CS 4510 Automata and Complexity 4/11/2022

Lecture 18

Lecturer: Frederic Faulkner *Scribe(s):*

1 Even More Undecidable Languages

1.1 Salutations!

Recall that last lecture we proved two new languages were undecidable, namely A_{TM} and ANY_{TM}. We did these proofs by correlating the decidability of each language with another language that we already knew was undecidable, with a tool known as a reduction. We will now consider several more examples:

Theorem 1 *$HELLO = \{\langle M \rangle \mid M$ accepts the input "hello"$\}$ is undecidable.*

Proof. We give a mapping reduction f from a known undecidable language, $HALT$, to $HELLO$. Note that f is a function that takes in both a machine description and a string, but it returns *only* a machine description. Why? Because elements of $HALT$ consist of a machine description and an input string, but elements of $HELLO$ consist only of a machine description.

$f(\langle M \rangle, x) = \langle M' \rangle$ where M' is constructed from M and x as follows:

```
M' on input $y$:
    1. run $M(x)$
    2. accept.
```

Again, we must prove that f is a mapping-reduction, i.e. that $\langle M \rangle, x \in HALT \iff \langle M' \rangle \in HELLO$.

- $\langle M \rangle, x \in HALT \rightarrow \langle M' \rangle \in HELLO$

  Suppose M halts on x. Then, in step 1, we finish simulating M, so we reach step 2. In other words, M' will accept everything. But if M' accepts everything, then M' definitely accepts "hello".

- $\langle M' \rangle \in HELLO \rightarrow \langle M \rangle, x \in HALT$

  Suppose M' accepts "hello". Then that must mean that reaches step 2 on the input "hello". But the only way for it to reach step 2 is to finish step 1, which means that M eventually stops running on input x.

This completes the reduction and proves that $HELLO$ is undecidable. Why? Remember, if we had a decider for $HELLO$, then we could do the conversion here on $HALT$-type problems and then use that $HELLO$-decider to decide $HALT$. But there cannot be any process that decides $HALT$, so there cannot be any $HELLO$-decider.

1.2 Behavior problems

Here is an important fact about M' as defined in the following proof: even though you have no idea what M and x are (or whether M will halt on x), you know that M' must have one of two languages. Either M' accepts everything, so $L(M') = \Sigma^*$, or M' accepts nothing, in which case $L(M') = \emptyset$. Now, it turns out that the pair $\Sigma^*, \emptyset$ has an important relationship with $HELLO$: machines with the former language are in $HELLO$, and machines with the latter are not. So M', by varying between those languages, varies from being a YES-instance of $HELLO$ to a NO-instance of hello.

You might notice that we have seen M' before. This is the same machine we constructed when reducing $HALT$ to A_{TM}, and it works for a lot of reductions. In other words, when you are trying to construct a new mapping reduction, basing that reduction on this definition of M' can be a good starting point. However, it will not work for all reductions. In particular, if machines whose languages are Σ^* or $\emptyset$ are both YES-instances or both NO-instances of the language you are discussing, then M' will not work in your reduction. (And of course, if the language you are reducing to is not a set of machine descriptions at all, then you will need to do something else entirely.) Let's see an example where the machine we construct in the mapping reduction must be (slightly) more complex than the ones we've seen so far.

1.3 XSalutations!

Theorem 2 *$XHELLO = \{\langle M\rangle \mid L(M) = \{\text{“hello”}\}\}$ is undecidable.*

This is the set of machines which not only accept “hello”, but also do not accept any other string. Again, to prove that this is not a decidable property, we will reduce from HALT. Our reduction function f takes in a machine description and an input ($\langle M\rangle$, x), and returns a machine description $\langle M'\rangle$. What should M' do? Note that $\langle M'\rangle$ must be in $XHELLO$ if M halts on x. That means that, when we build M', it must be possible that M' might accept “hello” and no other string. How do we do this? See the example below:

$f(\langle M\rangle, x) = \langle M'\rangle$ where M' is constructed from M and x as follows:

```
M' on input $y$:
    1. run $M(x)$
    2. if $y$ = “hello”: accept $y$
    3. else reject $y$
```

Again, we must prove that f is a mapping-reduction, i.e. that $\langle M\rangle, x \in HALT \iff \langle M'\rangle \in XHELLO$.

- $\langle M\rangle, x \in HALT \rightarrow \langle M'\rangle \in XHELLO$

  Suppose M halts on x. Then, in step 1, we finish simulating M, so we reach step 2. Then, in step 2, we will accept “hello” and reject everything else, so $\langle M'\rangle \in XHELLO$.

- $\langle M' \rangle \in XHELLO \rightarrow \langle M \rangle, x \in HALT$

  Suppose M' accepts "hello", and nothing else. In particular, the fact that M' accepts "hello" means that it reaches step 2, and so doesn't get stuck in step 1. So M eventually stops running on input x, and $\langle M \rangle, x \in HALT$.

2 Even Harder Than HALT

We already know that some languages are, in a sense, "harder" then HALT. HALT is not decidable, but it is recognizable. However, some languages, such as $\overline{HALT}$, are not even recognizable! We can use reductions to discuss non-recognizability in the same way that we use them to prove undecidability.

Theorem 3 *If $A \leq_m B$, and B is recognizable, then A is recognizable.*

Corollary 4 *If $A \leq_m B$, and A is not recognizable, then B is not recognizable either.*

The proofs are almost identical to the corresponding theorems for undecidability, and so are omitted. Note that proving a language non-recognizable is basically the same process as proving a language undecidable. You still find a mapping reduction from the language of your choice, but the only difference is that you must choose a known *non-recognizable* (rather than just undecidable) language as the basis of the reduction.

Speaking of non-recognizable languages, here is a new one:

Theorem 5 $E_{TM} = \{\langle M \rangle \mid M$ *accepts no strings*$\}$ *is not recognizable.*

Proof. Although we could prove this with a reduction, we could also notice that E_{TM} and ANY_{TM} are closely related. The latter contains machines that accept at least one string, and the former contains machines that do not accept at least one string, i.e. accept no strings. The two sets are almost complements of each other. In fact, $\overline{ANY_{TM}} = E_{TM} \cup \{s \mid s$ is not a valid TM description$\}$. Now, $\overline{ANY_{TM}}$ is the complement of a recognizable-but-not-decidable language, and so is not recognizable. But since $\overline{ANY_{TM}}$ is the union of two sets, at least one of those sets is also not recognizable. (It is easy to prove that the union of two recognizable languages is itself recognizable.) But it is a recognizable task to check if a string is a valid TM description, so E_{TM} must be non-recognizable instead.

Since E_{TM} is non-recognizable, we can use it in the following reduction.

Theorem 6 $EQ_{TM} = \{\langle M \rangle, \langle N \rangle \mid M$ *and* N *have the same language*$\}$ *is not recognizable.*

To prove this, we will do another mapping reduction, but we must pick a known non-recognizable language as the source of the reduction. We choose E_{TM}. Thus, we must define a function f that takes in a single machine description $\langle M \rangle$, and outputs two machine descriptions $\langle N \rangle, \langle N' \rangle$. (Why two? Because the elements of EQ_{TM} are pairs of Turing machines.)

$f(\langle M \rangle) = (\langle N \rangle, \langle N' \rangle)$ where $N = M$ and N' is defined as follows:

```
$N'$ on input $y$:
    1. reject y
```

How many strings does N accept? Exactly none, since it rejects immediately as soon as it is started for any input. So M has the same language as N if and only if M also accepts no strings, completing the reduction.