Design and Analysis of Algorithms CS218M

NP Completeness

Paritosh Pandya

Indian Institute of Technology, Bombay

Autumn, 2022

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.
- These algorithms belong to complexity class $\mathbb{P}$.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.
- These algorithms belong to complexity class $\mathbb{P}$.
- It is believed that such algorithms are tractable.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.
- These algorithms belong to complexity class $\mathbb{P}$.
- It is believed that such algorithms are tractable.
- A problem Q is polynomial time solvable if there an algorithm $A \in \mathbb{P}$ solving Q.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.
- These algorithms belong to complexity class $\mathbb{P}$.
- It is believed that such algorithms are tractable.
- A problem Q is polynomial time solvable if there an algorithm $A \in \mathbb{P}$ solving Q.
- We abuse the notation to say that $Q \in \mathbb{P}$.

Complexity Class $\mathbb{P}$: Polynomial time Solvable Problems

- We saw several problems for which we gave algorithms with time complexity $O(f(n))$ where $f(n)$ is a polynomial in n.
- Examples: integer multiplication, shortest path.
- These algorithms belong to complexity class $\mathbb{P}$.
- It is believed that such algorithms are tractable.
- A problem Q is polynomial time solvable if there an algorithm $A \in \mathbb{P}$ solving Q.
- We abuse the notation to say that $Q \in \mathbb{P}$.

Theorem

$\mathbb{P} \ = \ Co - \mathbb{P}$

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.
- Decision problem: Has "yes/no" output. E.g. For given k find if the graph has a path of length at most k.

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.
- Decision problem: Has "yes/no" output. E.g. For given k find if the graph has a path of length at most k.
- Two types of problems are inter-reducible with polynomial blowup.

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.
- Decision problem: Has "yes/no" output. E.g. For given k find if the graph has a path of length at most k.
- Two types of problems are inter-reducible with polynomial blowup.
- We will deal with decision problems only.

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.
- Decision problem: Has "yes/no" output. E.g. For given k find if the graph has a path of length at most k.
- Two types of problems are inter-reducible with polynomial blowup.
- We will deal with decision problems only.
- For problem Q we encode its instance X as a string $\langle X \rangle$ over some Σ. Then, Q is the set of "yes" instance strings. The corresponding decision problem answers whether a given string in Σ^* is in Q.

Some Technicalities

- Optimization problem: Gives output with optimal value while meeting desired constraints. E.g. Find the length of the shortest path in directed graph G.
- Decision problem: Has "yes/no" output. E.g. For given k find if the graph has a path of length at most k.
- Two types of problems are inter-reducible with polynomial blowup.
- We will deal with decision problems only.
- For problem Q we encode its instance X as a string $\langle X \rangle$ over some Σ. Then, Q is the set of "yes" instance strings. The corresponding decision problem answers whether a given string in Σ^* is in Q.
- Example:
  $PRIMES = \{x \in \{0,1\}^* \mid x \text{ represents a prime in binary}\}$.

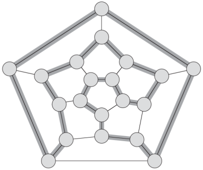
Examples

- *SAT* set of all strings representing satisfiable boolean formulas.

Facts

Examples

- *SAT* set of all strings representing satisfiable boolean formulas.
- *HAMILTON* is collection of all graphs which have a Hamiltonian cycle.



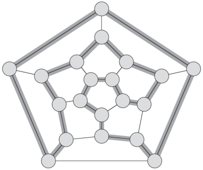
- *EULER* is collectionof all graphs which have a Euler cycle.

Facts

- $EULER \in \mathbb{P}$ (why?)

Examples

- *SAT* set of all strings representing satisfiable boolean formulas.
- *HAMILTON* is collection of all graphs which have a Hamiltonian cycle.



- *EULER* is collectionof all graphs which have a Euler cycle.
- *PATH* all graphs having a path from specified s to t of length at most k.

Facts

- $EULER \in \mathbb{P}$ (why?)
- $PATH \in \mathbb{P}$ (why?)

Beyond $\mathbb{P}$

- Question: Are there problems which are not polynomial time solvable?

- Question: Are there problems which are not polynomial time solvable?
- Answer: Yes
  - Satisfiability of Presburger Arithmetic formula (without multiplication symbol)
    $$\forall x \exists y.((x + x + y > 7) \wedge (x - y < 3)).$$
  - Equality of regular expressions with squaring.
  - Emptiness of Extended regular expressions (with intersection and complement).

Beyond $\mathbb{P}$

- Question: Are there problems which are not polynomial time solvable?
- Answer: Yes
  - Satisfiability of Presburger Arithmetic formula (without multiplication symbol)
    $$\forall x \exists y.((x + x + y > 7) \wedge (x - y < 3)).$$
  - Equality of regular expressions with squaring.
  - Emptiness of Extended regular expressions (with intersection and complement).
- However there are many natural problems whose status is unresolved.

Beyond $\mathbb{P}$

- Question: Are there problems which are not polynomial time solvable?
- Answer: Yes
  - Satisfiability of Presburger Arithmetic formula (without multiplication symbol)
    $$\forall x \exists y.((x + x + y > 7) \wedge (x - y < 3)).$$
  - Equality of regular expressions with squaring.
  - Emptiness of Extended regular expressions (with intersection and complement).
- However there are many natural problems whose status is unresolved.
- Examples:
  - *IND* Determining whether a graph G has a subset of k vertices where no two vertices are connected by an edge.
  - *HAMILTON* Existence of Hamiltonian cycle in a Graph.
  - *SAT* Satisfiability of boolean formula.

Beyond $\mathbb{P}$

- Question: Are there problems which are not polynomial time solvable?
- Answer: Yes
  - Satisfiability of Presburger Arithmetic formula (without multiplication symbol)
    $$\forall x \exists y.((x + x + y > 7) \wedge (x - y < 3)).$$
  - Equality of regular expressions with squaring.
  - Emptiness of Extended regular expressions (with intersection and complement).
- However there are many natural problems whose status is unresolved.
- Examples:
  - *IND* Determining whether a graph G has a subset of k vertices where no two vertices are connected by an edge.
  - *HAMILTON* Existence of Hamiltonian cycle in a Graph.
  - *SAT* Satisfiability of boolean formula.
- Above examples belong to an important class of problems called $\mathbb{NP}$ problems.

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

- Finding solution may be difficult and may involve exhaustive search.

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

- Finding solution may be difficult and may involve exhaustive search.
- It is easy to verify that the solution to a problem instance X is correct using a succinct certificate Y.

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

- Finding solution may be difficult and may involve exhaustive search.
- It is easy to verify that the solution to a problem instance X is correct using a succinct certificate Y.
- The certificate Y must be of size polynomial in the size of X.

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

- Finding solution may be difficult and may involve exhaustive search.
- It is easy to verify that the solution to a problem instance X is correct using a succinct certificate Y.
- The certificate Y must be of size polynomial in the size of X.
- The verification must be done by a poly-time algorithm $A(X, Y) \in \mathbb{P}$.

Class $\mathbb{NP}$: Polynomial time Verifiable Problems

Nondeterministic Polynomial Time Solvable Problems.
Characteristics:

- Finding solution may be difficult and may involve exhaustive search.
- It is easy to verify that the solution to a problem instance X is correct using a succinct certificate Y.
- The certificate Y must be of size polynomial in the size of X.
- The verification must be done by a poly-time algorithm $A(X, Y) \in \mathbb{P}$.
- Example: For boolean formula $((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$
  certificate $x_1 = 0,\ x_2 = 0,\ x_3 = 1,\ x_4 = 1$

Definition of Class $\mathbb{NP}$

Definition

$L \in \mathbb{NP}$ if and only if there exists a two input algorithm $A(x, y) \in \mathbb{P}$ and a constant c such that

$$L = \{x \in \Sigma^* \ | \ \exists y. |y| = O(|x|^c) \wedge A(x, y) = 1\}$$

Definition of Class $\mathbb{NP}$

Definition

$L \in \mathbb{NP}$ if and only if there exists a two input algorithm $A(x, y) \in \mathbb{P}$ and a constant c such that

$$L = \{x \in \Sigma^* \mid \exists y. |y| = O(|x|^c) \wedge A(x, y) = 1\}$$

Algorithms to Solve L: Given such A and c, for any input x

Definition of Class $\mathbb{NP}$

Definition

$L \in \mathbb{NP}$ if and only if there exists a two input algorithm $A(x, y) \in \mathbb{P}$ and a constant c such that

$$L = \{x \in \Sigma^* \mid \exists y. |y| = O(|x|^c) \wedge A(x, y) = 1\}$$

Algorithms to Solve L: Given such A and c, for any input x

- Nondeterministically guess y. Output $A(x, y)$.

Definition of Class $\mathbb{NP}$

Definition

$L \in \mathbb{NP}$ if and only if there exists a two input algorithm $A(x, y) \in \mathbb{P}$ and a constant c such that

$$L = \{x \in \Sigma^* \mid \exists y. |y| = O(|x|^c) \wedge A(x, y) = 1\}$$

Algorithms to Solve L: Given such A and c, for any input x

- Nondeterministically guess y. Output $A(x, y)$.
- Systematically enumerate every y and compute $A(x, y)$. If any instance $A(X, Y)$ has answer "yes" then we answer "yes" and return. Otherwise try next y. Finally. answer "no". The time of this Deterministic algorithm is $O(2^{Poly(|X|)})$.

$\mathbb{P}$ versus $\mathbb{NP}$

- $SAT \in \mathbb{NP}$ (How?)

- $SAT \in \mathbb{NP}$ (How?)
- $HAMILTON \in \mathbb{NP}$ (How?)

$\mathbb{P}$ versus $\mathbb{NP}$

- $SAT \in \mathbb{NP}$ (How?)
- $HAMILTON \in \mathbb{NP}$ (How?)
- $EULER \in \mathbb{NP}$ (How?)

- *SAT* $\in \mathbb{NP}$ (How?)
- *HAMILTON* $\in \mathbb{NP}$ (How?)
- *EULER* $\in \mathbb{NP}$ (How?)
- We will see many more examples of problems in $\mathbb{NP}$.

- $SAT \in \mathbb{NP}$ (How?)
- $HAMILTON \in \mathbb{NP}$ (How?)
- $EULER \in \mathbb{NP}$ (How?)
- We will see many more examples of problems in $\mathbb{NP}$.

Theorem

$\mathbb{P} \subseteq \mathbb{NP}$

- $SAT \in \mathbb{NP}$ (How?)
- $HAMILTON \in \mathbb{NP}$ (How?)
- $EULER \in \mathbb{NP}$ (How?)
- We will see many more examples of problems in $\mathbb{NP}$.

Theorem

$\mathbb{P} \subseteq \mathbb{NP}$

Natural Question: Is $SAT \in \mathbb{P}$

A Major Open Problem in CS [Cook71-Levin73]

Is $\mathbb{P} = \mathbb{NP}$?

CMI

ABOUT PROGRAMS PEOPLE MILLENNIUM PROBLEMS PUBLICATIONS EVE

The Millennium Prize Problems

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) established seven *Prize Problems*. The Prizes were conceived to record some of the most difficult problems with which mathematicians were grappling at the turn of the second millennium; to elevate in the consciousness of the general public the fact that in mathematics, the frontier is still open and abounds in important unsolved problems; to emphasize the importance of working towards a solution of the deepest, most difficult problems; and to recognize achievement in mathematics of historical magnitude.

Clay Millenium Problems (List)

Millennium Problems

Yang–Mills and Mass Gap

Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

Riemann Hypothesis

The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Navier–Stokes Equation

This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certitude, but also understanding.

Hodge Conjecture

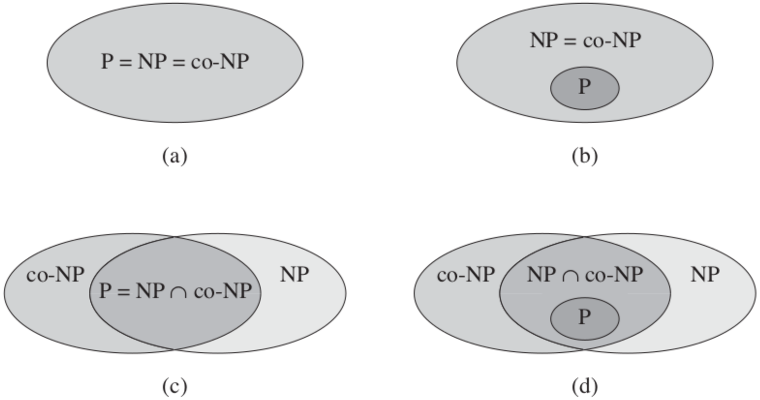
The answer to this conjecture determines how much of the topology of the solution set of a system of algebraic equations can be defined in terms of further algebraic equations. The Hodge conjecture is known in certain special cases, e.g., when the solution set has dimension less than four. But in dimension four it is unknown.

$$\overline{Prime} = \{ \langle n \rangle \mid n \text{ is not a prime} \}$$

$$\overline{Prime} \in NP$$

$$Prime \in Co\text{-}NP$$

Potential Scenarios



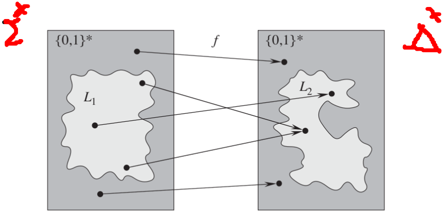
There is strong belief (but no proof!) that $\mathbb{P} \subset \mathbb{NP}$. .

Polynomial Time Reduction

Let $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Delta^*$ be two decision problems. A function $f : \Sigma^* \to \Delta^*$ is called a polynomial time reduction from L_1 to L_2 (denoted $f : L_1 \leq_P L_2$) iff

- f is total.
- There exists a c such that $f(x)$ is computable in time $O(|x|^c)$ for all x.
- $x \in L_1$ iff $f(x) \in L_2$.

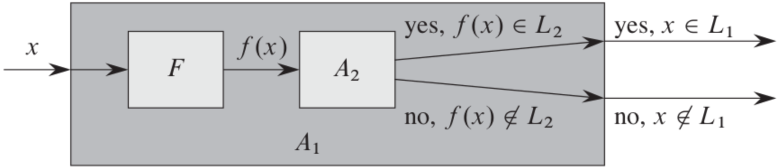
Notation $L_1 \leq_P L_2$ denotes $\exists f$ such that $f : L_1 \leq_P L_2$.



Theorem

If $f : L_1 \leq_P L_2$ and $L_2 \in \mathbb{P}$ then $L_1 \in \mathbb{P}$.

Proof:



Time taken for computing A_1 is polynomial in x (why?)

$|f(x)| = |x|^{c_1}$

$\mathbb{NP}$-Complete Problems

A problem $L \subseteq \Sigma^*$ is called $\mathbb{NP}$-complete (denoted $\mathbb{NPC}$) if

- $L \in \mathbb{NP}$.
- For every $L' \in \mathbb{NP}$, we have $L' \leq_P L$.
  (This shows that L is at least as hard as L'.)

If only second condition is satisfied we say that L is $\mathbb{NP}$-hard.

$\mathbb{NP}$-Complete Problems

A problem $L \subseteq \Sigma^*$ is called $\mathbb{NP}$-complete (denoted $\mathbb{NPC}$) if

- $L \in \mathbb{NP}$.
- For every $L' \in \mathbb{NP}$, we have $L' \leq_P L$.
  (This shows that L is at least as hard as L'.)

If only second condition is satisfied we say that L is $\mathbb{NP}$-hard.

Theorem (Proving $\mathbb{NPC}$ by reduction)

If $L' \leq_P L$ and L' is $\mathbb{NPC}$ then L is $\mathbb{NP}$-hard. Additionally if $L \in \mathbb{NP}$ then L is $\mathbb{NPC}$.

$\mathbb{NP}$-Complete Problems

A problem $L \subseteq \Sigma^*$ is called $\mathbb{NP}$-complete (denoted $\mathbb{NPC}$) if

- $L \in \mathbb{NP}$.
- For every $L' \in \mathbb{NP}$, we have $L' \leq_P L$.
  (This shows that L is at least as hard as L'.)

If only second condition is satisfied we say that L is $\mathbb{NP}$-hard.

Theorem (Proving $\mathbb{NPC}$ by reduction)

If $L' \leq_P L$ and L' is $\mathbb{NPC}$ then L is $\mathbb{NP}$-hard. Additionally if $L \in \mathbb{NP}$ then L is $\mathbb{NPC}$.

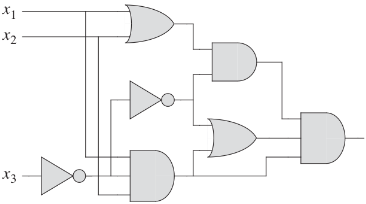
Proof: Let L, L' as above. Then for all $L'' \in \mathbb{NP}$, we have $L'' \leq_P L'$. By transitivity, then $L'' \leq_P L$. Hence, $L \in \mathbb{NPC}$.

Outline

We can prove a wide class of problems to be $\mathbb{NPC}$.

- (Initial Problem) Circuit Satisfaction Problem: Given a combinational circuit made out of AND, OR, NOT gates, decide whether any input makes the circuit output 1.
- (More problems by Reduction) Independat Set, Vertex Cover, k-Clique, set Packing, set Cover, sat, 3-sat, Hamiltonian Cycle, Travelling Salesman, 3-dimnesional-matching, graph-coloring, subsetsum

- Boolean Combinational Circuit made of *AND*, *OR NOT* gates, connected by wires.
- Input wires, single output wire, internal wire
- gates with n inputs, one output, fan out.
- Circuit satisfiability problem Give a circuit C is there an assignment of values to input wires which makes the output wire 1? (Similar to SAT).
  $CIRCUIT\_SAT = \{\langle C \rangle \mid C \text{ has a satisfying assignment } \}$.



$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.

$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.
- Let val be an assignmnt of booelan value 0 or 1 to each wire. We shall use val as succinct certificate.

$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.
- Let val be an assignmnt of booelan value 0 or 1 to each wire. We shall use val as succinct certificate.
- We can have a poly-time algorithm $A(\langle C \rangle, \langle val \rangle)$ which outputs "yes" if and only if
  - truth assignment at input and output of each logic element is consistent.
  - assignment to output wire is 1

$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.
- Let val be an assignmnt of booelan value 0 or 1 to each wire. We shall use val as succinct certificate.
- We can have a poly-time algorithm $A(\langle C \rangle, \langle val \rangle)$ which outputs "yes" if and only if
  - truth assignment at input and output of each logic element is consistent.
  - assignment to output wire is 1
- Hence, C is satisfiable iff there exists val such that $A(\langle C \rangle, \langle val \rangle)$ outputs "yes".

$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.
- Let val be an assignmnt of booelan value 0 or 1 to each wire. We shall use val as succinct certificate.
- We can have a poly-time algorithm $A(\langle C\rangle, \langle val\rangle)$ which outputs "yes" if and only if
  - truth assignment at input and output of each logic element is consistent.
  - assignment to output wire is 1
- Hence, C is satisfiable iff there exists val such that $A(\langle C\rangle, \langle val\rangle)$ outputs "yes".
- A is computable in time polynomial in $|C|$.

$CIRCUIT\_SAT \in NP$

- Let C given circuit with set of wires W.
- Let val be an assignmnt of booelan value 0 or 1 to each wire. We shall use val as succinct certificate.
- We can have a poly-time algorithm $A(\langle C \rangle, \langle val \rangle)$ which outputs "yes" if and only if
  - truth assignment at input and output of each logic element is consistent.
  - assignment to output wire is 1
- Hence, C is satisfiable iff there exists val such that $A(\langle C \rangle, \langle val \rangle)$ outputs "yes".
- A is computable in time polynomial in $|C|$.
- Hence, $CIRCUIT\_SAT \in NP$.

CIRCUIT_SAT is $\mathbb{NP}$-hard

(Proof Schema) Let $L \in \mathbb{NP}$. Hence, there is $A(x, y) \in \mathbb{P}$ s.t. $x \in L$ iff $\exists y. A(x, y) = 1$.

$CIRCUIT\_SAT$ is $\mathbb{NP}$-hard

(Proof Schema) Let $L \in \mathbb{NP}$. Hence, there is $A(x, y) \in \mathbb{P}$ s.t. $x \in L$ iff $\exists y. A(x, y) = 1$.

- A is like a machine program. Execution of A on given input x, y goes through a sequence of machine configurations:

  $$c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{T(n)}$$

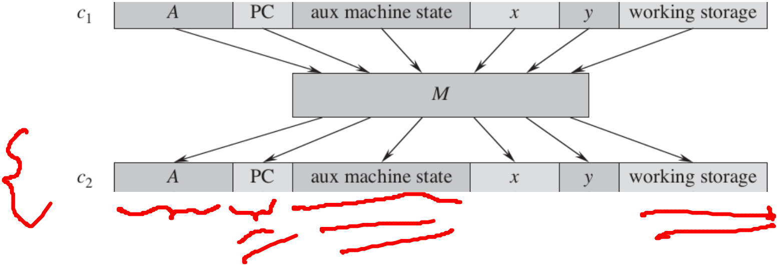
$CIRCUIT\_SAT$ is $\mathbb{NP}$-hard

(Proof Schema) Let $L \in \mathbb{NP}$. Hence, there is $A(x, y) \in \mathbb{P}$ s.t. $x \in L$ iff $\exists y. A(x, y) = 1$.

- A is like a machine program. Execution of A on given input x, y goes through a sequence of machine configurations:

$$c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{T(n)}$$

- Each configuration consists of code of A, program counter, machine registers, inputs x, y and working memory.



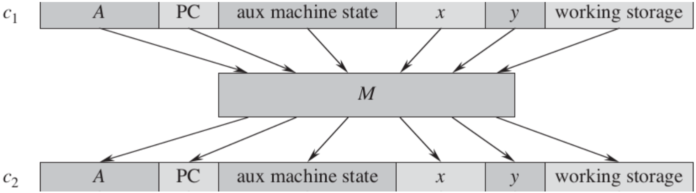
CIRCUIT_SAT is $\mathbb{NP}$-hard

(Proof Schema) Let $L \in \mathbb{NP}$. Hence, there is $A(x, y) \in \mathbb{P}$ s.t. $x \in L$ iff $\exists y.A(x, y) = 1$.

- A is like a machine program. Execution of A on given input x, y goes through a sequence of machine configurations:

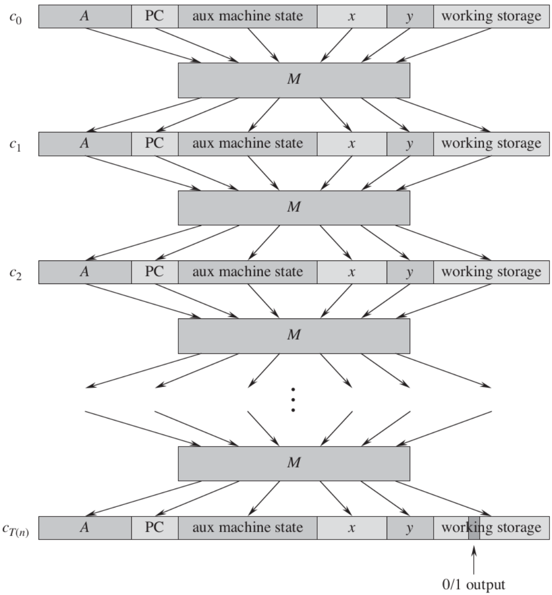
  $$c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{T(n)}$$

- Each configuration consists of code of A, program counter, machine registers, inputs x, y and working memory.



- We have a circuit M which inputs previous configuration and outputs next configuration.

Circuit C_A for $A(x, y)$ taking time $T(n)$



Reduction $f : L \leq_P CIRCUIT\_SAT$

Reduction $f : L \leq_P CIRCUIT\_SAT$

- For $x \in L$ we construct circuit $C_L(y)$ with input wires y such that $x \in L$ iff $\exists y. C_L(y) = 1$.

- For $x \in L$ we construct circuit $C_L(y)$ with input wires y such that $x \in L$ iff $\exists y. C_L(y) = 1$.
- $C_L(y)$ uses 1 copy of C_A.

Reduction $f : L \leq_P CIRCUIT\_SAT$

- For $x \in L$ we construct circuit $C_L(y)$ with input wires y such that $x \in L$ iff $\exists y. C_L(y) = 1$.
- $C_L(y)$ uses 1 copy of C_A.
- It takes y for c_0 as input. It presets $\langle A \rangle, PC = 1, x$ and sets internal state as well as storage to initial value.

Reduction $f : L \leq_P CIRCUIT\_SAT$

- For $x \in L$ we construct circuit $C_L(y)$ with input wires y such that $x \in L$ iff $\exists y. C_L(y) = 1$.
- $C_L(y)$ uses 1 copy of C_A.
- It takes y for c_0 as input. It presets $\langle A \rangle, PC = 1, x$ and sets internal state as well as storage to initial value.
- It outputs the result bit in last configuration as output wire.