(For CS302) Reductions to Propositional Satisfiability

Somenath Biswas
CSE, IITK

Sem I, 2012-13

1 Notion of reduction

Although the problem of deciding, given a formula, whether or not it is satisfiable, is important in its own rights, [1] what makes the satisfiability problem more important is that many problems *reduce* to satisfiability. Informally, a problem Π_1 reduces to another problem Π_2, if by solving Π_2, we get the answer for the problem Π_1. If we pause to think about it, we can come up with many examples of such reductions in all kinds of domains. In what we have seen of logic, there is already one example– the problem of testing for tautology-ness reduces to testing for satisfiability (and also, the other way around). We see below two examples of two very different looking problems both reducing to satisfiability.

2 Pigeon-hole principle

Pigeon-hole principle states that if there are m pigeons to go into n pigeon-holes, $m > n$, then there will be at least one pigeon-hole with more than one pigeon in it. More generally, for any function f with domain D and range R, both finite, and $|D| > |R|$, at least $\lceil D/R \rceil$ elements of the domain get mapped by f to the same element of the range. An even more general statement can be made in terms of random variables over an event space: if X is a random variable taking integral values, with μ_X denoting its expected value, then there exists an event where $X \geq \lceil \mu_X \rceil$ and there exists an event where $X \leq \lfloor \mu_X \rfloor$.

[1] Recall that it gives us a way of identifying what logical truths are in the context of propositional logic

Pigeon-hole principle is an immensely useful tool which surfaces in all kinds of combinatorial reasoning. Let us consider here the problem of proving the first version of the principle. Although we cannot use propositional logic to prove the principle as stated, we show here how we can code specific instances of the principle into propositional logic, so that the proof for that instance turns out to be equivalent to showing that a certain formula is not satisfiable.

Let us consider the case when there are 4 pigeons and three pigeon-holes. For coding this instance into the language of propositional logic, we shall use 12 propositional variables, denoted as:

$$p_{ij}, 1 \leq i \leq 4, 1 \leq j \leq 3$$

The intended semantics of p_{ij} is that it is true if the ith pigeon goes into the jth hole, else it is false.
Consider the formula: $(p_{11} \vee p_{12} \vee p_{13})$.
With the semantics of the propositional variables as above, this formula expresses that *the pigeon 1 goes into some hole.* Let us call this formula ψ_1. In a similar manner we define ψ_2 to express that *pigeon 2 goes into some hole,*, and ψ_3, ψ_4 to to express the same condition on pigeon 3 and pigeon 4 respectively. The formula Ψ defined as $(\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4)$ expresses the statement that *every pigeon goes in to some pigeon-hole.*

Next, consider the formula:

$$(p_{11} \Rightarrow (\neg p_{21} \wedge \neg p_{31} \wedge \neg p41))$$

It is easy to see that the above formula expresses that *if pigeon 1 goes into hole 1 then no other pigeon goes into hole 1.* Let us call this formula $\phi_{1,1}$. Similarly, we can define $\phi_{2,1}$ to express that if pigeon 2 goes into pigeon-hole 1, then no other pigeon can go into pigeon-hole 1. Similarly, define $\phi_{3,1}$ and $\phi_{4,1}$. Let us define Φ_1 as the conjunction of these formulas. Clearly, Φ_1 expresses that *the pigeon-hole 1 can have at most one pigeon.* Let us define Φ_2 and Φ_3 to express the condition on pigeon-hole 2 and pigeon-hole 3 respectively. Now, consider the formula Γ where

$$\Gamma \stackrel{\text{def}}{=} (\Psi \wedge \Phi_1 \wedge \Phi_2 \wedge \Phi_3)$$

We can make the following claim:

Claim 1 Γ *is satisfiable iff there is a way in which 4 pigeons can go into 3 pigeon-holes with each pigeon-hole getting at most one pigeon. In fact, every*

truth assignment to the propositional variables p_{ij}s under which Γ evaluates to true, is way of assigning 4 pigeons into 3 holes with no hole getting more than one pigeon.

Therefore, a proof that Γ is not satisfiable proves the 4 pigeons, 3 holes instance of the pigeon-hole principle.

3 Graph colouring

In the last section the problem of proving a statement was reduced to proving a certain formula was unsatisfiable. Here we consider a different kind of reduction– reduction of one computational problem to another. Suppose we want to find an algorithm for the computational problem Π_1. Such an algorithm, given any instance x (i.e., an input), will find the corresponding output. We say that Π_1 reduces to Π_2 if we can obtain an algorithm for Π_1 using any algorithm for Π_2. In this section we show the reduction of a computational problem on graphs to the satisfiability problem.

Definition 2 An undirected graph is said to be k–colourable if one can colour the vertices of the graph using no more than k colours such that no two adjacent vertices get the same colour.

The graph problem we consider is to give an algorithm which, given a graph G and k as input, tells us if the input graph is k-colourable or not. There are many situations where this algorithm will be useful. For example, suppose we have a number of courses and k time slots. We can find out whether or not the courses can be scheduled in these time slots without any conflict from this algorithm. (A conflict arises when two courses with some students in common, or with the same instructor, are both given the same time slot.) Build a graph with a vertex for each course, join two vertices with an edge if the corresponding two courses either have students in common, or have the same instructor. It can be easily seen that the resultant graph is k–colourable iff the courses can be scheduled in k time slots.[2]

For simplicity of exposition, let us work by fixing k to 3. What we do now is to show, given any graph G, how to obtain a formula Ψ_G such that G is 3-colourable iff Ψ_G is satisfiable.

[2]Incidentally, this is another example of reduction– the scheduling problem reducing to the graph colouring problem.

For a graph G with n vertices, the formula Ψ_G is built using $3n$ propositional variables, viz.,

$$b_1, b_2, \ldots, b_n, \quad r_1, r_2, \ldots, r_n, \quad y_1, y_2, \ldots, y_n$$

The intended semantics of b_i is that the ith vertex is coloured blue, of r_i is that the ith vertex is coloured red, and of y_i is that the ith vertex is coloured yellow.
Consider the formula $(b_i \vee r_i \vee y_i)$. This formula, call it σ_i codes the statement *the vertex i is coloured either blue, or red, or yellow.* (Actually, this leaves the possibility open that the vertex is coloured by more than one colour, but as it happens, this will not matter). Let Σ denote the formula

$$\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n$$

Clearly, Σ codes the statement that every vertex is coloured (at least by one colour).

Let e be an edge in the graph G which is between vertices i and j. Consider the formula

$$(\neg b_i \vee \neg b_j) \wedge (\neg r_i \vee \neg r_j) \wedge (\neg y_i \vee \neg y_j)$$

it can be seen that the above formula codes the statement *the two endpoints of the edge e are not coloured by the same colour.*
Let us denote this formula as ϕ_e.
Let Φ denote the formula obtained by taking the conjunction of all the ϕ_es, e ranging over all edges in the graph G. Define Ψ_G as $(\Sigma \wedge \Phi)$.

Claim 3 *The graph G is 3–colourable iff Ψ_G is satisfiable.*

Proof. Suppose G is 3-colourable. Consider such a colouring. Assign truth values to the propositional variables of Ψ_G according to this colouring, e.g., b_i is set true if the ith vertex gets the colour blue, else it is set false. Clearly, Ψ_G will evaluate to true under this truth assignment.

On the other hand, suppose that Ψ_G is satisfiable. Consider a satisfying truth assignment to variables of Ψ_G. Colour vertices of G according to this assignment: for every vertex i at least one of b_i, r_i, and y_i must have got true under the assignment as the formula σ_i evaluates to true. If exactly one of this three variables is true in the assignment, say for example, b_i, then colour vertex i blue. If two or more of these variables are set to true, then colour i by any of the corresponding colours. As for each edge e, ϕ_e evaluates to true, the end points of e will be coloured by different colours. Therefore, G is 3-colourable. .