Algorithms

Lecture 16: NP Completeness (Part 2)

Anxiao (Andrew) Jiang

CH 34. NP Completeness

We focus on “Decision Problems”.

P: the set of decision problems with this property: for every instance of the problem,
there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

CH 34. NP Completeness

We focus on “Decision Problems”.

P: the set of decision problems with this property: for every instance of the problem,

there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

NP: the set of decision problems with this property: for every instance of the problem,

If the answer is “YES”, there exists some data with which an algorithm can verify that the answer is indeed “YES” in polynomial time.

CH 34. NP Completeness

We focus on “Decision Problems”.

P: the set of decision problems with this property: for every instance of the problem, there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

NP: the set of decision problems with this property: for every instance of the problem,

Called “Certificate”: basically the solution we are looking for

If the answer is “YES”, there exists some data with which an algorithm can verify that the answer is indeed “YES” in polynomial time.

CH 34. NP Completeness

We focus on “Decision Problems”.

P: the set of decision problems with this property: for every instance of the problem, there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

NP: the set of decision problems with this property: for every instance of the problem,

Called “Certificate”: basically the solution we are looking for

If the answer is “YES”, there exists some data with which an algorithm can verify that the answer is indeed “YES” in polynomial time.

Two elements of an algorithm:

1) The process of finding a solution.
2) Verify the correctness of the solution (so it knows when to end).

Sub-case 1: answer is “YES”
Sub-case 2: answer is “NO”

CH 34. NP Completeness

We focus on “Decision Problems”.

P: the set of decision problems with this property: for every instance of the problem, there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

NP: the set of decision problems with this property: for every instance of the problem,

Called “Certificate”: basically the solution we are looking for

If the answer is “YES”, there exists some data with which an algorithm can verify that the answer is indeed “YES” in polynomial time.

Two elements of an algorithm:

1) ~~The process of finding a solution.~~
2) Verify the correctness of the solution (so it knows when to end).

The constraint for NP is weaker:

Sub-case 1: answer is “YES”

~~Sub-case 2: answer is “NO”~~

$$P \subseteq NP$$

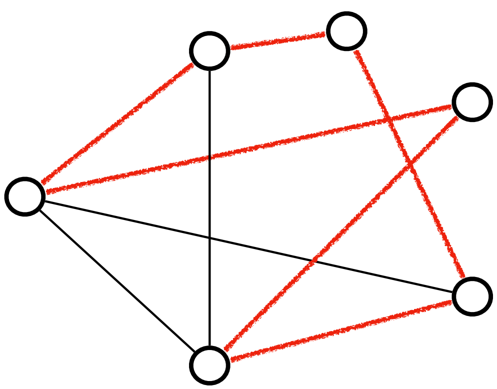
CH 34. NP Completeness

$P \subseteq NP$

Hamiltonian cycle Problem:

Input: A graph G=(V,E).

Question: Does G have a Hamiltonian cycle?



Finding a Hamiltonian cycle might be hard ……

But given a Hamiltonian cycle, it is easy to verify it is indeed A Hamiltonian cycle.

Hamiltonian cycle Problem $\in NP$

CH 34. NP Completeness

P: the set of decision problems with this property: for every instance of the problem, there exists an algorithm that can determine if the answer is YES or NO in polynomial time.

NP: the set of decision problems with this property: for every instance of the problem, If the answer is “YES”, there exists some data with which an algorithm can verify that the answer is indeed “YES” in polynomial time.

Co-NP: the set of decision problems with this property: for every instance of the problem, If the answer is “NO”, there exists some data with which an algorithm can verify that the answer is indeed “NO” in polynomial time.

CH 34. NP Completeness

$P \subseteq NP$

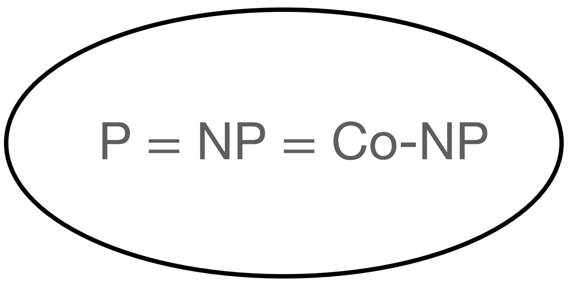
$P \subseteq \text{Co-NP}$

CH 34. NP Completeness

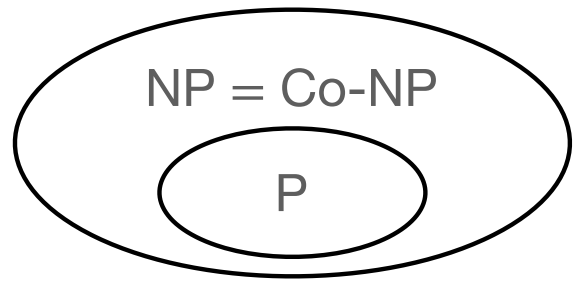
$P \subseteq NP$ $\qquad$ $P \subseteq \text{Co-NP}$

Four possible cases:

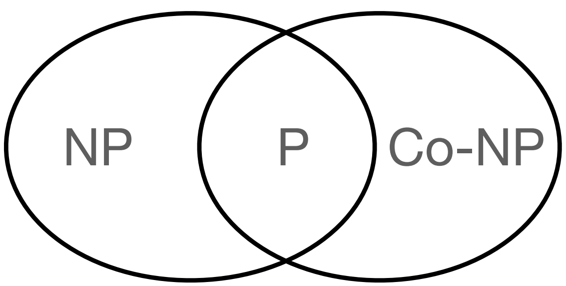
(a)



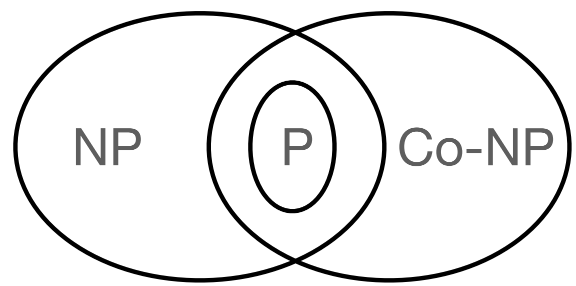
(b)



(c)



(d)



CH 34. NP Completeness

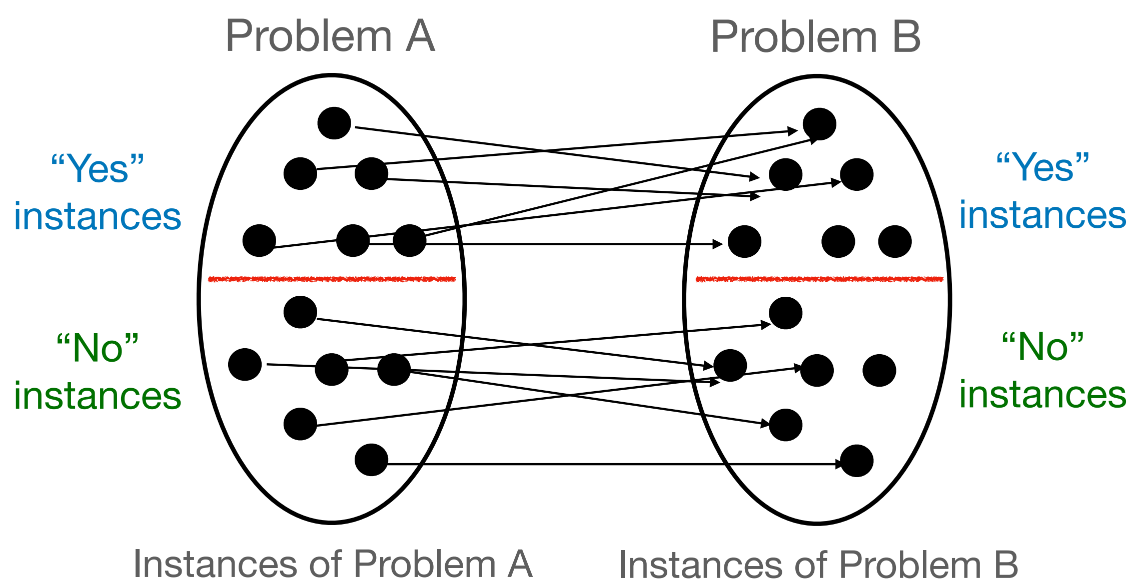
Polynomial-time reduction
from Problem A to Problem B:

$$A \leq_p B$$

Easier Harder

If Problem A is known to be "hard",
then we can use the above relation
to show that Problem B is also "hard".

Polynomial-time reduction:

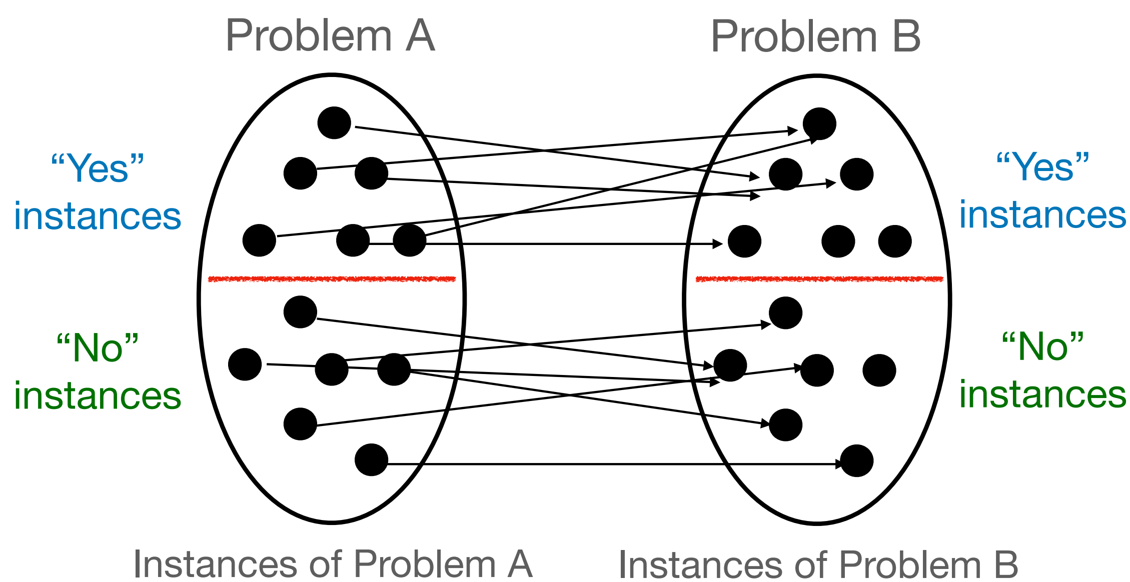


CH 34. NP Completeness

Lemma: If $A \leq_p B$, then $B \in P$ implies $A \in P$.

Proof: We can use the reduction, to solve A through solving B, all in polynomial time.

Polynomial-time reduction:



CH 34. NP Completeness

Lemma: If $A \leq_p B$, then $B \in P$ implies $A \in P$.

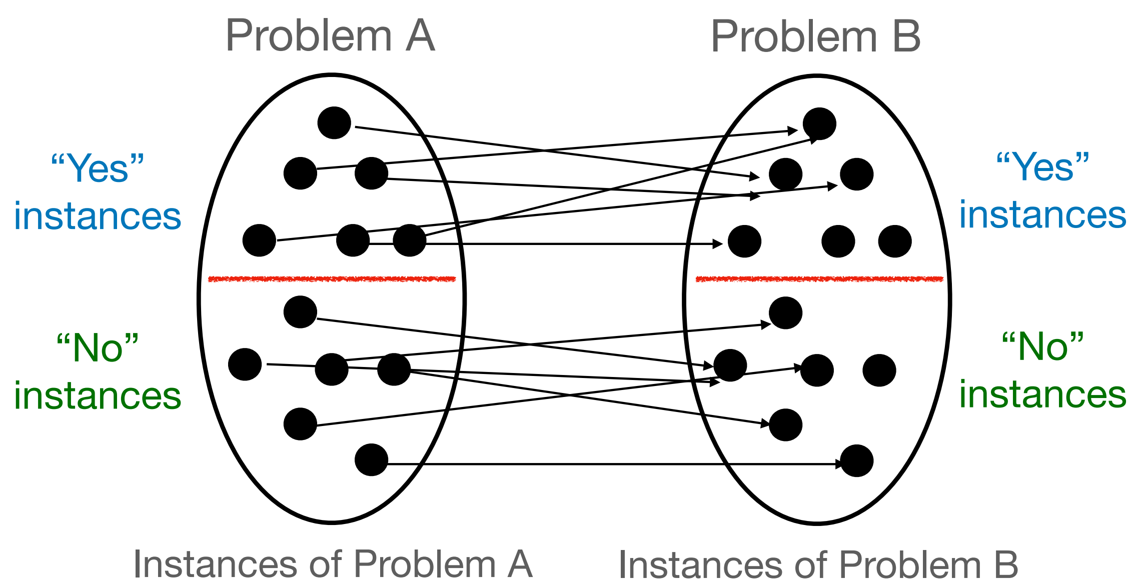
NP-complete definition:

A problem L is NP-complete if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$, we have $A \leq_p L$.

Polynomial-time reduction:



CH 34. NP Completeness

Lemma: If $A \leq_p B$, then $B \in P$ implies $A \in P$.

NP-complete definition:

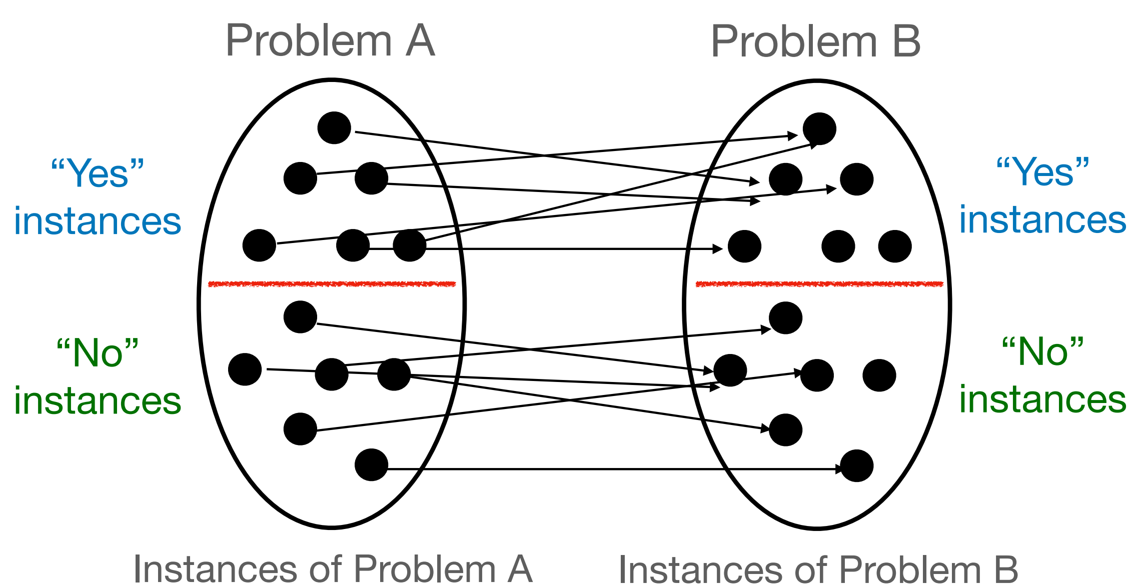
A problem L is NP-complete if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$, we have $A \leq_p L$.

So in a sense, an NP-complete problem is the "hardest" problem in NP, because It is "harder" (or "no easier") than all other problems in NP.

Polynomial-time reduction:



CH 34. NP Completeness

Lemma: If $A \leq_p B$, then $B \in P$ implies $A \in P$.

NP-complete (NPC) definition:

A problem L is NP-complete if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$, we have $A \leq_p L$.

So in a sense, an NP-complete problem is the "hardest" problem in NP, because It is "harder" (or "no easier") than all other problems in NP.

Theorem: If any NP-complete (NPC) problem can be solved in polynomial time, then P=NP.

CH 34. NP Completeness

Lemma: If $A \leq_p B$, then $B \in P$ implies $A \in P$.

NP-complete (NPC) definition:

A problem L is NP-complete if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$, we have $A \leq_p L$.

So in a sense, an NP-complete problem is the "hardest" problem in NP, because It is "harder" (or "no easier") than all other problems in NP.

Theorem: If any NP-complete (NPC) problem can be solved in polynomial time, then P=NP.

If $P \neq NP$, then no NPC problem can be solved in polynomial time.

CH 34. NP Completeness

NP-complete (NPC) definition:

A problem L is NP-complete
if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$,

we have $A \leq_p L$.

If only condition 2 is satisfied,
L is called "NP-hard".

CH 34. NP Completeness

How to prove a problem L is NP-complete ?

NP-complete (NPC) definition:

A problem L is NP-complete
if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$,

we have $A \leq_p L$.

CH 34. NP Completeness

NP-complete (NPC) definition:

A problem L is NP-complete
if it has two properties:

1) $L \in NP$

2) For every problem $A \in NP$,

we have $A \leq_p L$.

How to prove a problem L is NP-complete ?

Lemma: For any problem $A \in NPC$,

if $A \leq_p L$, then L is NP-hard;

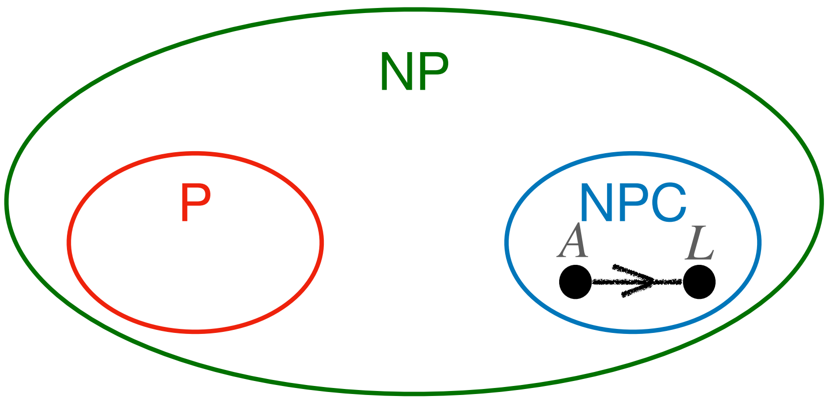
if $A \leq_p L$ and $L \in NP$, then $L \in NPC$.

CH 34. NP Completeness

Lemma: For any problem $A \in NPC$, if $A \leq_p L$, then L is NP-hard;

if $A \leq_p L$ and $L \in NP$, then $L \in NPC$.

Proof: What we have:



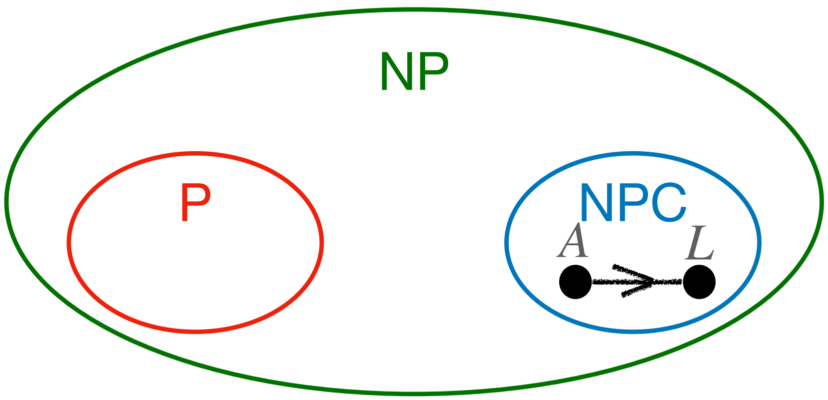
NP-complete (NPC) definition:
A problem L is NP-complete if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$, we have $A \leq_p L$.

CH 34. NP Completeness

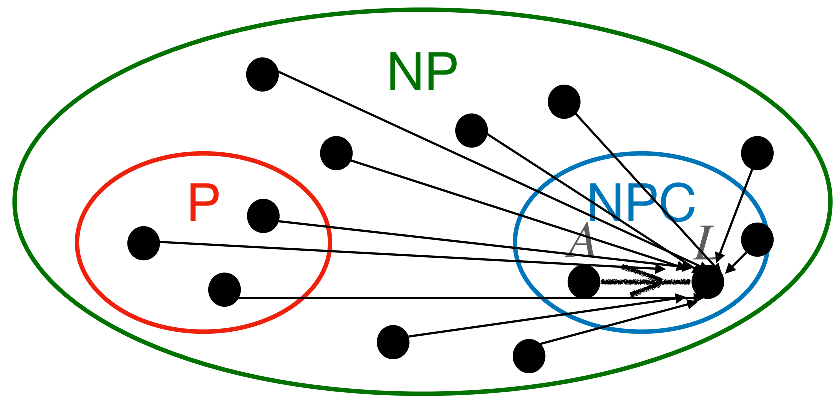
Lemma: For any problem $A \in NPC$, if $A \le_p L$, then L is NP-hard;

if $A \le_p L$ and $L \in NP$, then $L \in NPC$.

Proof: What we have:



What we want:



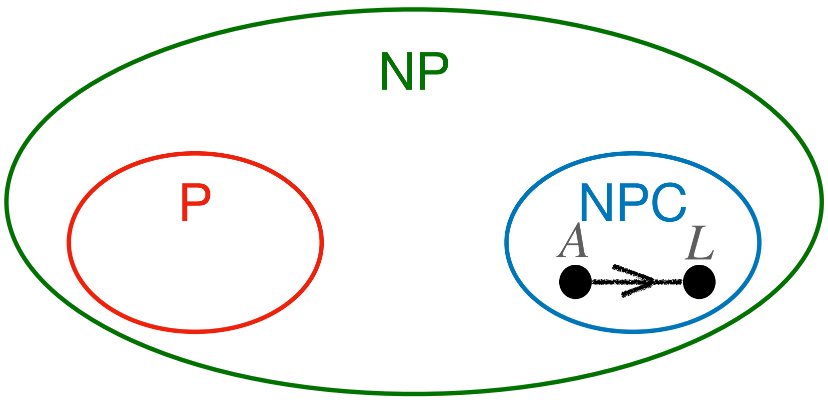
NP-complete (NPC) definition:
A problem L is NP-complete if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$, we have $A \le_p L$.

CH 34. NP Completeness

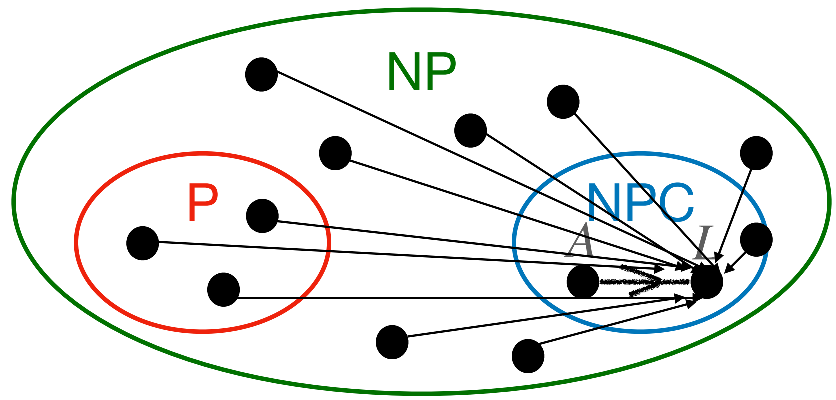
Lemma: For any problem $A \in NPC$, if $A \le_p L$, then L is NP-hard;

if $A \le_p L$ and $L \in NP$, then $L \in NPC$.

Proof: What we have:

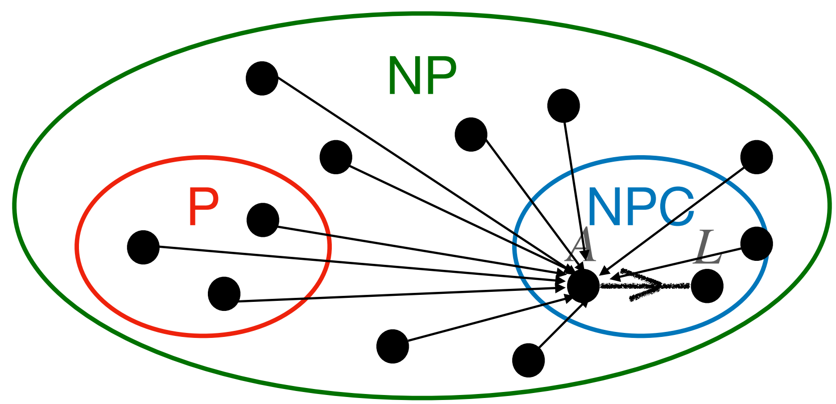


What we want:



NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$, we have $A \le_p L$.

What we do:



CH 34. NP Completeness

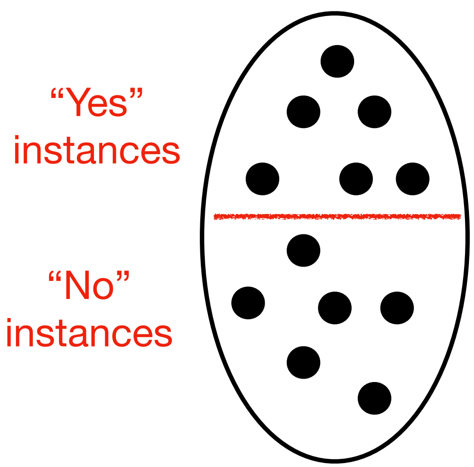
Lemma: For any problem $A \in NPC$, if $A \leq_p L$, then L is NP-hard;

if $A \leq_p L$ and $L \in NP$, then $L \in NPC$.

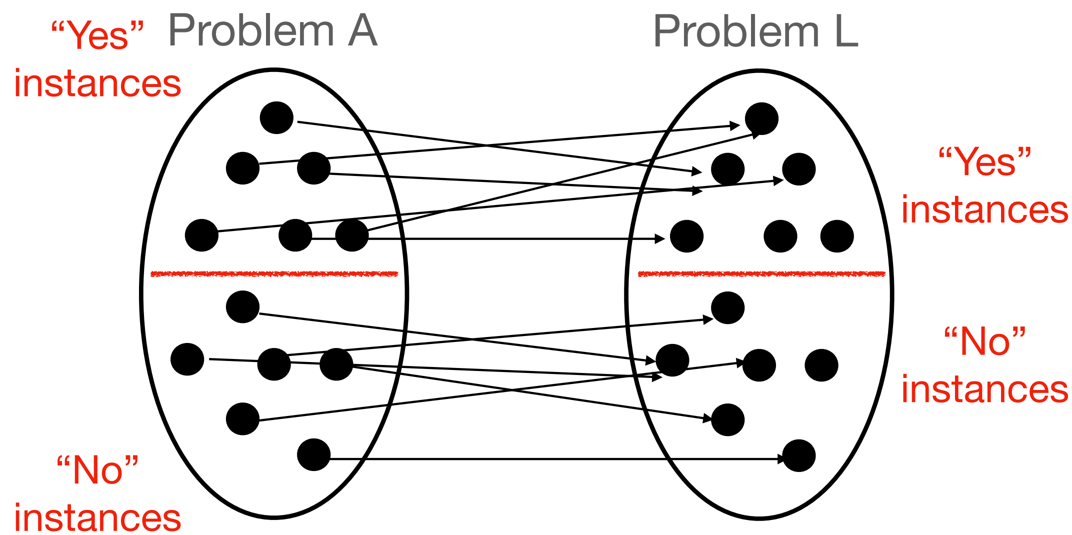
Proof:

Polynomial-time reduction
From A to L:

Any Problem B $\in NP$



Instances of Problem B



Instances of Problem A

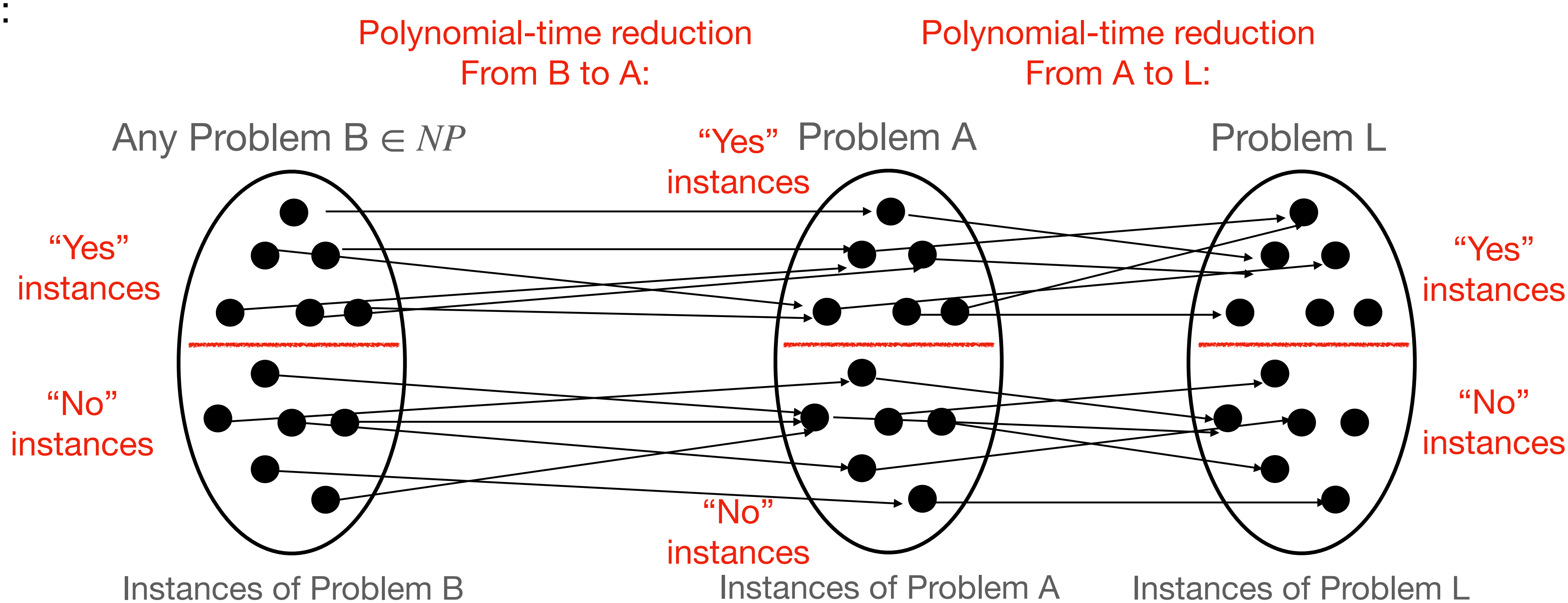
Instances of Problem L

CH 34. NP Completeness

Lemma: For any problem $A \in NPC$, if $A \leq_p L$, then L is NP-hard;

if $A \leq_p L$ and $L \in NP$, then $L \in NPC$.

Proof:



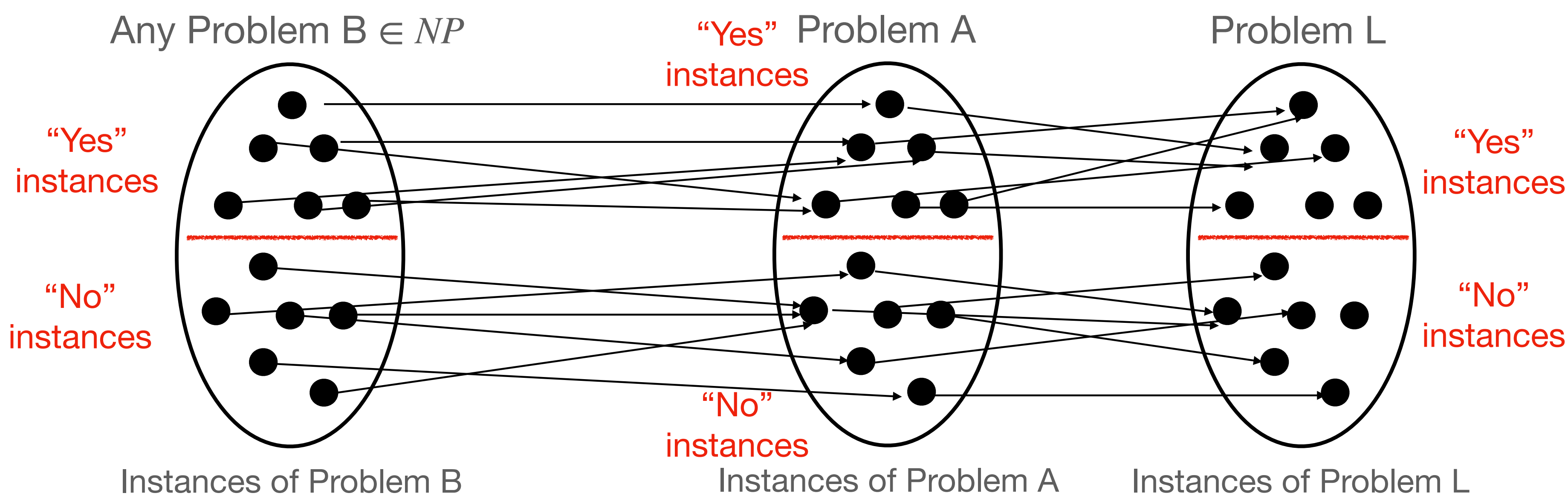
CH 34. NP Completeness

Lemma: For any problem $A \in NPC$, if $A \leq_p L$, then L is NP-hard;

if $A \leq_p L$ and $L \in NP$, then $L \in NPC$.

Proof:

Polynomial-time reduction from B to L:



CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC) ?

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a "certificate" and polynomial-time verification for YES-instances).

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a “certificate” and polynomial-time verification for YES-instances).

2) Pick a known NPC problem A and show $A \leq_p L$

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \le_p L$.

Lemma: For any problem $A \in NPC$,
if $A \le_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a "certificate" and polynomial-time verification for YES-instances).

2) Pick a known NPC problem A and show $A \le_p L$

2.1) Show mapping from A to L

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a "certificate" and polynomial-time verification for YES-instances).

2) Pick a known NPC problem A and show $A \leq_p L$

2.1) Show mapping from A to L

2.2) Show the mapping preserves the "YES/NO" answer

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a "certificate" and polynomial-time verification for YES-instances).

2) Pick a known NPC problem A and show $A \leq_p L$

  2.1) Show mapping from A to L

  2.2) Show the mapping preserves the "YES/NO" answer

  2.3) Show the mapping takes polynomial time

CH 34. NP Completeness

NP-complete (NPC) definition:
A problem L is NP-complete
if it has two properties:
1) $L \in NP$
2) For every problem $A \in NP$,
we have $A \leq_p L$.

Lemma: For any problem $A \in NPC$,
if $A \leq_p L$ and $L \in NP$,
then $L \in NPC$.

Standard technique for proving NP-completeness today

How to prove a problem L is NP-complete (NPC):

1) Show that $L \in NP$ (by showing a "certificate" and polynomial-time verification for YES-instances).

2) Pick a known NPC problem A and show $A \leq_p L$

 2.1) Show mapping from A to L

 2.2) Show the mapping preserves the "YES/NO" answer

 2.3) Show the mapping takes polynomial time