11 Reductions

We mentioned above the idea of solving problems by translating them into known soluble problems, or showing that a new problem is insoluble by translating a known insoluble into it. Reductions are the formal tool to implement this idea. It turns out that there are several subtly different variations – we will mainly use one form of reduction, which lets us make a fine analysis of uncomputability.

First, we introduce a technical term for a Register Machine whose purpose is to translate one problem into another.

Definition 12 A Turing transducer is an RM that takes an instance \( d \) of a problem \((D, Q)\) in \( R_0 \) and halts with an instance \( d' = f(d) \) of \((D', Q')\) in \( R_0 \).

Thus \( f \) must be a computable function \( D \to D' \) that translates the problem, and the transducer is the machine that computes \( f \).

Note that the reduction is just a transducer (i.e. translates the problem) that translates correctly (so that the answer to the translated problem is the same as the answer to the original). We’ll see several examples shortly.

A subtle point is that the definition requires \( d \in Q \) iff \( f(d) \in D' \) – we are not allowed to negate the answer and arrange that \( d \in Q \) iff \( f(d) \notin D' \), although you might think that’s a perfectly reasonably thing to do. We’ll see later that negation is surprisingly powerful, and if we allow it, we lose the ability to make certain distinctions.

Given an m-reduction from \( Q \) to \( Q' \), and an oracle for \( Q' \), we can build an RM that solves \( Q \): we take the transducer that translates \( d \) to \( f(d) \), and then we plug on to the end a call to \textsc{oracle}_{Q'}(0). Note that the oracle call must be the last thing the machine does – we can’t post-process the answer (to negate it, for example).

If we do want to allow post-processing, we use the more powerful notion of Turing reduction. The definition is usually given in terms of oracles:

Definition 14 A Turing reduction from \((D, Q)\) to \((D', Q')\) is an RM \( M \), equipped with an oracle for \( Q' \), such that \( M \) computes \( Q \).

That is, we build a machine to compute \( Q \), being allowed to use an oracle for \( Q' \) as much as we like. We are allowed, for example, not just to negate it, but solve multiple instances of \((D', Q')\) in order to solve a single instance of \((D, Q)\).

12 Showing undecidability by reduction

Unless you can do a diagonalization proof directly, reductions are the only way to show that something is undecidable. We reduce from an undecidable problem (e.g. the Halting Problem \( H \)) to the target problem \((D, Q)\):

Theorem 15 Let \((D, Q)\) be a problem. Suppose there is an m-reduction \( \text{Red}(H, Q) \) from \((RM, H)\) to \((D, Q)\). Then \( Q \) is undecidable.
We prove the theorem by contradiction. Suppose $Q$ is decidable. Then given $M \in RM$, feed it to $Red(H, Q)$ to decide $M \in H$. But if $Q$ is decidable, we can replace the oracle, and $Red(H, Q)$ is just an ordinary RM. Hence $H$ is decidable – contradiction. So $Q$ must be undecidable.

Constructing $Red$ is usually either very easy, or rather long and difficult! For a relatively simple example of a long and complicated reduction, see https://hal.archives-ouvertes.fr/hal-00204625v2 a simplified proof by Nicolas Ollinger of the famous ‘tiling problem’ due to Hao Wang.

13 The Uniform Halting Problem

Definition 16 The Uniform Halting Problem $UH$ is the subset of (codes of) Register Machines $M$ such that $M$ halts on every input. 

Obviously, solving $UH$ is at least as hard as solving $H$ (in fact, it’s harder, as we’ll see later), but we still need to prove it. The informal proof is quite easy, as on the slides:

- We need an m-reduction from $H$ to $UH$:
- given machine $M$, input $R$, build a machine $M'$ which ignores its input and overwrites it with $R$, then behaves as $M$.
- Then $M'$ halts on any input iff $M$ halts on $R$.

That is, however, a bit too brief to be a proof acceptable for an exam answer. Here is a full-marks exam answer:

Question: Show, by reduction from Halting, that the Uniform Halting problem is undecidable.

Answer: It is a theorem that if $Q$ can be reduced by a many–one (or Turing) reduction to $Q'$, and $Q$ is undecidable, then $Q'$ is undecidable. Let $(M, R)$ be a machine and input. Define $M'$ to ignore its input, and run $M$ on $R$. It is clear that the construction of $M'$ is computable, by, e.g., starting $M'$ with code that loads $R$ into its registers and then jumps to the start of $M$.

For a reduction, we need that $(M, R) \in H \iff M' \in UH$. This is true by construction: if $(M, R)$ halts, then $M'$, which behaves as $M$ on $R$, halts whatever its input; conversely, if $M'$ halts on all (or indeed any) input, then $M$ halts on $R$.

Thus we have a reduction and have shown the result.

If you don’t like writing words, here’s a reduced verbiage version, which is ok if it’s correct, but runs the risk of getting less partial credit if it has mistakes!

Answer: We know if $Q \leq_m Q'$ and $Q$ undecidable, then $Q'$ undecidable. Let $(M, R)$ be a machine and input. Define $M'$ to ignore its input and run $M$ on $R$. The function $(M, R) \mapsto M'$ is clearly computable, and $M'$ is an instance of $UH$. By construction, $M'$ halts on any input iff $M$ halts on $R$. Thus $H \leq_m UH$ and we are done.

Usually, the construction of the transducer is fairly obvious (or very difficult), and can be described informally. You must remember to write in an exam answer the check that $d \in Q$ iff $d' \in Q'$.

12