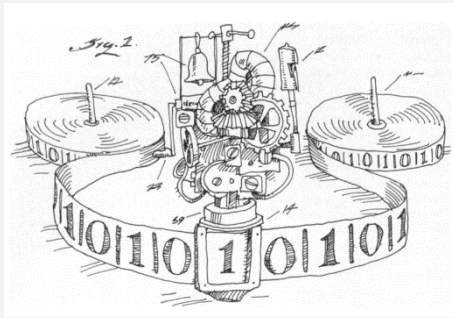
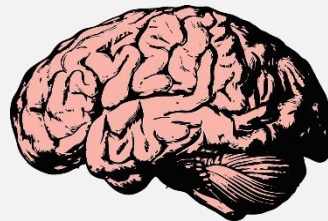
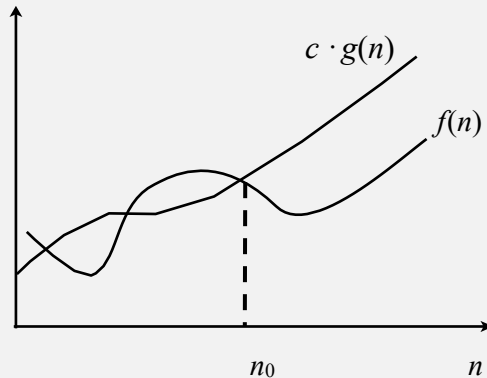


COMP 3602

Intro to Theory of Computation

Lecture 20



Announcements



- Complexity drill due Thu Mar 19
- Two more tests left:
 - Oral exam on NP-completeness: Mon Mar 23
 - Practice problems are posted
 - Sign up by Friday before
 - Written exam on all the material we covered in the course.
 - Fri Mar 27th In class
- Presentations in class:
 - Mar 30 and Apr 1, please sign up for your preferred day
 - All slides due Sun Mar 29th

1. Show that NP is closed under Kleene star, by showing that if L is in NP, then L^* is in NP as well, where every string in L^* is a concatenation of 0 or more strings from L , that is, $L^* = \{ x \mid x = x_1 \dots x_k \text{ for some } k \geq 0 \text{ and strings } x_1, \dots, x_k \text{ such that } \forall i \leq k \ x_i \in L \}$. Use the quantifier/verifier definition of NP.

2. Prove that the problems below are NP-complete. You can assume that the following problems are NP-complete:

3SAT, SubsetSum, Partition, HamPath (LongestPath with $k=|V|$), IndSet and Clique.

- a. DoubleSAT = $\{ \langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments} \}$.
- b. SixOccurrenceSAT = $\{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF with at most 6 occurrences of every variable} \}$
- c. kClique-kIndSet = $\{ \langle G, k \rangle \mid G \text{ contains both a clique of size } k \text{ and an independent set of size } k \}$
- d. Half-Clique = $\{ \langle G \rangle \mid G \text{ has a clique of size } \lceil |V|/2 \rceil \}$
- e. HamCycle = $\{ \langle G \rangle \mid G \text{ contains a cycle which goes through all vertices exactly once} \}$
- f. FourPartition = $\{ \langle (a_1, \dots, a_n) \rangle \mid \forall i \leq n, a_i \geq 0, \text{ and } 1 \dots n \text{ can be partitioned into four disjoint sets } S_1, S_2, S_3 \text{ and } S_4 \text{ such that } \sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i = \sum_{i \in S_3} a_i = \sum_{i \in S_4} a_i \text{ and } S_1 \cup S_2 \cup S_3 \cup S_4 = \{1, \dots, n\} \}$

2. Prove that the problems below are NP-complete. You can assume that the following problems are NP-complete:

3SAT, SubsetSum, Partition, HamPath (LongestPath with $k=|V|$), IndSet and Clique.

- a. DoubleSAT = $\{ \langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments} \}$.
- b. SixOccurrenceSAT = $\{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF with at most 6 occurrences of every variable} \}$
- c. kClique-kIndSet = $\{ \langle G, k \rangle \mid G \text{ contains both a clique of size } k \text{ and an independent set of size } k \}$
- d. Half-Clique = $\{ \langle G \rangle \mid G \text{ has a clique of size } \lceil |V|/2 \rceil \}$
- e. HamCycle = $\{ \langle G \rangle \mid G \text{ contains a cycle which goes through all vertices exactly once} \}$
- f. FourPartition = $\{ \langle (a_1, \dots, a_n) \rangle \mid \forall i \leq n, a_i \geq 0, \text{ and } 1 \dots n \text{ can be partitioned into four disjoint sets } S_1, S_2, S_3 \text{ and } S_4 \text{ such that } \sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i = \sum_{i \in S_3} a_i = \sum_{i \in S_4} a_i \text{ and } S_1 \cup S_2 \cup S_3 \cup S_4 = \{1, \dots, n\} \}$

- **HamPath, HamCycle and TravelingSalesmanProblem:**

- A Hamiltonian path in a graph is a path which visits every vertex exactly once.
 - That is, a longest path in G is of length $|V|=n$
- A Hamiltonian cycle is a simple cycle which visits every vertex exactly once
- $HamPath(G) = \{ \langle G \rangle \mid G \text{ is a directed graph that has a Hamiltonian path} \}$
- $HamCycle(G) = \{ \langle G \rangle \mid G \text{ is a directed graph that has a Hamiltonian cycle} \}$
- $TSP(G) = \{ \langle G, t \rangle \mid \text{a weighted directed } G \text{ has a Hamiltonian cycle of weight } \leq t \}$
- All three problems above are NP-hard for undirected graphs too.

Proving NP-completeness: template.

Goal: show that a language L' is NP-complete.

1. Show that $L' \in NP$

- Explain what potential solutions are and how to check them efficiently.
- That is, explain what y and R are in $L = \{x | \exists y |y| < poly(|x|) \wedge R(x, y)\}$,
- and why y is short and R is polytime computable.

2. Show that L' is NP-hard

- Pick a problem L we have showed to be NP-complete already.

- Show $L \leq_p L'$

a) Construct a polytime reduction f from L to L' ,

- that is, describe how to get $x' = f(x)$ for each x

» **Important: f cannot determine whether $x \in L$!**

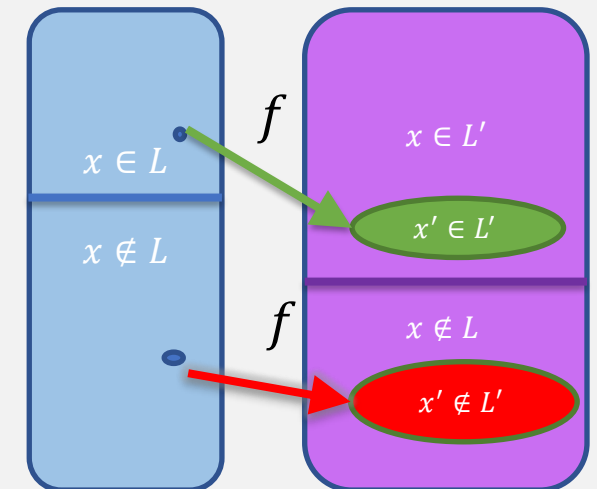
b) Show correctness: two directions.

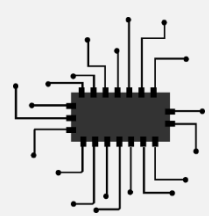
i. $\Rightarrow$ Given a solution y to x , construct a solution y' to $f(x)$

ii. $\Leftarrow$ Given a solution y' to $f(x)$, reconstruct a solution y to x

» **This applies only to $x' = f(x)$, not arbitrary element of L' !**

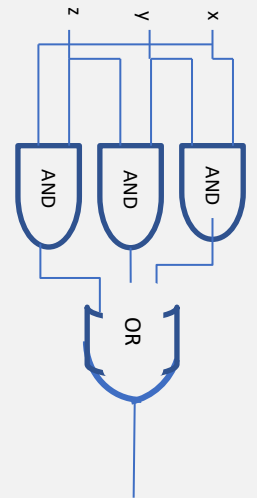
c) Explain briefly why f is polynomial-time computable





Tseitin transformation.

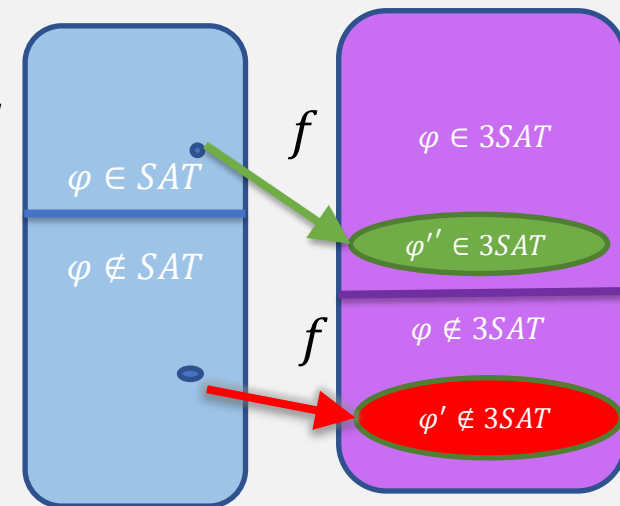
- Recursive definition of a propositional formula:
 - *A variable or a constant; if A, B are propositional formulas, then so are $\neg A, A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$*
- Tseitin transformation:
 - A polytime-computable function f that transforms an arbitrary propositional formula φ into a 3CNF formula φ' so that φ is satisfiable iff φ' is satisfiable, defined as follows:
 - For every connective introduce a variable coding its value
 - Eg, for each subformula C which is not a variable or a constant, introduce a variable z_C
 - For each variable/constant, leave them as coding their own value.
 - Make a 3CNF formula φ_C stating that z_C is computed correctly
 - Eg if C is of the form $A \rightarrow B$, with z_A, z_B coding values of A, B then want $(z_A \rightarrow z_B) \leftrightarrow z_C$
 - Converting $(z_A \rightarrow z_B) \leftrightarrow z_C$ to CNF, get $\varphi_C = (\neg z_A \vee z_B \vee \neg z_C) \wedge (z_A \vee z_C) \wedge (\neg z_B \vee z_C)$
 - Finally, state that the outermost connective has value true.
 - Eg if C was an outermost connective, add $\wedge z_C$.
 - The full formula φ' is $z_{C_{out}} \wedge \bigwedge_C \varphi_C$



- To prove that 3SAT is NP-hard, let's show $SAT \leq_p 3SAT$

- Construct a polytime-computable function f that transforms an arbitrary propositional formula φ into a 3CNF formula φ' so that φ is satisfiable iff φ' is satisfiable.

- Let $f(\varphi) = \varphi'$ where φ' is a Tseitin transformation of φ



- Correctness proof:

- $\Rightarrow$ If $a_1 \dots a_n$ was a satisfying assignment for φ , a satisfying assignment for φ' is $a_1 \dots a_n$ together with correct values for all z_C
- $\Leftarrow$ A satisfying assignment to φ' contains in particular a satisfying assignment to variables of φ : this is the assignment to its first n variables.

- Finally, the construction is polytime: recursively unwind φ from the outermost connective.

3. Show that if $P=NP$, then every language in P other than $A = \emptyset$ and $A = \Sigma^*$ is NP-complete with respect to polytime reductions.
4. Show that Unary-SubsetSum, a variant of SubsetSum where all numbers are given in unary, is solvable in polynomial time.

Questions/comments?

Proving NP-completeness: template.

Goal: show that a language L' is NP-complete.

1. Show that $L' \in NP$

- Explain what potential solutions are and how to check them efficiently.
- That is, explain what y and R are in $L = \{x | \exists y |y| < poly(|x|) \wedge R(x, y)\}$,
- and why y is short and R is polytime computable.

2. Show that L' is NP-hard

- Pick a problem L we have showed to be NP-complete already.

- Show $L \leq_p L'$

a) Construct a polytime reduction f from L to L' ,

- that is, describe how to get $x' = f(x)$ for each x

» **Important: f cannot determine whether $x \in L$!**

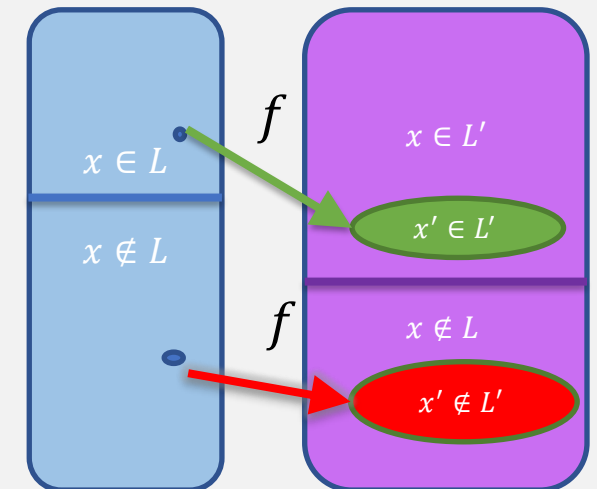
b) Show correctness: two directions.

i. $\Rightarrow$ Given a solution y to x , construct a solution y' to $f(x)$

ii. $\Leftarrow$ Given a solution y' to $f(x)$, reconstruct a solution y to x

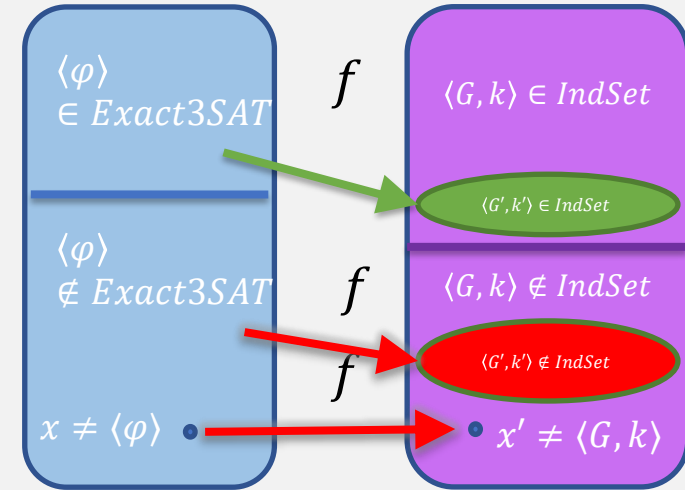
» **This applies only to $x' = f(x)$, not arbitrary element of L' !**

c) Explain briefly why f is polynomial-time computable



Independent Set is NP-complete.

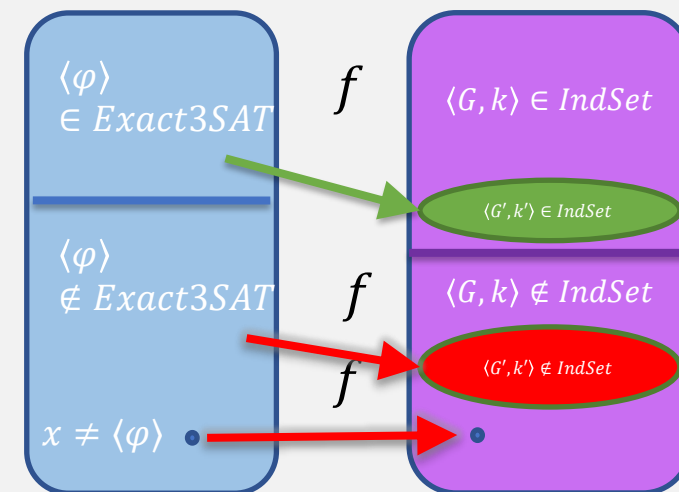
- $Exact3SAT = \{\varphi \mid \varphi \text{ is a satisfiable CNF formula with 3 literals per clause}\}$
 - Exercise: show that $3SAT \leq_p Exact3SAT$
 - Notation: n is the number of variables and m is the number of clauses
- IndSet: $\{\langle G, k \rangle \mid \text{graph } G \text{ has an independent set } S \text{ st } |S| \geq k\}$
- **Theorem: IndSet is NP-complete.**
 - IndSet is in NP: shown before.
 - Potential solution: set S of vertices.
 - Verify: that $|S| \geq k$, and $\forall v \in S \neg E(u, v)$
 - Checking $|S|^2$ edges takes time $O(|S|^2)$
 - IndSet is NP-hard: we will show that $Exact3SAT \leq_p IndSet$
 - Want: a polytime function f that, given any formula φ , constructs a pair $\langle G', k' \rangle$ such that a satisfying assignment to φ corresponds to an independent set in G' of size k' , and from any independent set of size k' in G' , can recover a satisfying assignment to φ .
 - If φ is not satisfiable, there should be no size k' independent sets in G' .
 - f does not know if φ is satisfiable!



- $Exact3SAT = \{\varphi \mid \varphi \text{ is a satisfiable CNF formula with 3 literals per clause}\}$
 - Notation: n is the number of variables and m is the number of clauses
- IndSet: $\{\langle G, k \rangle \mid \text{graph } G \text{ has an independent set } S \text{ st } |S| \geq k\}$

– IndSet is NP-hard: we will show that $Exact3SAT \leq_p IndSet$

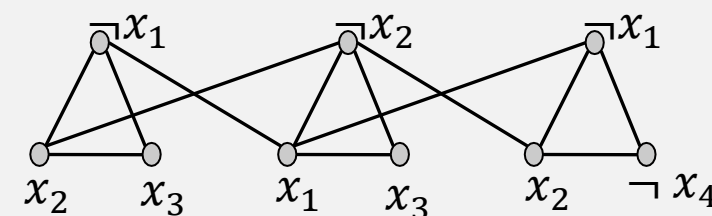
- Want: a polytime function f that, given φ , constructs a pair $\langle G', k' \rangle$ s.t. φ is satisfiable iff there is an ind. set in G' of size k' .
 - Idea: if φ is satisfiable by some assignment $a_1 \dots a_n$, then every clause has a literal set to true by this assignment.
- Construction:
 - Set $k' = m$, the number of clauses in φ
 - The vertices of G' are all occurrences of literals in φ
 - » That is, $|V'| = 3m$
 - For each clause in φ , create a triangle on vertices labeled by literals in that clause
 - » This will ensure that each clause can contribute at most one vertex to the independent set.
 - Finally, for each pair of contradictory literals (a variable and its negation), put an edge between them.



$$\begin{aligned} \varphi = & (\neg x_1 \vee x_2 \vee x_3) \\ & \wedge (x_1 \vee \neg x_2 \vee x_3) \\ & \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \end{aligned}$$

$$f(\varphi) = \langle G', k' \rangle,$$

where $k' = 3$ and G' is:



- IndSet is NP-hard: we will show that $Exact3SAT \leq_p IndSet$

- Construction:

- » Set $k' = m$, the number of clauses in φ
- » The vertices of G' are all occurrences of literals in φ
- » For each clause in φ , create a triangle
- » Connect every pair of contradictory literals.

- Correctness:

- $\Rightarrow$ Suppose $a_1 \dots a_n$ is a satisfying assignment to φ .

- » Pick, for every clause, one literal satisfied by this assignment. We will show that the set of corresponding vertices forms an independent set S of size $k' = m$ in G'
- » S contains one vertex for every clause, so $|S| = k'$.
- » It cannot be that some variable x is set to true in one clause and $\neg x$ is set to true in another. So there can be no edges between variables picked from different clauses, and therefore S is an independent set.

- $\Leftarrow$ Suppose S is an independent set of size k' in G' . Let's recover a satisfying assignment for φ from S .

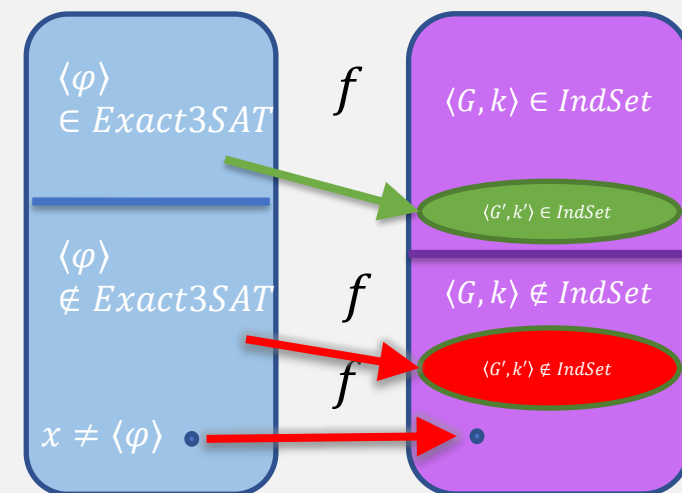
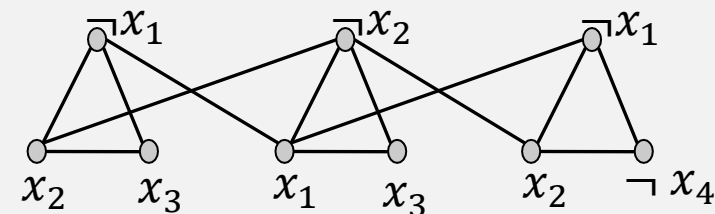
- » For every vertex in S , if it comes from a negated variable, set that variable to false, otherwise to true.
 - Since S is an independent set, and there are edges between contradictory literals, we will never try to set both a variable and its negation to the same value.
- » It might be that not all variables occur in S . That's OK: just set remaining variables arbitrarily (eg, all to false).
- » The resulting assignment satisfies φ since in every clause there is a true literal

- Polytime: f is computable in polytime since it only creates polynomially many vertices and edges, and each follows immediately from the description of φ .

- »

$$\begin{aligned}\varphi = & (\neg x_1 \vee x_2 \vee x_3) \\ & \wedge (x_1 \vee \neg x_2 \vee x_3) \\ & \wedge (\neg x_1 \vee x_2 \vee \neg x_4)\end{aligned}$$

$f(\varphi) = \langle G', k' \rangle$,
where $k' = 3$ and G' is:



CLIQUE is NP-complete.

- IndSet: $\{\langle G, k \rangle \mid \text{graph } G \text{ has an independent set } S \text{ st } |S| \geq k\}$
- Corollary: CLIQUE is NP-complete.
 - CLIQUE is in NP: shown before.
 - Potential solution: set S of vertices.
 - Check: that $|S| \geq k$, and $\forall u, v \in S \ E(u, v)$
 - Checking $|S|^2$ edges takes time $O(|S|^2)$
 - CLIQUE is NP-hard:
 - We just proved that IndSet is NP-complete
 - And we showed earlier that $\text{IndSet} \leq_p \text{CLIQUE}$
 - By transitivity of reductions, that means that CLIQUE is also NP-hard.

$x \neq \langle \varphi \rangle$

$x' \neq \langle G, k \rangle$



SubsetSum and friends

- Assume $\mathbb{N}$ starts with 0. All numbers in the following are written in binary.
- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $\text{Partition} = \{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$
- $\text{Knapsack} = \{\langle A, b, c \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } b, c \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} b \leq \sum_{i \in S} a_i \leq c\}$
- We will show that SubsetSum and Partition are NP-complete.
 - NP-completeness of Knapsack: exercise

Proving NP-completeness: template.

Goal: show that a language L' is NP-complete.

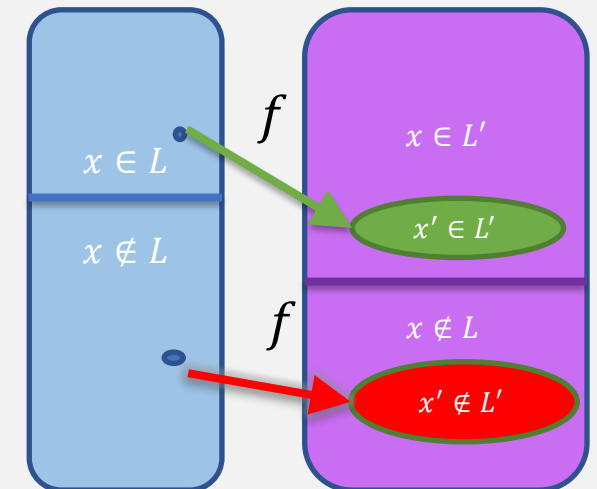
1. Show that $L' \in NP$

- Explain what potential solutions are and how to check them efficiently.
- That is, explain what y and R are in $L = \{x | \exists y |y| < poly(|x|) \wedge R(x, y)\}$,
- and why y is short and R is polytime computable.

2. Show that L' is NP-hard

- Pick a problem L we have showed to be NP-complete already.
- Show $L \leq_p L'$

- Construct a polytime reduction f from L to L' ,
 - that is, describe how to get $x' = f(x)$ for each x
 - » **Important: f cannot determine whether $x \in L$!**
- Show correctness: two directions.
 - $\Rightarrow$ Given a solution y to x , construct a solution y' to $f(x)$
 - $\Leftarrow$ Given a solution y' to $f(x)$, reconstruct a solution y to x
 - » **This applies only to $x' = f(x)$, not arbitrary element of L' !**
- Explain briefly why f is polynomial-time computable





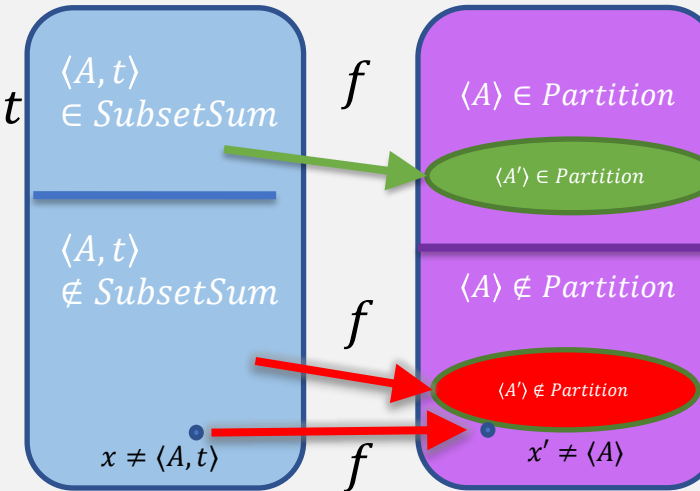
SubsetSum and friends

- Assume $\mathbb{N}$ starts with 0. All numbers in the following are written in binary.
- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $\text{Partition} = \{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$
- Claim: SubsetSum and Partition in NP.
 - Candidate solutions are the respective sets S , which are of polynomial size.
 - Taking sums and checking equalities and inequalities can be done in polynomial time.
- To prove NP-hardness, we will show $3SAT \leq_p \text{SubsetSum} \leq_p \text{Partition}$
 - We will do the easier reduction $\text{SubsetSum} \leq_p \text{Partition}$ first

- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $\text{Partition} = \{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$

• Claim: $\text{SubsetSum} \leq_p \text{Partition}$

- Want: $f(\langle A, t \rangle) = \langle A' \rangle$ such that
 $\exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t \leftrightarrow \exists S' \subseteq \{1, \dots, n'\} \sum_{i \in S'} a'_i = \sum_{i \notin S'} a'_i$
- Idea: If $t = \frac{1}{2} \sum_{i=1}^n a_i$, then A has a partition iff it has a set summing up to t
- What if $t > \frac{1}{2} \sum_{i=1}^n a_i$?
 - Then add a new element a_{n+1} to A so that $\frac{1}{2}(a_{n+1} + \sum_{i=1}^n a_i) = t$
- But what if $t < \frac{1}{2} \sum_{i=1}^n a_i$?
 - Notice that $\langle A, t \rangle \in \text{SubsetSum} \leftrightarrow \langle A, (\sum_{i=1}^n a_i) - t \rangle \in \text{SubsetSum}$
 - Add a new element a_{n+1} so that $\frac{1}{2}(a_{n+1} + \sum_{i=1}^n a_i) = \frac{1}{2} \sum_{i=1}^n a_i - t$
- Generally, we never take anything out in a reduction, always add to the instance.



$$f(\langle (a_1, \dots, a_n), t \rangle) = \begin{cases} \langle a_1, \dots, a_n \rangle, & t = \frac{1}{2} \sum_{i=1}^n a_i \\ \langle a_1, \dots, a_n, 2t - \sum_{i=1}^n a_i \rangle, & t > \frac{1}{2} \sum_{i=1}^n a_i \\ \langle a_1, \dots, a_n, \sum_{i=1}^n a_i - 2t \rangle, & t < \frac{1}{2} \sum_{i=1}^n a_i \end{cases}$$

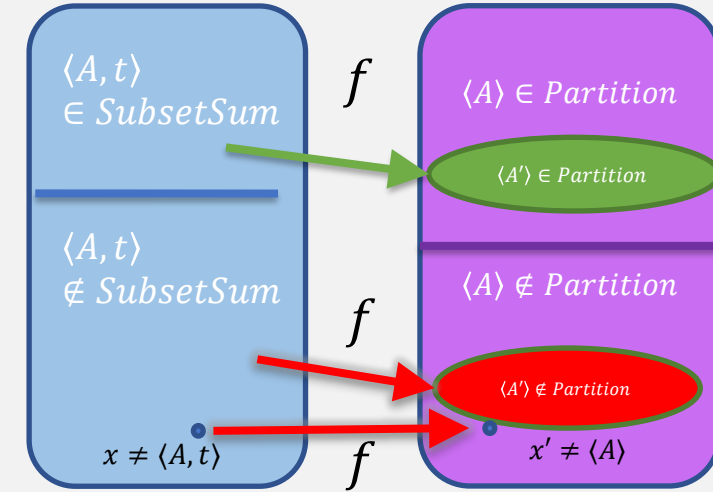
- SubsetSum = $\{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- Partition = $\{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$

• Claim: SubsetSum $\leq_p$ Partition

- Want: $f(\langle A, t \rangle) = \langle A' \rangle$ such that

$$\exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t \leftrightarrow \exists S' \subseteq \{1, \dots, n'\} \sum_{i \in S'} a'_i = \sum_{i \notin S'} a'_i$$

$$f(\langle (a_1, \dots, a_n), t \rangle) = \begin{cases} \langle a_1, \dots, a_n \rangle, & t = \frac{1}{2} \sum_{i=1}^n a_i \\ \langle a_1, \dots, a_n, 2t - \sum_{i=1}^n a_i \rangle, & t > \frac{1}{2} \sum_{i=1}^n a_i \\ \langle a_1, \dots, a_n, \sum_{i=1}^n a_i - 2t \rangle, & t < \frac{1}{2} \sum_{i=1}^n a_i \end{cases}$$



Correctness proof: Let $t < \frac{1}{2} \sum_{i=1}^n a_i$. Other cases: exercise.

$\Rightarrow$ Suppose that $\exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t$.

Take $S' = S \cup \{n+1\}$. $\sum_{i \in S'} a'_i = t + \sum_{i=1}^n a_i - 2t = \sum_{i=1}^n a_i - t = \sum_{i \notin S} a_i = \sum_{i \notin S'} a'_i$

$\Leftarrow$ Suppose that $\exists S' \subseteq \{1, \dots, n+1\} \sum_{i \in S'} a'_i = \sum_{i \notin S'} a'_i$.

If $n+1 \in S'$, take $S = S' - \{n+1\}$. Otherwise, take $S = \bar{S}' - \{n+1\}$

By construction, $\sum_{i \in S'} a'_i = \sum_{i \notin S'} a'_i = \sum_{i=1}^n a_i - t$. So $\sum_{i \in S} a_i = (\sum_{i=1}^n a_i - t) - (\sum_{i=1}^n a_i - 2t) = t$