Teoria da Computação
2018/2

Álvaro Moreira afmoreira@inf.ufrgs.br

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
http://www.inf.ufrgs.br

Contents

NP-Completeness (I)

- According to the definition, a language L is $\mathsf{NP-Complete}$ if
  - L is in NP, and
  - L is $\mathsf{NP-Hard}$
- Stephen Cook and Leonid Levin independently were the first to prove that a problem is NP-Complete (the problem SAT)

  Theorem: (Cook-Levin) SAT is NP-Complete.
- They did that by proving that

  (i) SAT is a problem in NP

  (ii) SAT is an $\mathsf{NP-Hard}$ problem

NP-Completeness (II)

- Later, Richard **Karp** showed that many problems of practical interest are NP-Complete problems
- To prove that a language L is NP-Complete, Karp didn't use the approach used by Cook and Levin, but instead he
  (i) proved that L is in NP and
  (ii) picked a language L' **already known to be NP-Complete** and showed that $L' \leq_p L$

Exercise: Reflect on why the strategy used by Karp works for proving that a language L is NP-Complete

The Cook-Levin Theorem

- Karp's work popularized the notion of NP-Completeness
- By using his strategy, thousands of other problems have been proved to be to NP-Complete

  Hence, the fundamental importance of the work of Cook and Levin since they established the **first** NP-Complete problem (SAT)
- The problem SAT could then be used in Karp's reductions
- Before knowing more about the proof of the Cook-Levin Theorem lets take a look at the problems SAT and 3SAT and to some other $\mathrm{NP}-\mathrm{Complete}$ problems

The SAT Problem (I)

- A Boolean formula consists of variables $x_1, \dots x_n$ and the logical operators AND ($\wedge$), NOT ($\neg$) and OR ($\vee$)

- if φ is a Boolean formula over variables $x_1, \dots x_n$, and $z \in \{0,1\}^n$, then $\varphi(z)$ denotes the value of φ when the variables of φ are assigned the values in z (1 is true and 0 is false)

- A formula φ is **satisfiable** if there there exists some assignment z such that $\varphi(z)$ is true, otherwise we say that it is **unsatisfiable**.

- The formula $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_1)$ with 3 variables, for instance, is satisfiable since the assignment $x_1 = 1$, $x_2 = 0$, and $x_3 = 1$ satisfies it

The SAT Problem (II)

- The SAT language consists of all satisfiable Boolean formulas

$$\text{SAT} = \{\langle \varphi \rangle \mid \varphi \text{ is a satisfiable Boolean formula}\}$$

- Any Boolean formula can be put in the **CNF** form (shorthand for Conjunctive Normal Form)

- A formula is in CNF if it is a conjunction of clauses, each clause is a disjunction of literals (variables and negation of variables). Example: $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4})$

- The 3SAT language consists of all satisfiable formula in the **3CNF** form (CNF form in which all clauses have at most 3 literals)

$$\text{3SAT} = \{\langle \varphi \rangle \mid \varphi \text{ is a satisfiable 3CNF Boolean formula}\}$$

Proving that SAT is NP-Complete

- We will see later the proof that SAT is NP-Complete (Cook-Levin Theorem)
- The most difficult part of this proof is to show that SAT is NP-hard, i.e, to prove that all problems in NP can be reduced to SAT
- The proof that SAT is in NP is easy

Exercise: Prove that SAT is in NP

Exercise: Assuming the fact that SAT has already been proved to be NP-Complete explain what could be a strategy to prove that 3SAT is NP-Complete

3SAT is NP-complete

To prove that 3SAT is NP-Complete we have to prove that (i) 3SAT is in NP and (ii) that 3SAT is NP-Hard

Exercise: Prove that $\mathrm{3SAT}$ is in NP

We prove the $\mathsf{NP-hardness}$ of 3SAT using Karp strategy: by reducing a language L we already know to be NP-Complete to 3SAT

Lemma: SAT $\leq_{\mathsf{P}}$ 3SAT.

Proof idea. We present a transformation that maps, in polynomial time, any CNF formula φ into a 3CNF formula ψ. The transformation should be such that $\phi \in \mathsf{SAT}$ iff $\psi \in \mathsf{3SAT}$ (i.e CNF formula φ is satisfiable if and only if 3CNF formula ψ is.

A CNF formula φ is a conjunction of clauses $\mathsf{C}_1 \wedge \mathsf{C}_2 \ldots \mathsf{C}_n$ where $\mathsf{n} \geq 1$.

Each clause is a disjunction of literals with all variables distinct (a literal is a variable or is negation)

Reducing SAT to 3SAT (II)

Let C be a clause of a 4CNF formula φ, say $C = x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4$.

We add a new variable z to the formula φ and replace C with the conjunction of clauses $C_1 = x_1 \vee \overline{x_2} \vee z$ and $C_2 = \overline{x_3} \vee x_4 \vee \bar{z}$.

If C is **true** then there is an assignment to z that satisfies both C_1 and C_2

If C is **false** then no matter what value we assign to z either C_1 or C_2 will be false

By applying this transformation to all clauses of a 4CNF formula φ we obtain a 3CNF formula ψ such that

$$\varphi \quad \text{is satisfiable} \quad \text{iff} \quad \psi \quad \text{is satisfiable}$$

Reducing SAT to 3SAT (III)

The same idea applied before to a 4CNF formula can be used to transform any kCNF formula into an equivalent 3CNF formula

By adding a new variable (z) , change every clause of $k > 3$ literals into an equivalent pair of clauses:

1. a clause C_1 of $k - 1$ literals, one of them being z, and
2. a clause C_2 of 3 literals, one of them being $\bar{z}$,

Applying this transformation repeatedly to the clause resulting from 1 above (with a new variable) yields a **polynomial-time** transformation of a CNF formula φ into an equivalent 3CNF formula ψ $\square$

This was just a *proof idea* of how to prove that $\mathsf{SAT} \leq_p \mathsf{3SAT}$.

Exercise: Study a detailed proof that $\mathsf{SAT} \leq_p \mathsf{3SAT}$

Why SAT and 3SAT?

- Cook and Levin were interested in mathematical logic where satisfiability propositional logic has a central role (the title of Cook' paper from 1971 is *"The complexity of theorem-proving procedures"*)

- 3SAT is convenient for proving that other problems are NP-Complete (it has small combinatorial structure and that makes easy to use in reductions)

- It has a practical importance by itself: it is a simple example of *constraint satisfaction problems* which appear in many fields of CS

Independent Set is NP-Complete (I)

- Recall that

$$L_{\text{INDSET}} = \{(G, k) \mid \exists S \subseteq V(G) \text{ s.t. } |S| \geq k \;\wedge\; \forall u, v \in S, \overline{uv} \notin E(G)\}$$

- In previous classes we saw that L_{INDSET} is in NP
- We still have to show that L_{INDSET} is NP-Hard.
- For doing that we show that

$$\text{3SAT} \leq_{\mathsf{P}} \text{INDSET}$$

Independent Set is NP-Complete (II)

We have to define

- a polynomial-time transformation f of strings representing 3CNF formulas φ to strings representing pairs (G, k) (i.e $f(\varphi) = (G, k)$)
- the transformation should be such that

$$\varphi \in L_{\mathrm{3SAT}} \quad \text{iff} \quad f(\varphi) \in L_{\mathrm{INDSET}}$$

Independent Set is NP-Complete (II)

The key to understand the reduction is to consider the following way to approach the 3SAT problem:

- choose one literal from each of the k clauses of a formula $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$
- if it is possible to make a choice such that all the k literals chosen have value 1 then the formula φ is satisfiable
- for all the literals to have value 1 they cannot be *conflicting*
- two literals are *conflicting* if one is a variable and the other is the negation of this same variable

Independent Set is NP-Complete (II)

The graph $G = (V, E)$ is constructed from φ as follows:

- for each clause $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ we construct 3 vertices v_{i1}, v_{i2} and v_{i3}
- we add edges between each one of these 3 vertices v_{i1}, v_{i2}, and v_{i3}
- we label the vertices v_{i1}, v_{i2} and v_{i3} with the literals l_{i1}, l_{i2}, and l_{i3} of clause C_i, respectively
- note that we now have k triangles where every two vertices of each triangle are connected by an edge

Independent Set is NP-Complete (III)

- Before proceeding with construction of the graph G observe that 2 vertices cannot be selected from the same triangle to form an independent set
- Hence, for an independent set S of size k to exist each one of k triangles should be able to *contribute* to S with a vertice
- This is in agreement to the view discussed before of being able to choose a literal form each clause such that all of then have value 1
- But how to avoid choosing conflicting literals?

Independent Set is NP-Complete (IV)

- To avoid choosing conflicting literals we conclude the construction of the graph G by adding an edge between any two nodes of different triangles which are labeled with conflicting literals
- after this last step of the construction of graph G we observe that:
  - if an independent set of size k still exist in G that means that the formula G is satisfiable, also
  - if the formula φ is satisfiable then the graph G constructed as above has an independent set S of k nodes

Independent Set is NP-Complete (V)

- The transformation f of a 3CNF formula φ to and input (G, k) for the TM that solves INDSET can be done in polynomial time, and
- We have shown that:

$$\varphi \in L_{3SAT} \quad \text{iff} \quad f(\varphi) \in L_{INDSET}$$

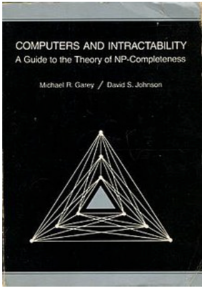
- Hence, we have that $3SAT \leq_P INDSET$ $\square$

This proof is in Section 8.2 of Kleinberg and Tardos book

Exercise: Study the proof in Sipser's book that CLIQUE is NP – Complete

More about NP-Complete Problems

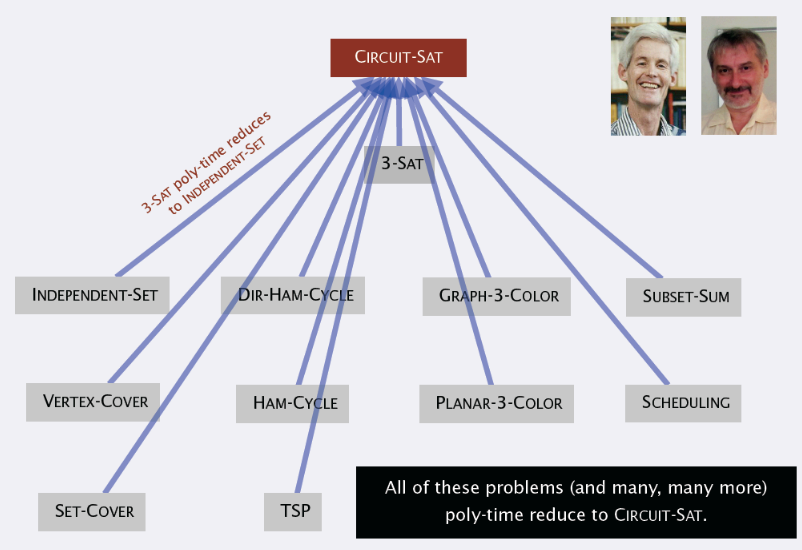
The book by Garey and Johnson, a classic, has many NP-Complete problems



Chapter 8 o "NP and Computational Intractability"of Kleinberg and Tardos book is also a great source with a focus on reductions

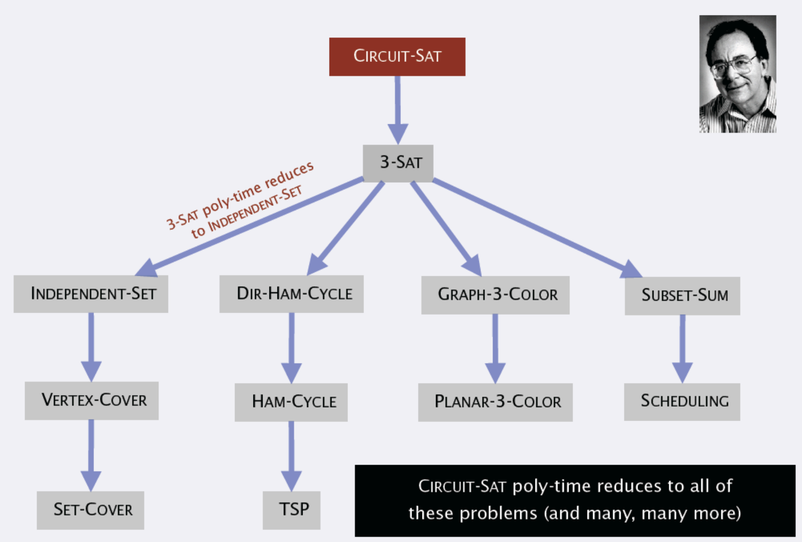
The web of reductions - Cook-Levin result

Theorem: (Cook-Levin) SAT is NP-Complete



The web of reductions - Karp reductions

Karp proved that many important problems are NP-Complete



The web of reductions

NP-Complete problems are inter-reducible.