Maynooth University (National University of Ireland, Maynooth)

Department of Computer Science

T. Naughton

CS605 : NP-completeness intro 2

March 2020

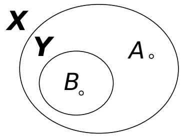
Recap: Reductions

- A polynomial transformation/reduction

  $B \leq A$

  means we can say that B is *no harder* than A

  (i.e. A is as hard or harder than B), and that B is in the same class as A.



$B \in \mathbf{Y}$

$B \in \mathbf{X}$

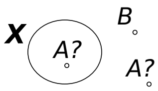
$A \in \mathbf{X}$

Recap: Reductions

- A reduction $B \leq A$ establishes the fact that if A is 'easy' then B is 'easy',

  and if B is 'hard' then A must be 'hard'.

- How to prove nonmembership of a class using a reduction: suppose we know that there is a problem B not in class X . By finding a reduction from B to A we can prove that A is not in X either.

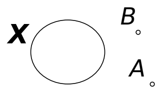


Recap: Reductions

- A reduction $B \leq A$ establishes the fact that if A is 'easy' then B is 'easy',

  and if B is 'hard' then A must be 'hard'.

- How to prove nonmembership of a class using a reduction: suppose we know that there is a problem B not in class X . By finding a reduction from B to A we can prove that A is not in X either.



Classes and polynomial reducibility

- If a language is in P then every language that is polynomially reducible to it is also in P.

- If a language is in NP then every language that is polynomially reducible to it is also in NP.

Completeness

<u>Definition</u> A, B are languages, X is a class of languages.

If $B \leq_m^P A$ for every $B \in X$, we say that

A is *X-hard* (*$\leq_m^P$-hard for X*).

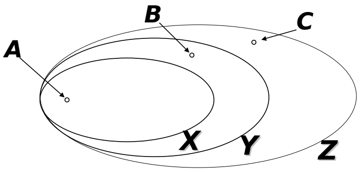
So, if A is X-hard then it is at least as hard as every problem in X.

Completeness

If A is *X-hard* and $A \in X$ then we say that

A is *X-complete* (*$\le_m^P$-complete for X*).

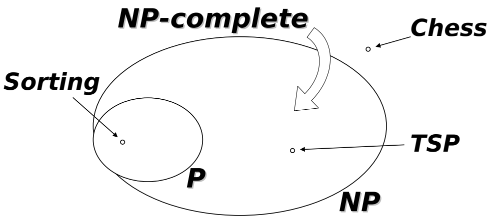
Completeness



- $A \in X$ and A is X-complete.
- $B \in Y$ and B is X-hard and Y-complete.
- $C \in Z$ and C is X-hard, Y-hard and Z-complete.

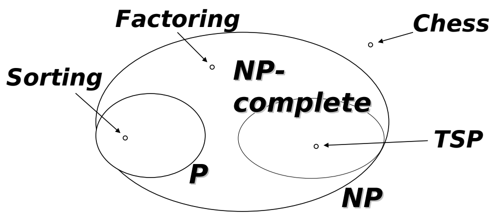
Completeness

- The NP-complete languages are therefore typical of the “hardest” languages in NP.



Completeness

- The NP-complete languages are therefore typical of the “hardest” languages in NP.



Proving NP-completeness

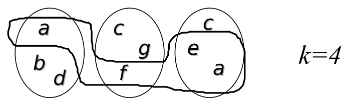
To prove that decision problem A is NP-complete we must

- Show that A is in NP
- Select a known NP-complete problem X
- Construct a transformation f from X to A
- Prove that f is polynomial

Hitting set

<u>Hitting set</u>

We are given a system $\{A_1, \ldots , A_m\}$ of finite sets and a natural number k . Is there a set with no more than k elements that intersects each A_i ?



The hitting set problem is NP-complete.

Hitting set

Proof sketch We reduce 3-SAT to this problem (therefore proving that this problem is at least as hard as 3-SAT).

For a given conjunctive 3-normal form *B* we regard the literals (boolean variables) in *B* and their complements as being separate symbols,

and construct a system of sets as follows:

Hitting set

- For each clause of *B* , we turn it into a set containing each of the literals and complemented literals occurring in it, and
- for each of the *n* literals x_i in *B*, create the set $\{x_i, \overline{x_i}\}$.

Hitting set

- The sets of this set system can be hit with at most *n* elements iff the normal form is satisfiable.
- Let's take an example...