Reductions in Classical Complexity

Daniel Tobias

University of California, Davis

November 7, 2022

Contents

Introduction

Preliminaries

Suppose Σ^* is the set of finite strings on an alphabet Σ.

Definition. A problem $X \subseteq \Sigma^*$ is a language interpreted as the set of strings that correspond to a "Yes" instance of the decision problem it defines.

- Call an *instance* of X a set of fixed inputs for the problem and denote it I_X. For example if the problem is k-coloring then I_X could be (G, k) where G is a graph and k is an integer.

Preliminaries

Suppose Σ^* is the set of finite strings on an alphabet Σ.

Definition. A problem $X \subseteq \Sigma^*$ is a language interpreted as the set of strings that correspond to a "Yes" instance of the decision problem it defines.

- Call an *instance* of X a set of fixed inputs for the problem and denote it I_X. For example if the problem is k-coloring then I_X could be (G, k) where G is a graph and k is an integer.
- I_X can be interpreted as a language, but this formality is not often considered depending on the type of reduction you're going for.

Preliminaries

We would like a way to say "problem Y is at least as hard as X" or "X' is no more difficult than Y" which we can denote $X \leq Y$. To accomplish this we will construct a **reduction**. Now I want to present two different notions of reducibility.

Reductions

Many-to-one Reductions

Definition. Suppose $X \subset \Sigma^*$, $Y \subset \Gamma^*$ are two problems. A many-to-one reduction is a computable function $f : \Sigma^* \to \Gamma^*$ such that if $x \in X$ then $f(x) \in Y$. Then we can say X is reducible to Y, $X \leq_m Y$.

If f is polynomial time, then this is called a **Karp** reduction $X \leq_P Y$.

$SAT \leq_P 3SAT$

- A clause is a sequence of boolean variables connected by logical disjunction ie: $x_1 \vee x_2 \vee x_3$ is a clause of length 3.
- A conjunctive normal form (CNF) formula is a sequence of clauses connected by logical conjunction ie: $c_1 \wedge c_2$.

The problem SAT is the decision problem: Is there an assignment to boolean variables $x_1, ..., x_N$ such that $\varphi(x_1, ..., x_N)$ is true where φ is a CNF formula.

The problem $3SAT$ is the same thing, but now in the sentence φ each clause is of length 3. We can easily see the $3SAT \leq_P SAT$ since an instance of $3SAT$ is an instance of SAT, but now we want to prove $SAT \leq_P 3SAT$.

$SAT \leq_P 3SAT$

We proceed by mapping each type of clause we may encounter in SAT to a clause of length 3. Let's count the number of boolean variables in each clause and discuss how to map to a clause with 3 variables.

- Clause has 1 variables ($c = x_1$): Introduce two new literals u and v. Now we can construct $c' = (x \vee u \vee v) \wedge (x_1 \vee u \vee \neg v) \wedge (x_1 \vee \neg u \vee v) \wedge (x_1 \vee \neg u \vee \neg v)$. Notice $c' \iff c$.
- Clause has 2 variables ($c = x_1 \vee x_2$): Introduce 1 new literal u and construct $c' = (x_1 \vee x_2 \vee u) \wedge (x_1 \vee x_2 \vee \neg u)$. Again $c' \iff c$.
- Clause has 3 variables: We don't need to do anything here.
- Clause has more than 3 variables: $c' = (x_1 \vee x_2 \vee u_1) \wedge (x_3 \vee \neg u_1 \vee u_2) \wedge (x_4 \vee \neg u_2 \vee u_3) \vee \cdots \vee (x_{k-2} \vee \neg u_{k-4} \vee u_{k-3}) \wedge (x_{k-1} \vee x_k \vee \neg u_{k-3})$.

$SAT \leq_P 3SAT$

That completes the reduction. Since each new clause c' coincides with the old clause c we can see that this reduction will preserve membership in the language. Now let's talk about a different type of reduction.

Turing reducibility

- An **oracle** to X is a function $\mathcal{O}$ which for any word $w \in \Sigma^*$ $\mathcal{O}(w)$ returns "yes" if $w \in X$.
- X is **Turing reducible** to Y if there is an algorithm which solves X using oracle access to Y. We can then write $X \leq_{TM} Y$.

Turing reductions are great for proving decidability. If $X \leq_{TM} Y$ then if Y is decidable so is X (and we can apply the contrapositive to show undecidability).

Turing reducibility

Suppose A is some complexity class. Then the set of problems solvable by an algorithm in A with oracle access to a language L is A^L. From the example Karp reduction done on the previous slides we can see that $P^{SAT} = P^{3SAT}$.

This is enough to make you wonder if there is a connection between these types of reductions.

Relating reductions

Say $X \leq_P Y$ through the mapping f. Then we can construct an algorithm which solves X with access to an oracle Y; we apply the mapping f to the word w, send the image to oracle for Y, and return the response from the oracle. This tells us $X \leq_{TM} Y$.

Conversely say we have $X \leq_{TM} Y$. Then there is an algorithm $\mathcal{A}$ which solves X with oracle $\mathcal{O}$ access to Y with additional restriction that we only query the oracle **once** in this algorithm. Then we can build a many to one reduction $f(w) = \widetilde{w}$ where $\widetilde{w}$ is the oracle query passed in $\mathcal{A}$.

Restricting reductions

Notice in the previous slide that I restricted the number of calls to the oracle in order to faithfully relate a Turing and a Karp reduction. In general we can bound to polynomial runtime of the algorithm and polynomial calls of an oracle to get a **Cook reduction**. It is vital keep track of the resources used in the reduction when trying to prove a problem belongs to a specific complexity class.

The end!