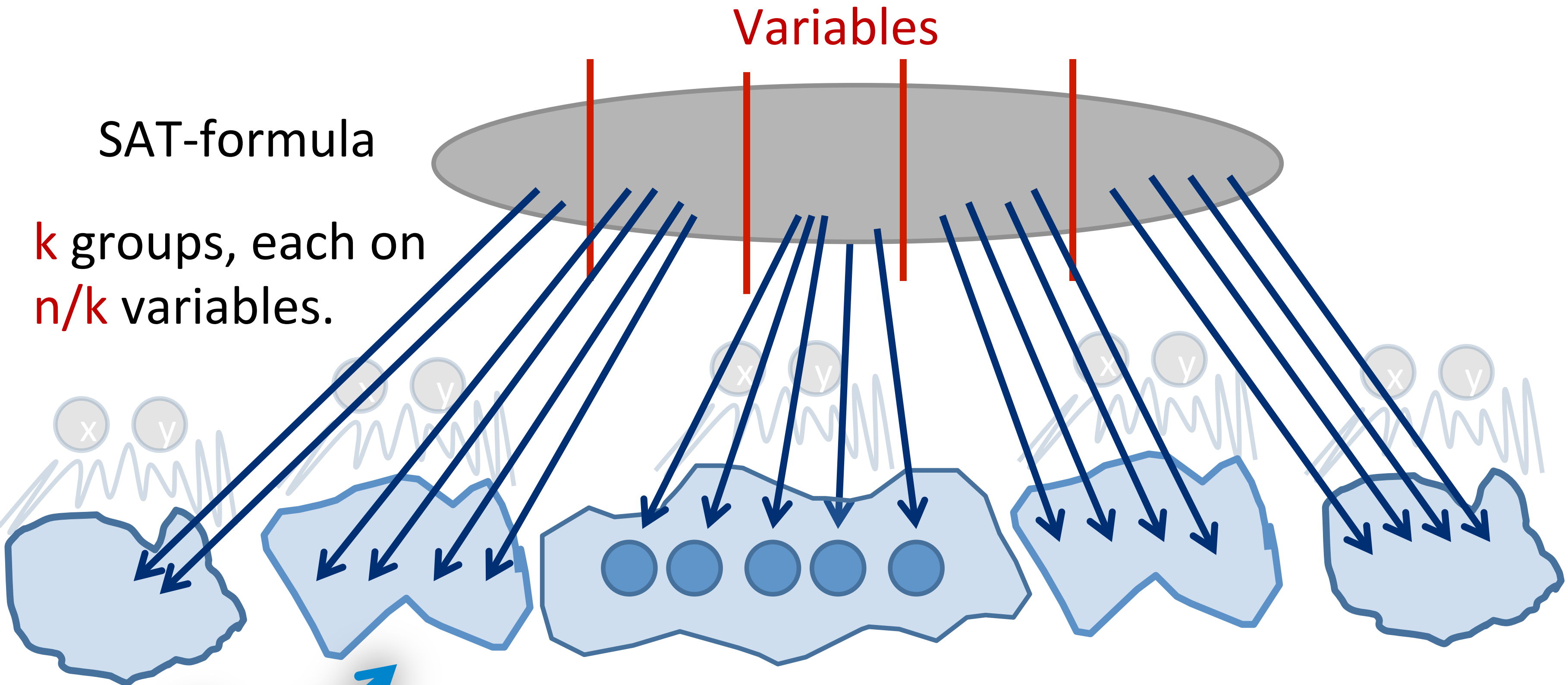
$n^{k-1}$?

# SAT → k-Dominating Set

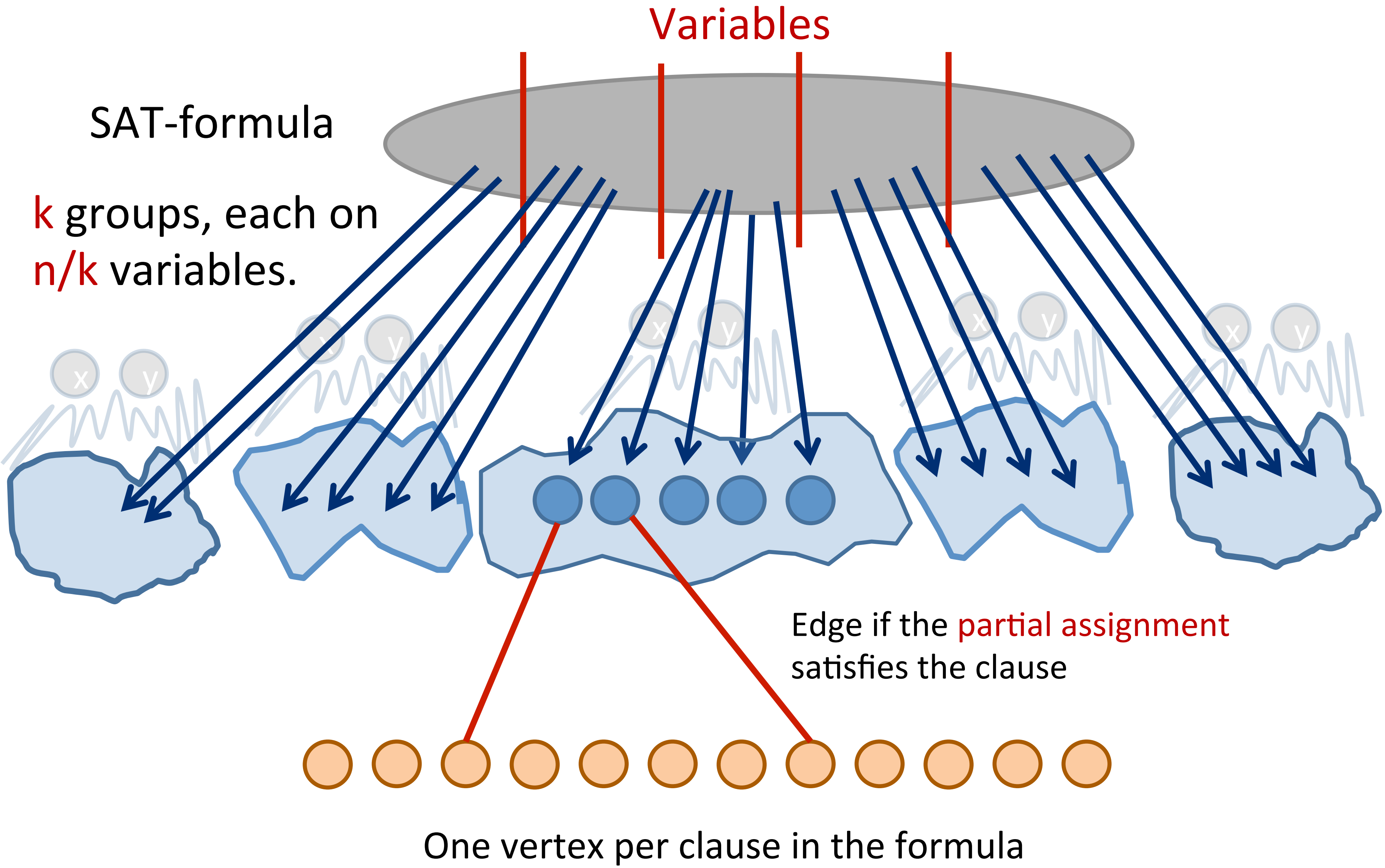



k groups, each on n/k variables.

One vertex for each of the $2^{n/k}$ assignments to the variables in the group.




Selecting one vertex from each cloud corrsponds to selecting an assignment to the variables.


Variables
SAT-formula
k groups, each on
n/k variables.
x
y
x
y
x
y
x
y
x
y
Edge if the partial assignment
satisfies the clause
One vertex per clause in the formula

# SAT → k-Dominating Set

analysis

Too fast algorithm for k-Dominating Set: $n^{k-0.01}$

For any fixed k (like k=3)

If $m \geq 2^{n/k}$ then $2^n$ is at most $m^k$,
which is polynomial!
So $m \leq 2n/k$

The output graph has
$k2^{n/k} + m \leq 2k \cdot 2^{n/k}$ vertices

$$(2k \cdot 2^{n/k})^{k-0.01}$$

$$\leq (2k)^{k} \cdot 2^{nk-0.01/k}$$

$$= O(1.999^{n})$$

# Dominating Set, wrapping up

A $O(n^{2.99})$ algorithm for 3-Dominating Set, or

a $O(n^{3.99})$ algorithm for 4-Dominating Set, or a

a $O(n^{4.99})$ algorithm for 5-Dominating Set, or a …

… would violate SETH.

# Treewidth

- We have seen: $2^t n^{O(1)}$, $3^t n^{O(1)}$, etc. algorithms and no $2^{o(t)} n^{O(1)}$ algorithms assuming ETH.

# Independent Set / Treewidth

**Input:** Graph $G$, integer $k$, tree-decomposition of $G$ of width $\leq t$.

**Question:** Does $G$ have an independent set of size at least $k$?

DP: $O(2^t n)$ time algorithm

Can we do it in $1.99^t \text{poly}(n)$ time?

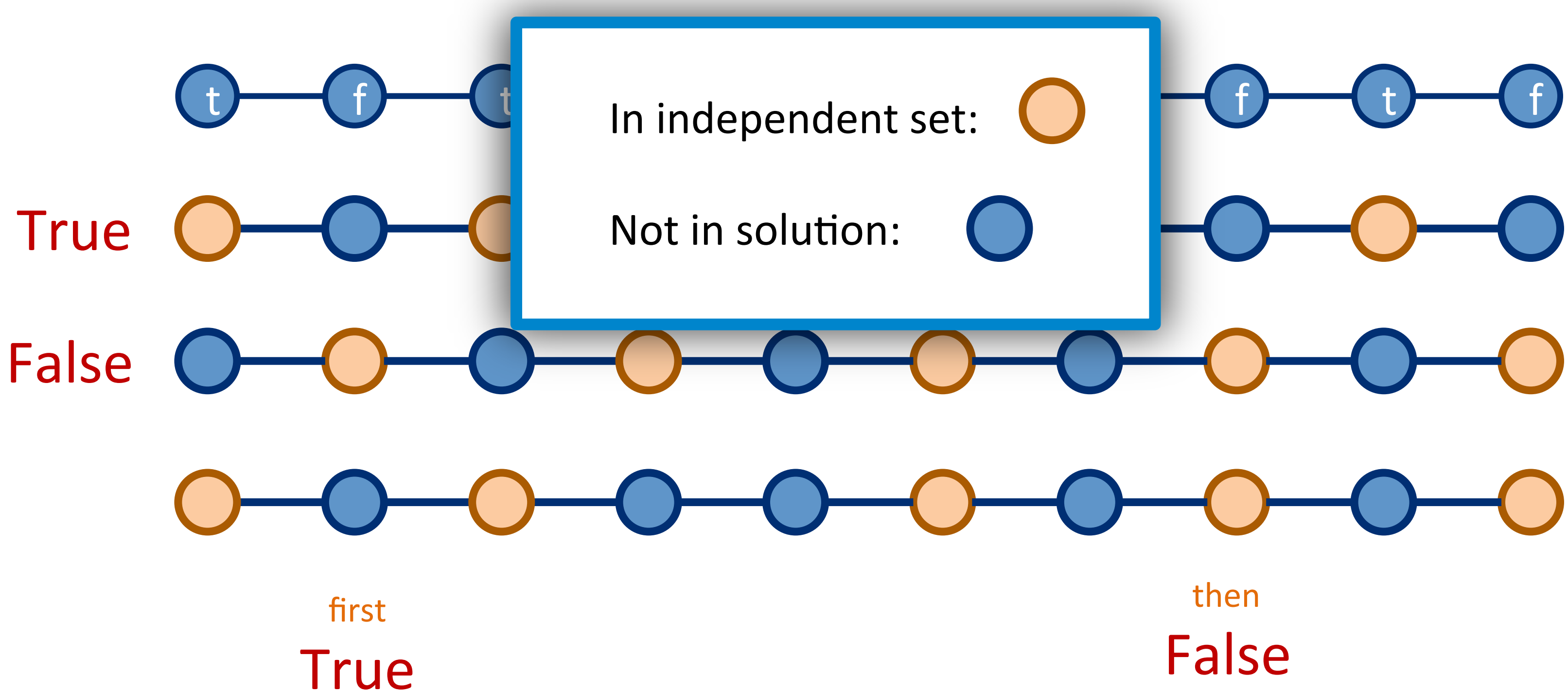
**Next:** If yes, then SETH fails!

# Independent Set / Treewidth

Will reduce $n$-variable $d$-SAT to Independent Set in graphs of treewidth $t$, where $t \leq n+d$.

So a $1.99^t \mathrm{poly}(N)$ algorithm for Independent Set gives a $1.99^{n+d} \mathrm{poly}(n) \leq O(1.999^n)$ time algorithm for $d$-SAT.

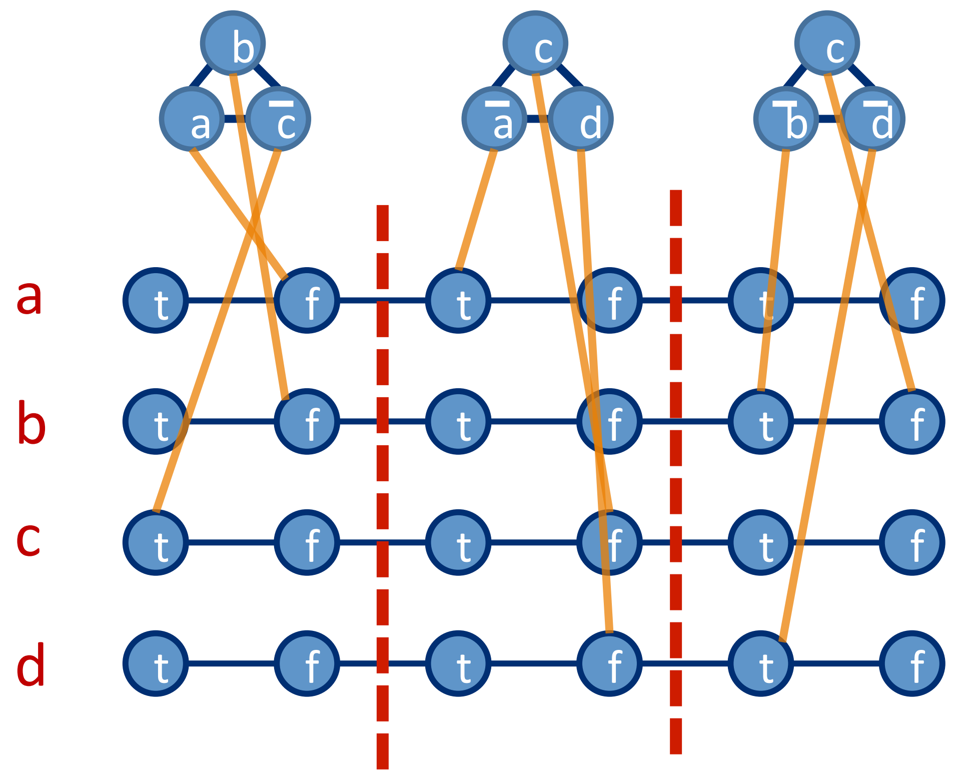
# Independent Sets on an Even Path



# d-SAT ≤ Independent Set

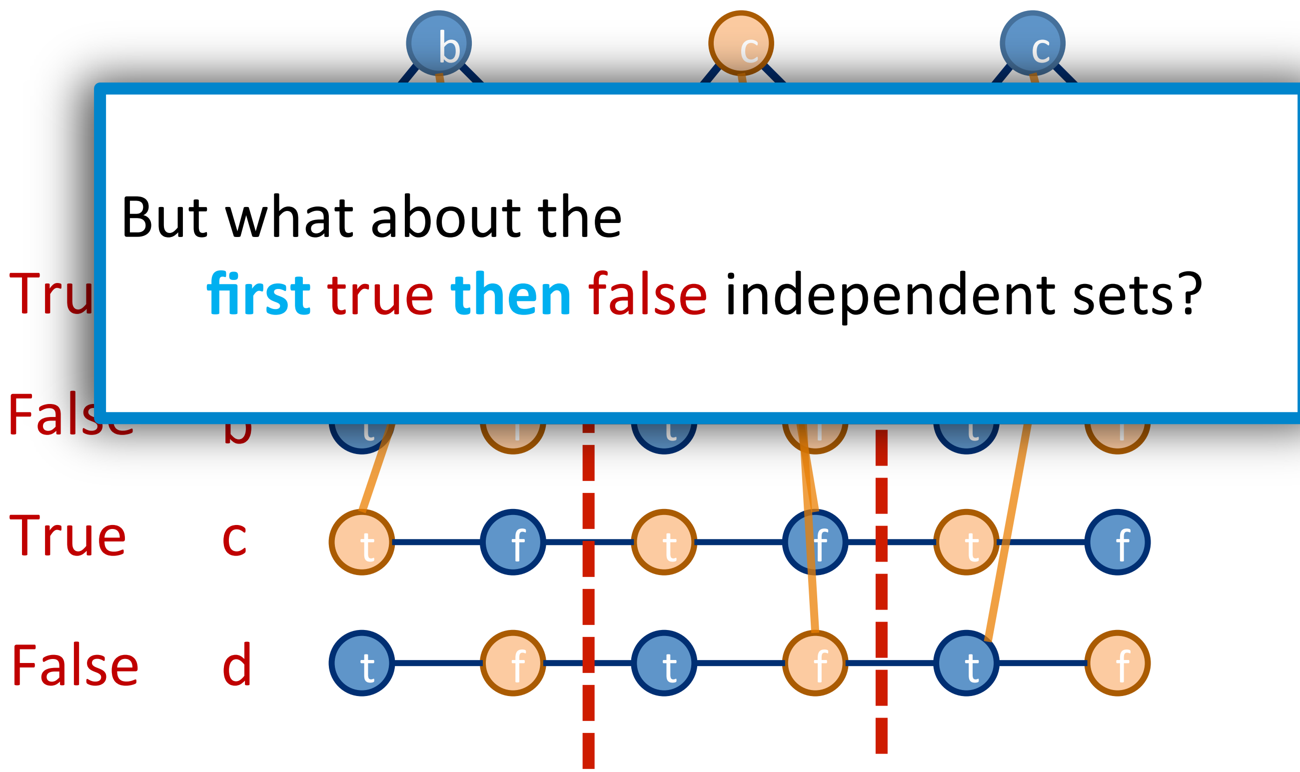
proof by example

$$\psi = (a \lor b \lor \overline{c}) \land (\overline{a} \lor c \lor d) \land (\overline{b} \lor c \lor \overline{d})$$



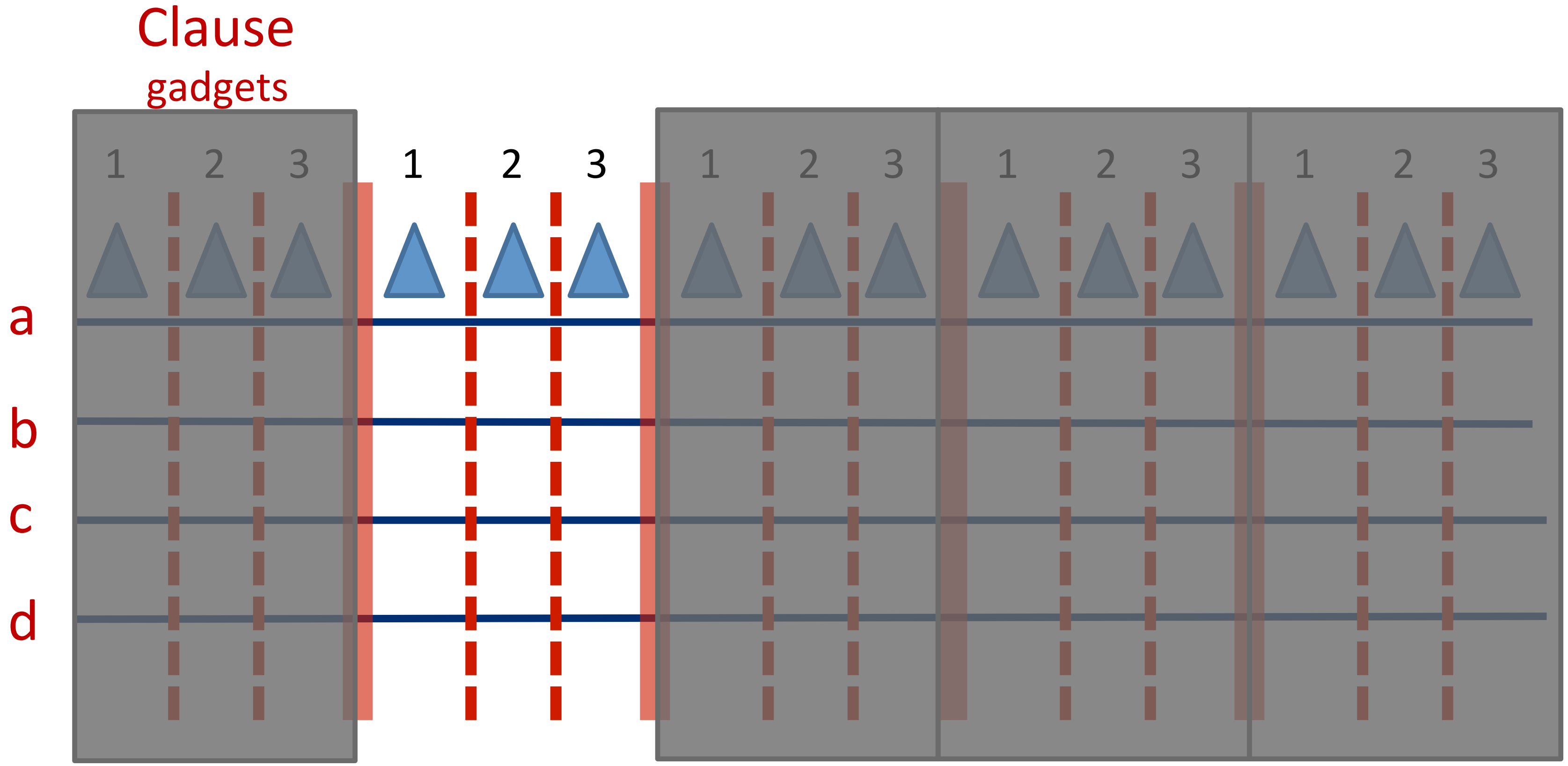
# Independent Sets ↔ Assignments

$$\psi = (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee c \vee d) \wedge (\bar{b} \vee c \vee \bar{d})$$




But what about the
first true then false independent sets?

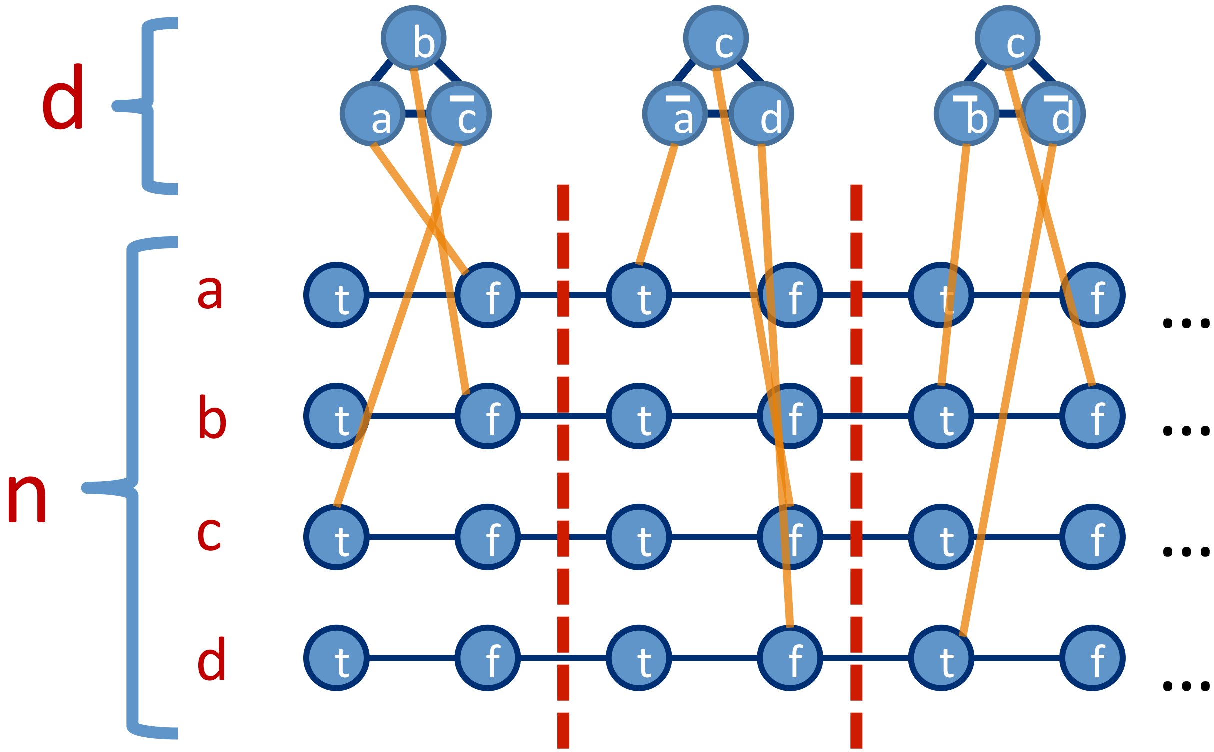
# Dealing with true→false




Every variable flips true→false at most once!

# Treewidth Bound

by picture




Formal proof - exercise

# Independent Set / Treewidth

wrap up

Reduced $n$-variable $d$-SAT to Independent Set in graphs of treewidth $t$, where $t \leq n+d$.

A $1.99^t \mathrm{poly}(N)$ algorithm for Independent Set gives a $1.99^{n+d} \mathrm{poly}(n) \leq O(1.999^n)$ time algorithm for $d$-SAT.

Thus, no $1.99^t$ algorithm for Independent Set assuming SETH

Assuming SETH, the following algorithms are optimal:

- $2^t \cdot \text{poly}(n)$ for Independent Set
- $3^t \cdot \text{poly}(n)$ for Dominating Set
- $c^t \cdot \text{poly}(n)$ for $c$-Coloring
- $3^t \cdot \text{poly}(n)$ for Odd Cycle Transversal
- $2^t \cdot \text{poly}(n)$ for Partition Into Triangles
- $2^t \cdot \text{poly}(n)$ for Max Cut
- $2^t \cdot \text{poly}(n)$ for #Perfect Matching
- …

# $3^t$ lower bound for Dominating Set?

Need to reduce k-SAT formulas on n-variables to Dominating Set in graphs of treewidth t, where

$$3\uparrow t \approx 2\uparrow n$$

So $t \approx n/\log 3 \approx 0.58n$

# Hitting Set / n

**Input:** Family $F = \{S_1,\ldots,S_m\}$ of sets over universe $U = \{v_1, \ldots, v_n\}$, integer $k$.

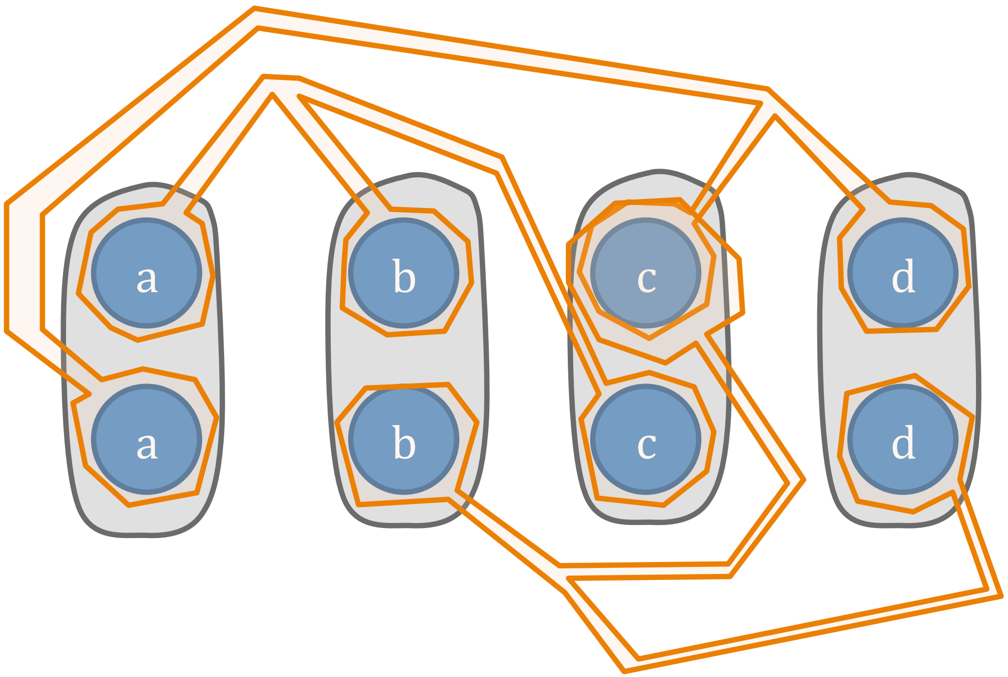
**Question:** Does there exist a set $X \subseteq U$ of size at most $k$ such that for every $S_i \in F$, $S_i \cap X \neq \emptyset$?

Naive algorithm runs in $O(2^n nm)$ time.

**Next:** $1.41^n \text{poly}(n,m)$ implies that SETH fails

# d-SAT $\leq$ Hitting Set

$$\psi = (a \lor b \lor \bar{c}) \land (\bar{a} \lor c \lor d) \land (\bar{b} \lor c \lor \bar{d})$$




Budget = 4

# d-SAT vs Hitting Set

A $c^n$ algorithm for Hitting Set makes a $c^{2n}$ algorithm for d-SAT.

Since $1.41^{2n} < 1.9999^n$, a $1.41^n$ algorithm for Hitting Set violates the SETH.

Have a $2^n$ algorithm and a $1.41^n$ lower bound.

**Next:** $2^n$ lower bound

# Hitting Set

For any fixed $\epsilon > 0$, will reduce k-SAT with n variables to Hitting Set with universe with at most $(1+\epsilon)n$ elements.

So a $1.99^n$ algorithm for Hitting Set gives a $1.99^{n(1+\epsilon)} \leq 1.999^n$ time algorithm for k-SAT

# Some deep math

For every $\epsilon > 0$ there exists a natural number g such that, for t = $\lfloor g(1+\epsilon) \rfloor_{\mathrm{odd}}$ we have:

$$\binom{t}{\lceil t/2 \rceil} \geq 2^{g}$$

Why is this relevant?

# d-SAT ≤ Hitting Set

Group the variables into groups of size g, and set

t = ⌊

Element

Solution budget $\lceil t/2 \rceil$ from each group

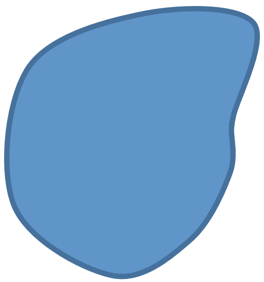
Will force $\geq \lceil t/2 \rceil$ from each group

→ Exactly $\lceil t/2 \rceil$ from each group

Elements $\leq (1+\epsilon) \cdot$ variables

# Analyzing a group

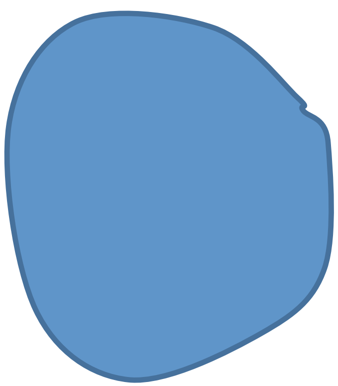
Group of g variables



$2^g$ assignments to variables

Injection

$\binom{t}{\lceil t/2 \rceil}$ subsets of elements of size exactly $\lceil t/2 \rceil$.



Group of t elements

# Forcing solution $\lceil t/2 \rceil$ vertices in a group?

Add all subsets of the group of size $\lceil t/2 \rceil$ to the family F.

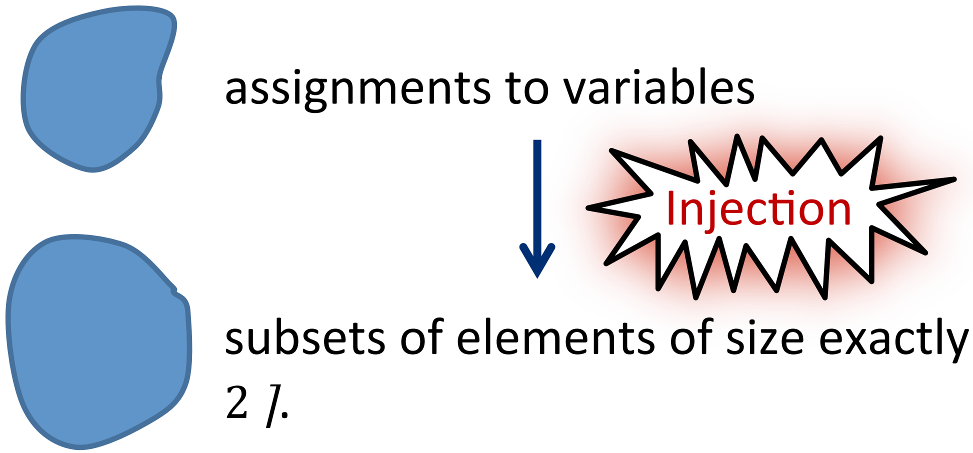
Lets call these sets guards

Any set that picks less than $\lceil t/2 \rceil$ elements the group misses a guard.

Any set that picks at least $\lceil t/2 \rceil$ elements from each group hits all the guards

# Analyzing a group

Group of g variables




Group of t elements

What about the element subsets of size $\lceil t/2 \rceil$ that do not correspond to assignments?

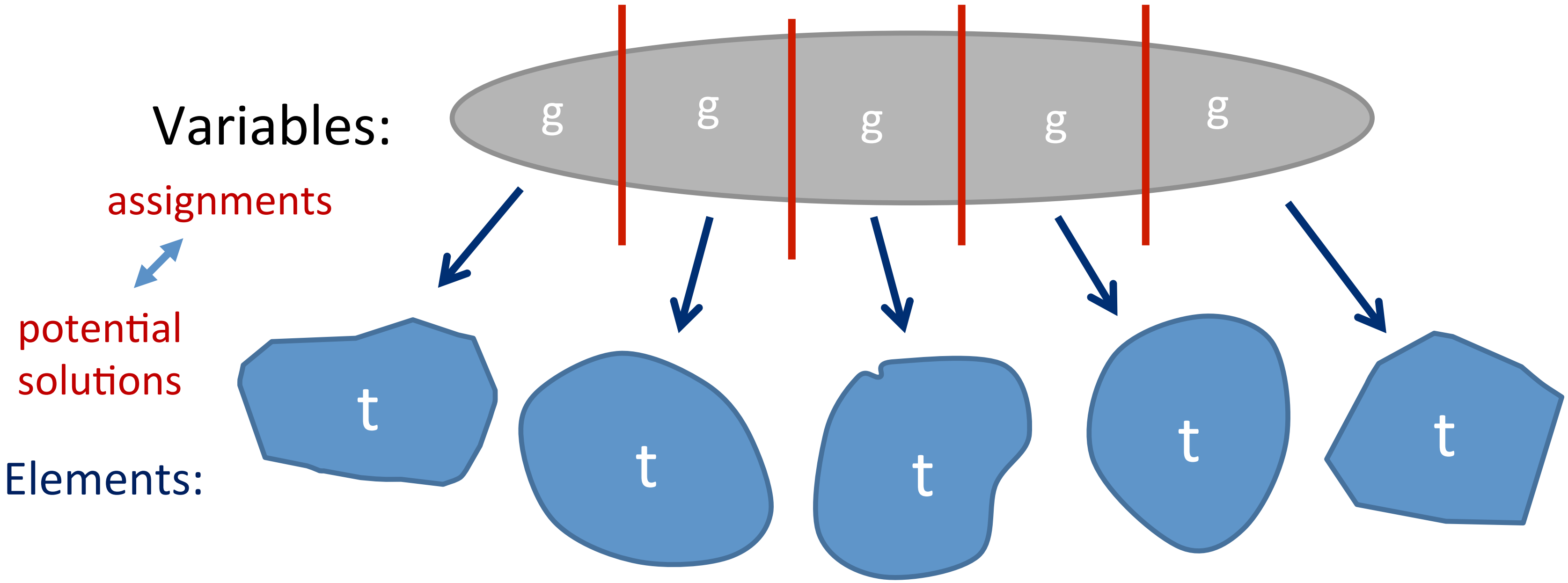
# Sets of size $\lfloor t/2 \rfloor$

Adding a set of size $\lfloor t/2 \rfloor$ to the family F ensures that the «group complement» set is not picked.

All other sets of size $\lceil t/2 \rceil$ in the group may still be picked in solution.

Forbid sets of size $\lceil t/2 \rceil$ that do not correspond to assignments.

# d-SAT ≤ Hitting Set




**Want:**

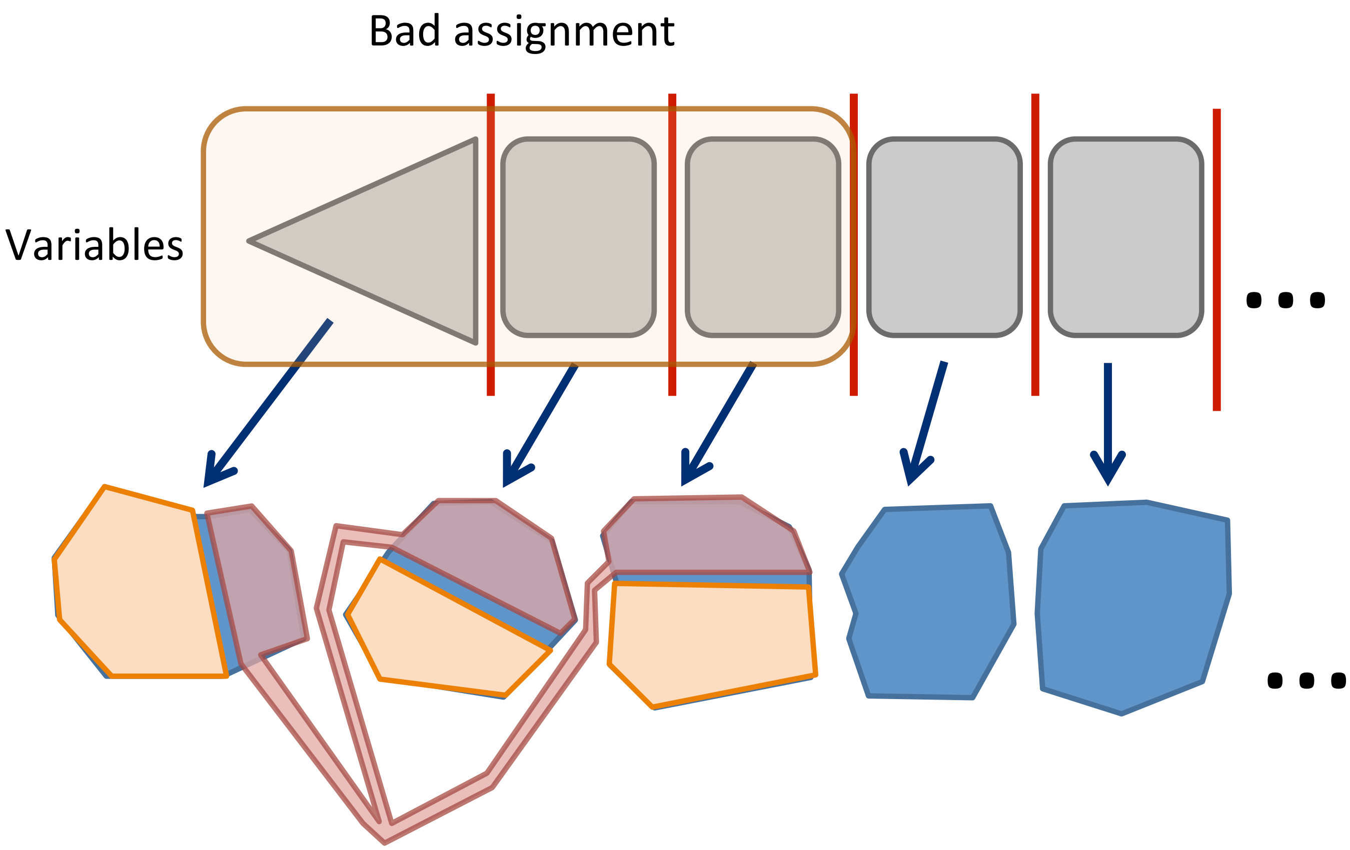
Solutions ↔ <u>Satisfying</u> assignments

# Forbidding partial assignments

Pick any d groups of variables, and consider some assignment to these variables.

If this assignment falsifies $\psi$ we want to forbid the corresponding set in the Hitting Set instance from being selected.

# Forbidding partial assignments




Set added to F to forbid the bad assignment

# Forbidding partial assignments

For each bad assignment to at most d groups, forbid it by adding a «bad assignment guard»

This adds $O(n^d 2^{gd}) = O(n^d)$ sets to F.

# Satisfying Assignments ↔ Hitting Sets

A satisfying assignment has no bad sub-assignments → corresponds to a hitting set.

A hitting set corresponds to an assignment.

If this assignment falsified a clause C, the assignment would be bad for the $\leq$ d groups C lives in, and miss a bad assignment guard.

# Hitting Set wrap up

Can reduce $n$ variable d-SAT to $n(1+\epsilon)$ element Hitting Set.

So a $c^n$ algorithm for Hitting Set yields a $(c+\epsilon)^n$ algorithm for $d$-SAT.

A $1.99^n$ algorithm for Hitting Set would violate SETH.

# Conclusions

SETH can be used to give very tight running time bounds.

SETH recently has been used to give lower bounds for polynomial time solvable problems, and for running time of approximation algorithms.

# Important Open Problems

Can we show a $2^n$ lower bound for Set Cover assuming SETH?

Can we show a 1.00001 lower bound for 3-SAT assuming SETH?