Monash University
Faculty of Information Technology

Lecture 21
Polynomial-time reductions

Slides by Graham Farr (2012)

FIT2014 Theory of Computation

Overview

- Comparing languages
- Definition of polynomial time reduction
- Examples
- Properties

Polynomial-time reductions

- Some languages are easier to decide than others.
- How to compare languages?
- Could we use time complexity?
  - seldom known exactly; usually we just know an upper bound (e.g., big-O)

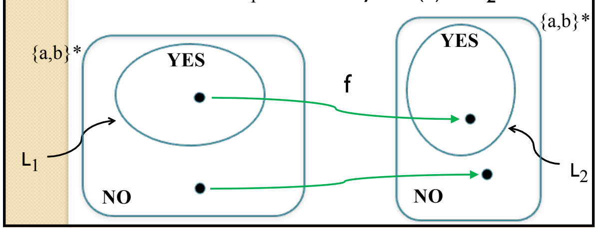
Polynomial-time reductions

A *polynomial-time reduction* from L_1 to L_2 is a polynomial-time computable function

$$f : \{a,b\}^* \rightarrow \{a,b\}^*$$

such that, for all x in $\{a,b\}^*$,

$$x \in L_1 \quad \text{if and only if} \quad f(x) \in L_2$$



Polynomial-time reductions

- Polynomial-time reductions are also called:
  - polynomial-time mapping reductions
  - polynomial-time many-one reductions
  - polynomial transformations
  - Karp reductions
- If there is a polynomial-time reduction from L_1 to L_2, then we write $L_1 \leq_P L_2$.

Examples

INDEPENDENT SET $\leq_P$ CLIQUE

Complement $\overline{G}$ of G: edges $\leftrightarrow$ non-edges

Independent sets in G correspond to cliques in $\overline{G}$.

G has an independent set of size $\geq k$ if and only if $\overline{G}$ has a clique of size $\geq k$.

So: $(G, k) \in$ INDEPENDENT SET if and only if $(\overline{G}, k) \in$ CLIQUE.

Construction of $(\overline{G}, k)$ from (G, k) is polynomial time.

So $(G, k) \mapsto (\overline{G}, k)$ is a polynomial-time reduction from INDEPENDENT SET to CLIQUE.

Examples

VERTEX COVER $\leq_P$ INDEPENDENT SET

If G is a graph and X is a subset of V(G), then X is a vertex cover of G if and only if V(G) \ X is an independent set of G.

So: $(G, k) \in$ VERTEX COVER if and only if

$(G, n-k) \in$ INDEPENDENT SET.

The construction is polynomial time.

So the function $(G, k) \mapsto (G, n-k)$ is a polynomial-time reduction.

Examples

2-SAT $\leq_P$ 3-SAT

Given a Boolean formula ϕ in CNF with 2 literals per clause, how can we transform it to another Boolean formula ϕ' in CNF with 3 literals/clause, such that ϕ is satisfiable if and only if ϕ' is satisfiable?

For each i: Suppose i-th clause in ϕ is $x \vee y$.

Create a new variable w_i which appears nowhere else. Replace clause $x \vee y$ by two clauses:

$(x \vee y \vee w_i) \wedge (x \vee y \vee \neg w_i)$

Examples

Then show that

- this construction takes polynomial time
- ϕ is satisfiable if and only if ϕ' is satisfiable

Examples

SUBGRAPH ISOMORPHISM

{ (G, H) : G is isomorphic to a subgraph of H }

GRAPH ISOMORPHISM $\leq_P$ SUBGRAPH ISOMORPHISM

$(G, H) \mapsto (G, H)$

Polynomial time!

Does it work the other way round?

PARTITION

$$\left\{ (s_1, s_2, \ldots, s_n) : \text{for some subset } J \text{ of } \{1,2,\ldots,n\}, \sum_{i \in J} s_i = \sum_{i \in \{1,\ldots,n\} \setminus J} s_i \right\}$$

SUBSET SUM

$$\left\{ (s_1, s_2, \ldots, s_n, t) : \text{for some subset } J \text{ of } \{1,2,\ldots,n\}, \sum_{i \in J} s_i = t \right\}$$

PARTITION $\leq_P$ SUBSET SUM

$(s_1, s_2, \ldots, s_n) \mapsto (s_1, s_2, \ldots, s_n, (s_1 + s_2 + \ldots + s_n)/2)$

Can you show SUBSET SUM $\leq_P$ PARTITION ?

Examples

Others to try:

3-COLOURABILITY $\leq_P$ GRAPH COLOURING

where

GRAPH COLOURING = { (G,k) : G is k-colourable }

2-COLOURABILITY $\leq_P$ 3-COLOURABILITY

HAMILTONIAN CIRCUIT $\leq_P$ HAMILTONIAN PATH

2-COLOURABILITY $\leq_P$ 2-SAT

SATISFIABILITY $\leq_P$ 3-SAT

3-COLOURABILITY $\leq_P$ SATISFIABILITY

Properties

Reflexive: For any L, $L \leq_P L$.

Transitive: If $L_1 \leq_P L_2$ and $L_2 \leq_P L_3$ then $L_1 \leq_P L_3$.

Properties

Theorem

If $L_1 \leq_P L_2$ and L_2 is in P, then L_1 is in P.

Proof.

Let f be a polynomial-time reduction from L_1 to L_2, and let D be a poly-time decider for L_2.

Decider for L_1:

Input: x

Use f to construct f(x), and D to decide whether or not f(x) is in L_2.

Accept x if and only if D accepts f(x).

Is it polynomial time?

Properties

If f has time complexity $O(n^k)$, then the length of its output string f(x) must also be $O(n^k)$, since a TM can, in t steps, output no more than t symbols.

The decider D runs in polynomial time, so suppose it has time complexity $O(n^{k'})$, where n is the size of the input to D.

If D is given f(x) as input, then the time D takes on it is $O(|f(x)|^{k'})$, where $|f(x)|$ = length of string f(x).

Since $|f(x)| = O(n^k)$, we find that D takes time $O(n^{kk'})$, where $n = |x|$.

Properties

Total time taken by our decider for L_1 is

time taken by f on x + time taken by D on f(x)

$= O(n^k) + O(n^{kk'}) = O(n^{kk'})$, which is polynomial time.

End of proof

Properties

Exercises

Prove: if L_1 is in P and L_2 is any language, then $L_1 \leq_P L_2$.

The fine print: some caveats regarding trivial cases are needed here. What are they?

Prove:

Theorem

If $L_1 \leq_P L_2$ and L_2 is in NP, then L_1 is in NP.

Revision

- Sipser, section 7.4, pp299-303.