BILL AND NATHAN, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!
NPC SAT-type Problems

Exposition by William Gasarch—U of MD
NPC Problems on Boolean Formulas

Exposition by William Gasarch—U of MD
Bounding

(1) Literals Per Clause
(2) Occurrences of a Var

Exposition by William Gasarch—U of MD
Two Types of SAT

1. **kSAT-b**: Clauses have $\leq k$ literals, each var occurs $\leq b$ times.

2. **EU-kSAT-b**: Clauses have $k$ literals, each var occurs $\leq b$ times.

*Caveat* Do not allow $x$ and $\neg x$ in the same clause.
Two Types of SAT

1. \textbf{kSAT-}b: Clauses have $\leq k$ literals, each var occurs $\leq b$ times.

2. \textbf{EU-}k\textbf{SAT-}b: Clauses have $k$ literals, each var occurs $\leq b$ times.

\textbf{Caveat} Do not allow $x$ and $\neg x$ in same clause.
Two Types of SAT

1. **kSAT-\(b\):** Clauses have \(\leq k\) literals, each var occurs \(\leq b\) times.

2. **EU-kSAT-\(b\):** Clauses have \(k\) literals, each var occurs \(\leq b\) times.

**Caveat** Do not allow \(x\) and \(\neg x\) in same clause.

**Caveat** Do not allow \(x\) and \(\neg x\) in same clause.
Two Types of SAT

1. **kSAT-b**: Clauses have $\leq k$ literals, each var occurs $\leq b$ times.

2. **EU-kSAT-b**: Clauses have $k$ literals, each var occurs $\leq b$ times.

**Caveat** Do not allow $x$ and $\neg x$ in same clause.

**Caveat** Do not allow $x$ and $x$ in same clause.

**Occur** $(x \lor y) \land (\neg x \lor z)$: $x$ occurs TWICE.
Two Types of SAT

1. **$k\text{SAT-}b$**: Clauses have $\leq k$ literals, each var occurs $\leq b$ times.
2. **EU-$k\text{SAT-}b$**: Clauses have $k$ literals, each var occurs $\leq b$ times.

**Caveat** Do not allow $x$ and $\neg x$ in same clause.
**Caveat** Do not allow $x$ and $x$ in same clause.
**Occur** $(x \lor y) \land (\neg x \lor z)$: $x$ occurs TWICE.

SAT means no bound on number of literals-per-clause.
We will look at all four of these for various values of $k, b$. 
1. 1SAT:

No Bound on $b$
No Bound on $b$

1. 1SAT: P,
   \[ \phi \in 1\text{SAT} \text{ iff there is no } x \text{ such that both } x \text{ and } \neg x \text{ occur.} \]

2. 2SAT:

3. 3SAT: NPC by Cook.
   The $k=1$ and $k=2$ cases are of course still in P if you bound $b$.
   Hence we look at $k=3$ and bound on $b$. 
No Bound on $b$

1. **1SAT:** P, 
   \[ \phi \in 1\text{SAT} \text{ iff there is no } x \text{ such that both } x \text{ and } \neg x \text{ occur.} \]

2. **2SAT:** P. Known result. Sketch: Convert every clause \[ L_1 \lor L_2 \] into \[ (\neg L_1 \rightarrow L_2) \land (\neg L_2 \rightarrow L_1) \]. Make a directed graph with literals as vertices and the \( \rightarrow \) as edges. \( \phi \in 2\text{SAT} \text{ iff there is no path from an } x \text{ to a } \neg x. \)

3. **3SAT:** NPC by Cook.
   
The \( k = 1 \) and \( k = 2 \) cases are of course still in P if you bound \( b \).
1. **1SAT**: P, 
   \( \phi \in \text{1SAT} \) iff there is no \( x \) such that both \( x \) and \( \neg x \) occur.

2. **2SAT**: P. Known result. Sketch: Convert every clause \( L_1 \lor L_2 \) into \((\neg L_1 \rightarrow L_2) \land (\neg L_2 \rightarrow L_1)\). Make a directed graph with literals as vertices and the \( \rightarrow \) as edges. \( \phi \in \text{2SAT} \) iff there is no path from an \( x \) to a \( \neg x \).

3. **3SAT**: NPC by Cook.

   The \( k = 1 \) and \( k = 2 \) cases are of course still in \( P \) if you bound \( b \). Hence we look at \( k = 3 \) and bound on \( b \).
$k = 3$ and $b = 1, 2$

3SAT-1:
$k = 3$ and $b = 1, 2$

3SAT-1: P. Always satisfiable, just set all literals that appear to T. EU version would still be in P.
$k = 3$ and $b = 1, 2$

3SAT-1: P. Always satisfiable, just set all literals that appear to T. EU version would still be in P.

3SAT, all vars occur $\leq 2$. P

1) Input $\phi$ in 3CNF, all vars occurs $\leq 2$. 
3SAT, all vars occur $\leq 2$. P

1) Input $\phi$ in 3CNF, all vars occurs $\leq 2$.
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
These operations may solve problem.
3SAT, all vars occur $\leq 2$. P

1) Input $\phi$ in 3CNF, all vars occurs $\leq 2$.
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
3SAT, all vars occur \( \leq 2 \). P

1) Input \( \phi \) in 3CNF, all vars occurs \( \leq 2 \).
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
   These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
4) A clause with all NEG literals we call a NEG-clause.
1) Input $\phi$ in 3CNF, all vars occurs $\leq 2$.
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
   These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
4) A clause with all NEG literals we call a NEG-clause.
   If no NEG-clauses then SAT easily.
3SAT, all vars occur ≤ 2. P

1) Input $\phi$ in 3CNF, all vars occurs ≤ 2.
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
4) A clause with all NEG literals we call a NEG-clause.
If no NEG-clauses then SAT easily.
IF there is a NEG-clause then set a var in it to F.
3SAT, all vars occur $\leq 2$. 

1) Input $\phi$ in 3CNF, all vars occurs $\leq 2$.
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
   These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
4) A clause with all NEG literals we call a NEG-clause.
   If no NEG-clauses then SAT easily.
   IF there is a NEG-clause then set a var in it to F.
   (Numb NEG-clauses) + (Numb of clauses) DECREASES.
3SAT, all vars occur \( \leq 2 \). P

1) Input \( \phi \) in 3CNF, all vars occurs \( \leq 2 \).

2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
   These operations may solve problem.

3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.

4) A clause with all NEG literals we call a NEG-clause.
   If no NEG-clauses then SAT easily.
   IF there is a NEG-clause then set a var in it to F.
   (Numb NEG-clauses) + (Numb of clauses) DECREASES.
   Eventually satisfy all clauses.
3SAT, all vars occur \( \leq 2 \). P

1) Input \( \phi \) in 3CNF, all vars occurs \( \leq 2 \).
2) If a literal is only pos, set T, if only neg, set F. If clause has 1 literal, set true.
   These operations may solve problem.
3) Every clause has 2 or 3 literals, every literal occurs as pos and neg. We show SAT.
4) A clause with all NEG literals we call a NEG-clause.
   If no NEG-clauses then SAT easily.
   IF there is a NEG-clause then set a var in it to F.
   \((\text{Numb NEG-clauses}) + (\text{Numb of clauses})\) DECREASES.
   Eventually satisfy all clauses.

Moral This was a clever trick! To prove \( P \neq NP \) would need to show that no clever trick will get SAT into P. Hard!
3SAT, all vars occur $\leq 3$

3SAT-3: There are $\leq 3$ clauses per literal and every var occurs $\leq 3$ times.
3SAT, all vars occur $\leq 3$

3SAT-3: There are $\leq 3$ clauses per literal and every var occurs $\leq 3$ times.
In $P$? NPC? Breakout Rooms!
3SAT, all vars occur $\leq 3$. NPC

We will prove this NPC. Erika- how will we do it?
3SAT, all vars occur $\leq 3$. NPC

We will prove this NPC. Erika- how will we do it? By a Reduction
1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$
times such that $\phi \in \text{SAT}$ iff $\phi' \in \text{SAT}$. 
We will prove this NPC. Erika- how will we do it? By a Reduction
1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$ times such that $\phi \in$ SAT iff $\phi' \in$ SAT.
2) If a var occurs $\leq 3$ times then leave it alone.
3SAT, all vars occur $\leq 3$. NPC

We will prove this NPC. Erika- how will we do it? By a Reduction

1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$ times such that $\phi \in \text{SAT}$ iff $\phi' \in \text{SAT}$.

2) If a var occurs $\leq 3$ times then leave it alone.

3) If a var occurs $m \geq 4$ times then
We will prove this \textbf{NPC}. Erika- how will we do it? By a Reduction

1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$ times such that $\phi \in \text{SAT}$ iff $\phi' \in \text{SAT}$.

2) If a var occurs $\leq 3$ times then leave it alone.

3) If a var occurs $m \geq 4$ times then

a) Add new vars $x_1, \ldots, x_m$. Replace $i$th occurrence of $x$ with $x_i$. 

\textit{Moral}

Going from $b \leq 2$ to $b \leq 3$ matters!
3SAT, all vars occur $\leq 3$. NPC

We will prove this NPC. Erika- how will we do it? By a Reduction
1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$
times such that $\phi \in SAT$ iff $\phi' \in SAT$.
2) If a var occurs $\leq 3$ times then leave it alone.
3) If a var occurs $m \geq 4$ times then
   a) Add new vars $x_1, \ldots, x_m$. Replace $i$th occurrence of $x$ with $x_i$.
   b) Add the clauses $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, $\ldots$, $x_{m-1} \rightarrow x_m$, $x_m \rightarrow x_1$.
   (Formally $x_1 \rightarrow x_2$ is $(\neg x_1 \lor x_2$.)

Clearly $\phi \in 3CNF$ and all variables occur $\leq 3$ times.
Clearly $\phi \in SAT$ iff $\phi' \in SAT$.
Moral Going from $b \leq 2$ to $b \leq 3$ matters!
3SAT, all vars occur $\leq 3$. NPC

We will prove this NPC. Erika- how will we do it? By a Reduction

1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$ times such that $\phi \in$ SAT iff $\phi' \in$ SAT.
2) If a var occurs $\leq 3$ times then leave it alone.
3) If a var occurs $m \geq 4$ times then
   a) Add new vars $x_1, \ldots, x_m$. Replace $i$th occurrence of $x$ with $x_i$.
   b) Add the clauses $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, $\ldots$, $x_{m-1} \rightarrow x_m$, $x_m \rightarrow x_1$.
   (Formally $x_1 \rightarrow x_2$ is $(\neg x_1 \lor x_2$.)

Clearly $\phi \in$ 3CNF and all variables occur $\leq 3$ times.
We will prove this NPC. Erika- how will we do it? By a Reduction
1) Input $\phi$ in 3CNF. Want $\phi'$ 3CNF with all vars occurring $\leq 3$
times such that $\phi \in \text{SAT}$ iff $\phi' \in \text{SAT}$.
2) If a var occurs $\leq 3$ times then leave it alone.
3) If a var occurs $m \geq 4$ times then
   a) Add new vars $x_1, \ldots, x_m$. Replace $i$th occurrence of $x$ with $x_i$.
   b) Add the clauses $x_1 \rightarrow x_2$, $x_2 \rightarrow x_3$, $\ldots$, $x_{m-1} \rightarrow x_m$, $x_m \rightarrow x_1$.
   (Formally $x_1 \rightarrow x_2$ is $(\neg x_1 \lor x_2$.)
Clearly $\phi \in 3\text{CNF}$ and all variables occur $\leq 3$ times.
Clearly $\phi \in \text{SAT}$ iff $\phi' \in \text{SAT}$.
3SAT, all vars occur \( \leq 3 \). NPC

We will prove this NPC. Erika- how will we do it? By a Reduction

1) Input \( \phi \) in 3CNF. Want \( \phi' \) 3CNF with all vars occurring \( \leq 3 \) times such that \( \phi \in \text{SAT} \) iff \( \phi' \in \text{SAT} \).

2) If a var occurs \( \leq 3 \) times then leave it alone.

3) If a var occurs \( m \geq 4 \) times then
   a) Add new vars \( x_1, \ldots, x_m \). Replace \( i^{\text{th}} \) occurrence of \( x \) with \( x_i \).
   b) Add the clauses \( x_1 \rightarrow x_2, x_2 \rightarrow x_3, \ldots, x_{m-1} \rightarrow x_m, x_m \rightarrow x_1 \).
   (Formally \( x_1 \rightarrow x_2 \) is \( (\neg x_1 \lor x_2) \).)

Clearly \( \phi \in 3\text{CNF} \) and all variables occur \( \leq 3 \) times.

Clearly \( \phi \in \text{SAT} \) iff \( \phi' \in \text{SAT} \)

**Moral** Going from \( b \leq 2 \) to \( b \leq 3 \) matters!
EU-3SAT-3: Every clause has exactly 3 literals. Every variable occurs \( \leq 3 \) times. P? NPC?
EU-3SAT-3?: Every clause has exactly 3 literals. Every variable occurs \( \leq 3 \) times. P? NPC?
Go to breakout rooms to work on this.
EU-3SAT-3 is in $P$

EU-3SAT-3 with $b \leq 3$ is in $P$. 
EU-3SAT-3 is in $P$

EU-3SAT-3 with $b \leq 3$ is in $P$.
This needs a known Theorem and its Corollary.
For this slide $G = (A, B, E)$ is a bipartite graph.
A **Matching of $A$ into $B$** is a set of disjoint edges so that every element of $A$ is an endpoint of some edge. View as an injection of $A$ into $B$.
$X \subseteq A$. $E(X) = \{ y \in Y : (\exists x \in X)[(x, y) \in E] \}$.

**Hall’s Matching Theorem**
If, for all $X \subseteq A$, $|E(X)| \geq |X|$ then there exists a matching from $A$ to $B$.

**Corollary**
If there exists $k$ such that (1) for every $x \in A$, $\deg(x) \geq k$, and (2) for every $y \in B$, $\deg(y) \leq k$, then there is a matching from $A$ to $B$.
We will use these on the next slide.
EU-3SAT-3 is in $P$

EU-3SAT-3 with $b \leq 3$ is in $P$.
This needs a known Theorem and its Corollary.
For this slide $G = (A, B, E)$ is a bipartite graph.
A **Matching of $A$ into $B$** is a set of disjoint edges so that every element of $A$ is an endpoint of some edge. View as an injection of $A$ into $B$.
$X \subseteq A$. $E(X) = \{y \in Y : (\exists x \in X)[(x, y) \in E]\}$.

**Hall’s Matching Theorem** If, for all $X \subseteq A$, $|E(X)| \geq |X|$ then there exists a matching from $A$ to $B$. 
EU-3SAT-3 is in \( P \)

EU-3SAT-3 with \( b \leq 3 \) is in \( P \).
This needs a known Theorem and its Corollary.
For this slide \( G = (A, B, E) \) is a bipartite graph.
A **Matching of \( A \) into \( B \)** is a set of disjoint edges so that every element of \( A \) is an endpoint of some edge. View as an injection of \( A \) into \( B \).
\( X \subseteq A. \ E(X) = \{y \in Y : (\exists x \in X)[(x, y) \in E]\}\).

**Hall’s Matching Theorem** If, for all \( X \subseteq A, |E(X)| \geq |X| \) then there exists a matching from \( A \) to \( B \).

**Corollary** If there exists \( k \) such that (1) for every \( x \in A, \ \deg(x) \geq k \), and (2) for every \( y \in B, \deg(y) \leq k \), then there is a matching from \( A \) to \( B \).
EU-3SAT-3 is in $P$

EU-3SAT-3 with $b \leq 3$ is in $P$.
This needs a known Theorem and its Corollary.
For this slide $G = (A, B, E)$ is a bipartite graph.
A **Matching of $A$ into $B$** is a set of disjoint edges so that every element of $A$ is an endpoint of some edge. View as an injection of $A$ into $B$.

$X \subseteq A$. $E(X) = \{y \in Y : (\exists x \in X)[(x, y) \in E]\}$.

**Hall’s Matching Theorem** If, for all $X \subseteq A$, $|E(X)| \geq |X|$ then there exists a matching from $A$ to $B$.

**Corollary** If there exists $k$ such that (1) for every $x \in A$, $\deg(x) \geq k$, and (2) for every $y \in B$, $\deg(y) \leq k$, then there is a matching from $A$ to $B$.

We will use these on the next slide.
Every EU-3CNF-3 fml is Satisfiable

Let $\phi$ be EU-3CNF-3. $\phi = C_1 \lor \cdots \lor C_m$.

Form a bipartite graph:

1. Clauses on the left, variables on the right.
2. Edge from $C$ to $x$ if either $x$ or $\neg x$ is in $C$.

Every clause has degree 3.
Every **EU-3CNF-3** fml is Satisfiable

Let $\phi$ be **EU-3CNF-3**. $\phi = C_1 \lor \cdots \lor C_m$.

Form a bipartite graph:

1. Clauses on the left, variables on the right.
2. Edge from $C$ to $x$ if either $x$ or $\neg x$ is in $C$.

Every clause has degree 3. Every variable has degree $\leq 3$.

By Corollary there is a matching of $C$’s to $V$’s. This gives a satisfying assignment.
Every EU-3CNF-3 fml is Satisfiable

Let $\phi$ be EU-3CNF-3. $\phi = C_1 \lor \cdots \lor C_m$.

Form a bipartite graph:

1. Clauses on the left, variables on the right.
2. Edge from $C$ to $x$ if either $x$ or $\neg x$ is in $C$.

Every clause has degree 3. Every variable has degree $\leq 3$.

By Corollary there is a matching of $C$’s to $V$’s. This gives a satisfying assignment.

**Moral** The algorithm used a THEOREM in math that perhaps you did not know. To prove $P \neq NP$ would need to say this can’t happen. Hard!
A Variant of SAT

Exposition by William Gasarch—U of MD
1-in-3-SAT

**Def 1-in-3-SAT** (1-in-3-SAT) is the problem of, given a formula $D_1 \land \cdots \land D_m$ find an assignment that satisfies *exactly* one literal-per-clause. We will show that 1-in-3-SAT is NPC.
1-in-3-SAT

**Def 1-in-3-SAT** (1-in-3-SAT) is the problem of, given a formula $D_1 \land \cdots \land D_m$ find an assignment that satisfies exactly one literal-per-clause. We will show that 1-in-3-SAT is NPC.

*Is this a Natural Question?* VOTE, though this is an opinion question.
Def 1-in-3-SAT is the problem of, given a formula $D_1 \land \cdots \land D_m$ find an assignment that satisfies exactly one literal-per-clause. We will show that 1-in-3-SAT is NPC.

Is this a Natural Question? VOTE, though this is an opinion question.

My Opinion The problem is not natural.
Def **1-in-3-SAT** (1-in-3-SAT) is the problem of, given a formula $D_1 \land \cdots \land D_m$ find an assignment that satisfies *exactly* one literal-per-clause. We will show that 1-in-3-SAT is NPC.

**Is this a Natural Question?** VOTE, though this is an opinion question.

**My Opinion** The problem is *not* natural.

**So why are we studying it** Discuss.
Def 1-in-3-SAT (1-in-3-SAT) is the problem of, given a formula $D_1 \land \cdots \land D_m$ find an assignment that satisfies exactly one literal-per-clause. We will show that 1-in-3-SAT is NPC.

Is this a Natural Question? VOTE, though this is an opinion question.

My Opinion The problem is not natural.

So why are we studying it Discuss.

Its a means to an end We will show natural problems NPC by using reductions from 1-in-3-SAT. We will do a reduction from a variant of 1-in-3-SAT.
1-in-3-SAT is NPC

Given $\phi = C_1 \land \cdots \land C_m$ in 3CNF create the $\phi'$ as follows:
1-in-3-SAT is NPC

Given $\phi = C_1 \land \cdots \land C_m$ in 3CNF create the $\phi'$ as follows:
Replace clause $(L_1 \lor L_2 \lor L_3)$ with

$$(\neg L_1 \lor a \lor b) \land (b \lor L_2 \lor c) \land (c \lor d \lor \neg L_3).$$

where $a, b, c, d$ are new variables.
Given $\phi = C_1 \land \cdots \land C_m$ in 3CNF create the $\phi'$ as follows:
Replace clause $(L_1 \lor L_2 \lor L_3)$ with

$$(\neg L_1 \lor a \lor b) \land (b \lor L_2 \lor c) \land (c \lor d \lor \neg L_3).$$

where $a, b, c, d$ are new variables.
Leave it to the reader to prove

$$\phi \in 3SAT \text{ iff } \phi' \in 1\text{-in-}3\text{-SAT}.$$
Mono 1-in-3-SAT

**Mono 1-in-3-SAT (mono-1-in-3-SAT):** Given a formula $E_1 \land \cdots \land E_p$ where all vars occur positively, is there an assignment that satisfies exactly one literal-per-clause.
Mono 1-in-3-SAT

**Mono 1-in-3-SAT (mono-1-in-3-SAT):** Given a formula $E_1 \land \cdots \land E_p$ where all vars occur positively, is there an assignment that satisfies exactly one literal-per-clause.

**Thm** $1$-in-$3$-SAT $\leq$ mono-$1$-in-$3$-SAT

Given $3$CNF form $\phi(x_1, \ldots, x_n) = C_1 \lor \cdots \lor C_k$ want $\phi'$ such that $\phi \in 1$-in-$3$-SAT iff $\phi' \in$ mono-$1$-in-$3$-SAT.
Mono 1-in-3-SAT

**Mono 1-in-3-SAT (mono-1-in-3-SAT):** Given a formula \( E_1 \land \cdots \land E_p \) where all vars occur positively, is there an assignment that satisfies **exactly** one literal-per-clause.

**Thm** 1-in-3-SAT \( \leq \) mono-1-in-3-SAT

Given 3CNF form \( \phi(x_1, \ldots, x_n) = C_1 \lor \cdots \lor C_k \) want \( \phi' \) such that \( \phi \in \text{1-in-3-SAT} \) iff \( \phi' \in \text{mono-1-in-3-SAT} \).

1) New Vars \( t, f \) and new clause \( E = (t \lor f \lor f) \). Any 1-in-3-SAT assignment of \( \phi \) will set \( t \) to \( T \) and \( f \) to \( F \).
Mono 1-in-3-SAT

**Mono 1-in-3-SAT (mono-1-in-3-SAT):** Given a formula $E_1 \land \cdots \land E_p$ where all vars occur positively, is there an assignment that satisfies **exactly** one literal-per-clause.

**Thm** 1-in-3-SAT $\leq$ mono-1-in-3-SAT

Given 3CNF form $\phi(x_1, \ldots, x_n) = C_1 \lor \cdots \lor C_k$ want $\phi'$ such that $\phi \in$ 1-in-3-SAT iff $\phi' \in$ mono-1-in-3-SAT.

1) New Vars $t, f$ and new clause $E = (t \lor f \lor f)$. Any 1-in-3-SAT assignment of $\phi$ will set $t$ to $T$ and $f$ to $F$.

2) For each $x_j$ have new var $x'_j$ and clause $D_j = (f \lor x_j \lor x'_j)$. Any 1-in-3-SAT assignment for $\phi$ will set $x_j, x'_j$ to opposites.
Mono 1-in-3-SAT

**Mono 1-in-3-SAT** (**mono-1-in-3-SAT**): Given a formula $E_1 \land \cdots \land E_p$ where all vars occur positively, is there an assignment that satisfies **exactly** one literal-per-clause.

**Thm** 1-in-3-SAT $\leq$ mono-1-in-3-SAT

Given 3CNF form $\phi(x_1, \ldots, x_n) = C_1 \lor \cdots \lor C_k$ want $\phi'$ such that $\phi \in 1\text{-in-3-SAT}$ iff $\phi' \in \text{mono-1-in-3-SAT}$.

1) New Vars $t$, $f$ and new clause $E = (t \lor f \lor f)$. Any 1-in-3-SAT assignment of $\phi$ will set $t$ to $T$ and $f$ to $F$.

2) For each $x_j$ have new var $x_j'$ and clause $D_j = (f \lor x_j \lor x_j')$. Any 1-in-3-SAT assignment for $\phi$ will set $x_j$, $x_j'$ to opposites.

3) For each $C_i$ let $C_i'$ be obtained by replacing every $\overline{x_j}$ with $x_j'$. 
Mono 1-in-3-SAT

**Mono 1-in-3-SAT (mono-1-in-3-SAT):** Given a formula $E_1 \land \cdots \land E_p$ where all vars occur positively, is there an assignment that satisfies **exactly** one literal-per-clause.

**Thm** $1$-in-$3$-SAT $\leq$ mono-$1$-in-$3$-SAT

Given 3CNF form $\phi(x_1, \ldots, x_n) = C_1 \lor \cdots \lor C_k$ want $\phi'$ such that $\phi \in 1$-in-$3$-SAT iff $\phi' \in$ mono-$1$-in-$3$-SAT.

1) New Vars $t, f$ and new clause $E = (t \lor f \lor f)$. Any 1-in-3-SAT assignment of $\phi$ will set $t$ to $T$ and $f$ to $F$.

2) For each $x_j$ have new var $x_j'$ and clause $D_j = (f \lor x_j \lor x_j')$. Any 1-in-3-SAT assignment for $\phi$ will set $x_j, x_j'$ to opposites.

3) For each $C_i$ let $C_i'$ be obtained by replacing every $\overline{x_j}$ with $x_j'$.

$$\phi' = C'_1 \land \cdots \land C'_k \land D_1 \land \cdots \land D_n \land E.$$ 

Leave it to the reader to show $\phi \in 1$-in-$3$-SAT iff $\phi' \in$ mono-$1$-in-$3$-SAT.
A Puzzle we Prove Hard Using mono-1-in-3-SAT

Exposition by William Gasarch—U of MD
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem!

1) A carry can be at most 1. Hence $M = 1$.

2) Since $M = 1$, $S + M +$ poss carry $\leq 10$. Since there is a carry, $S + M +$ poss carry $= 10$ so $O = 0$.

3) Can keep on reasoning like this and we find:

\[
\begin{array}{c}
  9 \quad 5 \quad 6 \quad 7 \\
+ \quad 1 \quad 0 \quad 8 \quad 5 \\
\hline
  1 \quad 0 \quad 6 \quad 5 \quad 2 \\
\end{array}
\]

The Solution to The SEND MORE MONEY Cryptarithms
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! NOT!
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! **NOT!**
We will use it to show that a puzzle we DO care about is NPC.
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! NOT!
We will use it to show that a puzzle we DO care about is NPC

\[
\begin{array}{cccc}
S & E & N & D \\
\hline
+ & M & O & R & E \\
\hline
M & O & N & E & Y \\
\end{array}
\]

The SEND MORE MONEY Cryptarithms
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! NOT!
We will use it to show that a puzzle we DO care about is NPC

\[
\begin{array}{cccccc}
S & E & N & D \\
+ & M & O & R & E \\
\hline
M & O & N & E & Y \\
\end{array}
\]

The SEND MORE MONEY Cryptarithms

1) A carry can be at most 1. Hence \( M = 1 \).
Why is \textit{mono-1-in-3-SAT} Important?

We care about the \textit{mono-1-in-3-SAT} problem! \textbf{NOT!}
We will use it to show that a puzzle we \textbf{DO} care about is \textit{NPC}

\[
\begin{array}{cccc}
S & E & N & D \\
+ & M & O & R \\
\hline
M & O & N & E \\
\end{array}
\]

The SEND MORE MONEY Cryptarithms

1) A carry can be at most 1. Hence \( M = 1 \).
2) Since \( M = 1 \), \( S + M + \text{poss carry} \leq 10 \). Since there is a carry, \( S + M + \text{poss carry} = 10 \) so \( O = 0 \).
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! NOT!
We will use it to show that a puzzle we DO care about is NPC

\[
\begin{array}{cccc}
S & E & N & D \\
+ & M & O & R & E \\
\hline
M & O & N & E & Y \\
\end{array}
\]

The SEND MORE MONEY Cryptarithms
1) A carry can be at most 1. Hence \( M = 1 \).
2) Since \( M = 1 \), \( S + M + \text{poss carry} \leq 10 \). Since there is a carry, \( S + M + \text{poss carry} = 10 \) so \( O = 0 \).
3) Can keep on reasoning like this and we find:
Why is mono-1-in-3-SAT Important?

We care about the mono-1-in-3-SAT problem! NOT!
We will use it to show that a puzzle we DO care about is NPC

\[
\begin{array}{cccc}
S & E & N & D \\
+ & M & O & R & E \\
\hline
M & O & N & E & Y \\
\end{array}
\]

The SEND MORE MONEY Cryptarithms
1) A carry can be at most 1. Hence \( M = 1 \).
2) Since \( M = 1 \), \( S + M + \) poss carry \( \leq 10 \). Since there is a carry, \( S + M + \) poss carry = 10 so \( O = 0 \).
3) Can keep on reasoning like this and we find:

\[
\begin{array}{cccc}
9 & 5 & 6 & 7 \\
+ & 1 & 0 & 8 & 5 \\
\hline
1 & 0 & 6 & 5 & 2 \\
\end{array}
\]

The Solution to The SEND MORE MONEY Cryptarithms
How Did We Solve SEND + MORE = MONEY?

We initially did some reasoning to cut down the number of poss.
How Did We Solve SEND + MORE = MONEY?

We initially did some reasoning to cut down the number of poss.
But past a certain point we had to try all possibilities.
How Did We Solve SEND+MORE=MONEY?

We initially did some reasoning to cut down the number of poss. But past a certain point we had to try all possibilities. Is the general problem NPC?
How Did We Solve SEND + MORE = MONEY?

We initially did some reasoning to cut down the number of poss. But past a certain point we had to try all possibilities.

Is the general problem NPC? Spoiler Alert:
How Did We Solve $\text{SEND} + \text{MORE} = \text{MONEY}$?

We initially did some reasoning to cut down the number of possibilities. But past a certain point we had to try all possibilities. Is the general problem NPC? Spoiler Alert: Yes
Definition of Cryptarithms Problem

We want to show that Cryptarithms is NPC. We need a definition.
We want to show that Cryptarithms is NPC. We need a definition. 

CRYPTARITHM

Input $B, m \in \mathbb{N}$. $\Sigma$ is alphabet of $B$ letters.
$x_0, \ldots, x_{m-1}$. Each $x_i \in \Sigma$.
$y_0, \ldots, y_{m-1}$. Each $y_i \in \Sigma$.
$z_0, \ldots, z_m$. Each $z_i \in \Sigma$. The symbol $z_m$ is optional.
Definition of Cryptarithmetic Problem

We want to show that Cryptarithmetic is NPC. We need a definition.

**CRYPTARITHM**

**Input** $B, m \in \mathbb{N}$. $\Sigma$ is alphabet of $B$ letters.

$x_0, \ldots, x_{m-1}$. Each $x_i \in \Sigma$.

$y_0, \ldots, y_{m-1}$. Each $y_i \in \Sigma$.

$z_0, \ldots, z_m$. Each $z_i \in \Sigma$. The symbol $z_m$ is optional.

**Question** Does there exist an injection $\Sigma \rightarrow \{0, \ldots, B - 1\}$ so that the arithmetic below is correct in base $B$?

\[
\begin{array}{cccc}
  x_{m-1} & \cdots & x_0 \\
  + & y_{m-1} & \cdots & y_0 \\
  \hline
  z_m & z_{m-1} & \cdots & z_0
\end{array}
\]
We Show CRYPTARITHM is NPC

Thm CRYPTARITHM is NPC.
We Show CRYPTARITHM is NPC

**Thm** CRYPTARITHM is NPC. Erika- How will we prove this?
We Show CRYPTARITHM is NPC

**Thm** CRYPTARITHM is NPC. Erika- How will we prove this? We show mono-1-in-3-SAT \( \leq \) CRYPTARITHM. We show an algorithm that will:
We Show CRYPTARITHM is NPC

**Thm** CRYPTARITHM is NPC. Erika- How will we prove this?
We show mono-1-in-3-SAT $\leq$ CRYPTARITHM. We show an algorithm that will:

**Input** $\phi(x_1, \ldots, x_n) = C_1 \land \cdots \land C_m$ where all vars occur positive.
We Show \textsf{CRYPTARITHM} is NPC

\textbf{Thm} CRYPTARITHM is NPC. Erika- How will we prove this? We show \textsf{mono-1-in-3-SAT} $\leq$ CRYPTARITHM. We show an algorithm that will:

\textbf{Input} $\phi(x_1, \ldots, x_n) = C_1 \land \cdots \land C_m$ where all vars occur positive.

\textbf{Output} An instance $J$ of CRYPTARITHM such that TFAE
Thm CRYPTARITHM is NPC. Erika- How will we prove this? We show mono-1-in-3-SAT ≤ CRYPTARITHM. We show an algorithm that will:

Input $\phi(x_1, \ldots, x_n) = C_1 \land \cdots \land C_m$ where all vars occur positive.

Output An instance $J$ of CRYPTARITHM such that TFAE

1. Exists assignment that satisfies exactly one var per clause.
2. Exists solution to CRYPTARITHM $J$. 

We Show CRYPTARITHM is NPC

**Thm** CRYPTARITHM is NPC. Erika- How will we prove this?
We show mono-1-in-3-SAT ≤ CRYPTARITHM. We show an algorithm that will:

**Input** \( \phi(x_1, \ldots, x_n) = C_1 \land \cdots \land C_m \) where all vars occur positive.

**Output** An instance \( J \) of CRYPTARITHM such that TFAE

1. Exists assignment that satisfies exactly one var per clause.
2. Exists solution to CRYPTARITHM \( J \).

We do the reduction in three parts, so three more slides!
We call the parts **gadgets**.
0 and 1

We have 0, 1 ∈ Σ that will live up their name.
0 and 1

We have $0, 1 \in \Sigma$ that will live up their name.
We have $p, q \in \Sigma$ that will help $0$ maps to $0$, $1$ maps to $1$. 
0 and 1

We have $0, 1 \in \Sigma$ that will live up their name. We have $p, q \in \Sigma$ that will help $0$ maps to $0$, $1$ maps to $1$. We then make this part of $J$: 
0 and 1

We have $0, 1 \in \Sigma$ that will live up their name. We have $p, q \in \Sigma$ that will help 0 maps to 0, 1 maps to 1. We then make this part of $J$:

$$
\begin{array}{c}
0 \ p \ 0 \\
0 \ p \ 0 \\
0 \ p \ 0 \\
1 \ q \ 0
\end{array}
$$
We have $0, 1 \in \Sigma$ that will live up their name. We have $p, q \in \Sigma$ that will help $0$ maps to $0$, $1$ maps to $1$. We then make this part of $J$:

$$
\begin{array}{c}
0 \ p \ 0 \\
0 \ p \ 0 \\
\hline
1 \ q \ 0
\end{array}
$$

We leave it to the reader to show that this ensures $0$ maps to $0$ and $1$ maps to $1$. 
Vars $\equiv 0, 1 \pmod{4}$

For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
If $v$ is true then $v \equiv 1 \pmod{4}$.
For every variable \( v \) we have a symbol \( v \in \Sigma \). Our intent is
If \( v \) is true then \( v \equiv 1 \pmod 4 \).
If \( v \) is false then \( v \equiv 0 \pmod 4 \).
**Vars ≡ 0, 1 (mod 4)**

For every variable \( v \) we have a symbol \( v \in \Sigma \). Our intent is
If \( v \) is true then \( v \equiv 1 \) (mod 4).
If \( v \) is false then \( v \equiv 0 \) (mod 4).
The following gadget ensures that \( v \equiv 0, 1 \) (mod 4).

\[
\begin{array}{cccccc}
0 & b & c & 0 & a & 0 \\
0 & b & c & 0 & a & 0 \\
0 & v & d & 0 & b & 0
\end{array}
\]
For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
If $v$ is true then $v \equiv 1 \pmod{4}$.
If $v$ is false then $v \equiv 0 \pmod{4}$.
The following gadget ensures that $v \equiv 0, 1 \pmod{4}$.

\[
\begin{array}{cccccc}
0 & b & c & 0 & a & 0 \\
0 & b & c & 0 & a & 0 \\
0 & v & d & 0 & b & 0 \\
\end{array}
\]

Since $a + a = b$ with no carry, $b \equiv 0 \pmod{2}$. 
For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
If $v$ is true then $v \equiv 1 \pmod{4}$.
If $v$ is false then $v \equiv 0 \pmod{4}$.
The following gadget ensures that $v \equiv 0, 1 \pmod{4}$.

\[
\begin{array}{cccccc}
0 & b & c & 0 & a & 0 \\
0 & b & c & 0 & a & 0 \\
\hline
0 & v & d & 0 & b & 0 \\
\end{array}
\]

Since $a + a = b$ with no carry, $b \equiv 0 \pmod{2}$.
Since $c + c = d$ the carry is $C \in \{0, 1\}$. 
For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
If $v$ is true then $v \equiv 1 \pmod{4}$.
If $v$ is false then $v \equiv 0 \pmod{4}$.
The following gadget ensures that $v \equiv 0, 1 \pmod{4}$.

\[
\begin{array}{cccccc}
0 & b & c & 0 & a & 0 \\
0 & b & c & 0 & a & 0 \\
0 & v & d & 0 & b & 0 \\
\end{array}
\]

Since $a + a = b$ with no carry, $b \equiv 0 \pmod{2}$.
Since $c + c = d$ the carry is $C \in \{0, 1\}$.
Since $b + b = v$, $v = 2b + C$, so $v \equiv 0, 1 \pmod{4}$. 

Note: Do this for all vars $v$, using a different $a, b, c$ for each one.
For every variable $v$ we have a symbol $v \in \Sigma$. Our intent is
If $v$ is true then $v \equiv 1 \pmod{4}$.
If $v$ is false then $v \equiv 0 \pmod{4}$.
The following gadget ensures that $v \equiv 0, 1 \pmod{4}$.

\[
\begin{array}{cccccc}
0 & b & c & 0 & a & 0 \\
0 & b & c & 0 & a & 0 \\
0 & v & d & 0 & b & 0 \\
\end{array}
\]

Since $a + a = b$ with no carry, $b \equiv 0 \pmod{2}$.
Since $c + c = d$ the carry is $C \in \{0, 1\}$.
Since $b + b = v$, $v = 2b + C$, so $v \equiv 0, 1 \pmod{4}$.

**Note** Do this for all vars $v$, using a different $a, b, c$ for each one.
Clauses Need to have Exactly One Var True

Clause is $(x \lor y \lor z)$.

Gadget is:

\begin{align*}
0 & \quad 0 & x & 0 & 1 & 0 & b \\
0 & \quad 0 & y & 0 & 1 & 0 & c \\
0 & \quad 0 & d & 0 & 1 & 0 & d
\end{align*}

\begin{align*}
+ & = b \\
+ & = c \\
= & = d + 1
\end{align*}

Note For each clause use a different $a$, $b$, $c$, $I$.

So if $J$ has a solution then $\phi$ has a 1-in-3 assignment.

Need if $\phi$ has a 1-in-3 assignment then $J$ has sol. Left to reader.
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{ccccccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\(a + a = b\), so \(b \equiv 0 \pmod{2}\). 

Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccc}
0 & 1 & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\[a + a = b, \text{ so } b \equiv 0 \pmod{2}.\]

\[b + b = c, \text{ so } c \equiv 0 \pmod{4}.\]
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\(a + a = b,\) so \(b \equiv 0 \pmod{2}\).

\(b + b = c,\) so \(c \equiv 0 \pmod{4}\).

\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\(a + a = b\), so \(b \equiv 0 \pmod{2}\).
\(b + b = c\), so \(c \equiv 0 \pmod{4}\).
\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).
\(x + y = l\) so \(x + y \equiv l \pmod{4}\).
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\(a + a = b\), so \(b \equiv 0 \pmod{2}\).
\(b + b = c\), so \(c \equiv 0 \pmod{4}\).
\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).
\(x + y = l\) so \(x + y \equiv l \pmod{4}\).
\(l + z = d\) so \(x + y + z \equiv 1 \pmod{4}\).
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0 \\
\end{array}
\]

\(a + a = b\), so \(b \equiv 0 \pmod{2}\).

\(b + b = c\), so \(c \equiv 0 \pmod{4}\).

\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).

\(x + y = l\) so \(x + y \equiv l \pmod{4}\).

\(l + z = d\) so \(x + y + z \equiv 1 \pmod{4}\).

**Note** For each clause use a different \(a, b, c, l\).
**Clauses Need to have Exactly One Var True**

Clause is \((x \lor y \lor z)\). Gadget is:

<table>
<thead>
<tr>
<th></th>
<th>l</th>
<th></th>
<th>x</th>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>b</th>
<th>0</th>
<th>a</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>z</td>
<td></td>
<td>y</td>
<td></td>
<td>0</td>
<td>c</td>
<td>0</td>
<td>b</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>d</td>
<td></td>
<td>l</td>
<td></td>
<td>0</td>
<td>d</td>
<td>0</td>
<td>c</td>
<td>0</td>
<td>b</td>
</tr>
</tbody>
</table>

\(a + a = b\), so \(b \equiv 0 \pmod{2}\).

\(b + b = c\), so \(c \equiv 0 \pmod{4}\).

\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).

\(x + y = l\) so \(x + y \equiv l \pmod{4}\).

\(l + z = d\) so \(x + y + z \equiv 1 \pmod{4}\).

**Note** For each clause use a different \(a, b, c, l\).

So if \(J\) has a solution then \(\phi\) has a 1-in-3 assignment.
Clauses Need to have Exactly One Var True

Clause is \((x \lor y \lor z)\). Gadget is:

\[
\begin{array}{cccccccc}
0 & l & 0 & x & 0 & 1 & 0 & b & 0 & a & 0 \\
0 & z & 0 & y & 0 & c & 0 & b & 0 & a & 0 \\
0 & d & 0 & l & 0 & d & 0 & c & 0 & b & 0
\end{array}
\]

\(a + a = b\), so \(b \equiv 0 \pmod{2}\).
\(b + b = c\), so \(c \equiv 0 \pmod{4}\).
\(d = c + 1\) so \(d \equiv 1 \pmod{4}\).
\(x + y = l\) so \(x + y \equiv l \pmod{4}\).
\(l + z = d\) so \(x + y + z \equiv 1 \pmod{4}\).

**Note** For each clause use a different \(a, b, c, l\).

So if \(J\) has a solution then \(\phi\) has a 1-in-3 assignment.
Need if \(\phi\) has a 1-in-3 assignment then \(J\) has sol. Left to reader.