3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 8

Some undecidable, unrecognisable languages

Theorem 1. *The language*

$$A_{TM} = \{\langle M, \sigma \rangle : M \text{ is a Turing machine which accepts } \sigma\}$$

is undecidable.

Proof. Suppose we had a Turing machine D that decides A_{TM}. Then we can define another Turing machine E that decides the language $L = \{\langle M \rangle : D \text{ rejects } \langle M, \langle M \rangle\rangle\}$.

If E gets a string $\langle M \rangle$ it simulates D with $\langle M, \langle M \rangle\rangle$ as input. If D rejects E accepts, and vice versa.

Now if D accepts $\langle E, \langle E \rangle\rangle$, then E rejects $\langle E \rangle$, meaning D rejects $\langle E, \langle E \rangle\rangle$.

If D rejects $\langle E, \langle E \rangle\rangle$ then E accepts $\langle E \rangle$, meaning D accepts $\langle E, \langle E \rangle\rangle$.

In both cases we get a contradiction, and so no such D can exist. □

Observe there is a subtle issue that in $\langle M, \langle M \rangle\rangle$ the encoding $\langle M \rangle$ inside the brackets is over the alphabet of M, whereas $\langle M, \langle M \rangle\rangle$ is an encoding over the alphabet of D. This doesn't matter since E "knows" D's encoding scheme, and so can construct $\langle M, \langle M \rangle\rangle$ without D knowing where it came from.

Suppose that there exists a Turing machine U which can take as input (the encoding of) Turing machine M and input σ to M and simulates M on σ. We will take it as fact that such a Turing machine, known as a **universal Turing machine** exists. (Constructing a universal Turing machine is not very difficult, but somewhat tedious).

Then we have:

Theorem 2. *A_{TM} is recognisable.*

Proof. On input $\langle M, \sigma \rangle$, U does the following: Simulate M on σ. If M ever accepts, then accept, if it ever rejects, then reject.

Hence, if $\langle M, \sigma \rangle \in A_{TM}$, it will be accepted by U, and if $\langle M, \sigma \rangle \notin A_{TM}$, it will either be rejected by U, or U will loop. Therefore, U recognises A_{TM}.

□

A language L over over Σ is **co-recognisable** if its complement $\overline{L} = \Sigma^* \setminus L$ is Turing recognisable.

Theorem 3. *A language is decidable if and only if it is recognisable and co-recognisable.*

Proof. Suppose a language L over alphabet Σ is decidable. Then L is recognisable. Also, its complement $\overline{L}$ is decidable, and therefore recognisable.

Now suppose L and $\overline{L}$ are both Turing recognisable, and suppose M_1 and M_2 recognise them respectively. Then on input $\sigma \in \Sigma^*$, either M_1 or M_2 will halt with accept. A multitape Turing machine M can be constructed that simulates both M_1 and M_2 at the same time, and halts when the first of them halts. If M_1 halted, it halts with accept, and if M_2 halted, it halts with reject. Thus, there is a Turing machine that accepts $\sigma \in L$ and rejects $\sigma \notin L$, meaning L is decidable. □

Corollary 4. $\overline{A_{TM}}$ *is not recognisable.*

Proof. Theorem 2 says A_{TM} is recognisable, so if $\overline{A_{TM}}$ were recognisable, then by Theorem 3 A_{TM} would be decidable. But we know from Theorem 1 that this is not the case. Hence, $\overline{A_{TM}}$ is not recognisable. □

The halting problem

Theorem 5. *The language*

$$H_{TM} = \{\langle M, \sigma \rangle : M \text{ is a Turing machine that halts on } \sigma\}$$

is undecidable.

Proof. By contradiction. Suppose H_{TM} is decided by some Turing machine D. Now suppose D accepts $\langle M, \sigma \rangle$. Then M halts on σ. We can then use a universal Turing machine to simulate M on σ to see if M accepted or rejected σ. On the other hand, if D rejects $\langle M, \sigma \rangle$, then M does not accept σ.

We have thereby used D and U to decide the language A_{TM}. But by Theorem 1 we know we cannot do this. Therefore D does not exist and H_{TM} is undecidable.

□

The proof of Theorem 5 used the principle of **reducibility**. If you have two problems A and B, then A **reduces** to B if solving B will give you a solution to A. In this particular case, we reduced the problem of deciding A_{TM} to the problem of deciding H_{TM}, so finding a solution to the latter was used to give a solution for the former (and we know that no such solution could exist, so the solution to the halting problem could not exist either).