CS 373 Summer 2011
Theory of Computation
Lucas Cook

Reduction Cheatsheet

2011 July 22

Here, we provide a pair of outlines for doing reduction proofs. Notice that our proofs in class follow these outlines exactly!

1 Reduction Overview

Reductions allow us to prove undecidability or unrecognizability of a 'target' language L. The proof is in the form of a contradiction: we assume that L is decidable/recognizable, and show that we can then make a decider/recognizer for some known undecidable/unrecognizable language K. When writing out reductions as TM code, there are two things that you need to do:

1. Give a machine *for* K that uses a blackbox *for* L.
2. In words, argue that this machine actually decides/recognizes K.

This is called a ***reduction from*** K ***to*** L. Notice that the blackbox is for the target language L, not K! This is because we are assuming that L is decidable/recognizable in order to derive a contradiction about K.

In lecture, we showed the following reductions (and maybe more):

1. L_{DIAG} to A_{TM}, showing that A_{TM} is undecidable.
2. A_{TM} to $HALT$, showing that $HALT$ is undecidable.
3. A_{TM} to E_{TM}, showing that E_{TM} is undecidable.
4. E_{TM} to EQ_{TM}, showing that EQ_{TM} is undecidable. (mapping reduction)
5. $\overline{A_{TM}}$ to $REGULAR$, showing that $REGULAR$ is unrecognizable. (mapping reduction)
6. $\overline{A_{TM}}$ to $FINITE$, showing that $FINITE$ is unrecognizable. (mapping reduction)
7. $\overline{A_{TM}}$ to $INFINITE$, showing that $INFINITE$ is unrecognizable. (mapping reduction)

2 Reductions for Undecidability

Here is a general outline for a proof of the undecidability of L using a reduction from K, where K is known to be undecidable already. Informally, we call M_L the "black box" for L since we assume it works without knowing how.

Proof:

We will provide a reduction from K to L. Assume that the machine M_L decides langauge L. Then the following machine decides language K:

M_K: on the input w for problem K
... // Somehow, use M_L to figure out the answer to K.
// (If you think you can decide K directly without M_L, something is wrong!
// You have shown K is decidable, which defies our claim!)
// Then **accept** or **reject** w

Since M_L terminates in finite time on all inputs, M_K does as well.
//Now we argue that $\mathcal{L}(M_K) = K$. Usually this involves reasoning
// about different cases for $w \in K$ or $w \notin K$.
...
But this is a contradiction, since K is known to be undecidable.

■

Inside of M_K, we can use M_L as many times as we want and however we want, as long as the whole computation stops in finite time and decides K.

3 Mapping Reductions for Undecidability and Unrecognizability

Mapping reductions are a special type of reduction that allows us to show both undecidability and unrecognizability. When a mapping reduction exists, we use the symbol $\leq_M$. The main outline of a mapping reduction from K to L follows. It works with either the 'decidable' or 'unrecognizable' terms, as long as your are consistent.

Proof:

We show $K \leq_m L$. Assume that the machine M_L *decides/recognizes* langauge L. Then the following machine *decides/recognizes* language K:

M_K: on the input w for problem K
... // Somehow, transform w into some other input w' for problem L in finite time.
run M_L on input w'
if M_L accepts, **accept** w.
else reject w //This means M_L rejected w'

//Argue that $\mathcal{L}(M_K) = K$. Usually this is accomplished by doing
//a case for all $w \in K$ and a case for all $w \notin K$.

...

Since there is a mapping reduction $K \leq_m L$ and K is *undecidable/recognizable*, then L is *undecidable/unrecognizable*.

■

The mapping reduction requires that M_L is called exactly once at the end of M_K, and the output of M_L is used as the output of M_K. Thus a mapping reduction is a special type of reduction. This requirement is what gives us the mathematical statement:

$$w \in K \Leftrightarrow w' \in L$$

where w' is some input for L that is made from w. (i.e. There exists computable f such that $f(w) = w'$.)