Theory of Computation, CSCI 438 spring 2022
Cantor's Diagonalization Method (the existence of an uncountable set), 202-207
April 18

1. Prove Theorem 4.17: The real numbers are uncountable. (page 205)

 Proof using Cantor's Diagonalization:
 If the real numbers are countable, then all subsets of the real numbers must also be countable. Consider the real numbers between 0 and 1 (non-inclusive), denoted (0, 1). Suppose that the real numbers (0, 1) are countable. Then there is a 1-1 mapping between these numbers and the natural numbers. Having a 1-1 mapping with the natural numbers is the same as being able to list the real numbers between 0 and 1.

 Construct a new real number between 0 and 1 by changing the decimal digit in the ith column of the ith real number. Change the digit to something other than a 0 or 9 (since 0.999… is the same as 1.000…). This new number is a real number between 0 and 1, yet, it is different than every number on the list. This creates the needed contradiction – that the numbers can be listed, yet there is a number that is not on the list. This proves that the real numbers between 0 and 1 are not countable. If the real numbers between 0 and 1 are not countable, the entire set of real numbers is also not countable.

2. Use Cantor's Diagonalization technique to give another proof that for A_{TM} is not decidable. That is, there is not a decider for A_{TM}.

   The set of all Turing machine descriptions $\{<TM_1>, <TM_2>, \ldots\}$ is countable. This is because Turing machine descriptions are essentially programs. Programs must be written in some language, say Σ (don't get this confused with the input alphabet of a Turing machine) and the strings of Σ* for any alphabet Σ are countable. (Many of these strings will not make legal encodings of TM, but that is ok. It isn't hard to create validators for languages which say 'yes' this string is a valid program in the language, or "no" this string is not a valid program in the language.) TM are countable in the same way that proofs in a formal system are countable.

   List all TM down the rows (labeling the rows). List all encodings of TM across the columns (labeling the columns).

   Suppose, by way of contradiction, that A_{TM} is decidable. Then in every cell, for TM_j and $<TM_k>$, if A_{TM} accepts $(TM_j, <TM_k>)$ then write "accept"; else write "reject.

	$<TM_1>$	$<TM_2>$	$<TM_3>$	$<TM_4>$...
TM_1	accept	reject	reject	accept
TM_2	accept	reject	reject	accept
TM_3	reject	accept	reject	Reject
TM_4	accept	reject	reject	accept
⋮	⋮	⋮	⋮	⋮

   Given this table, it is possible to describe a new TM by going down the diagonal are switching accept to reject, and reject to accept.

	$<TM_1>$	$<TM_2>$	$<TM_3>$	$<TM_4>$...
TM_1	~~accept~~ reject	reject	reject	accept
TM_2	accept	~~reject~~ accept	reject	accept
TM_3	reject	accept	~~reject~~ accept	reject
TM_4 ⋮	accept	reject	reject	~~accept~~ reject

   Let this new TM be D.

We have defined a new TM D. This new TM must be somewhere in the list going down the side and across the top.

	<TM_1>	<TM_2>	<TM_3>	<TM_4>	…	<D>	…
TM_1	~~accept~~ reject	reject	reject	accept			
TM_2	accept	~~reject~~ accept	reject	accept			
TM_3	reject	accept	~~reject~~ accept	reject			
TM_4	accept	reject	reject	~~accept~~ reject			
…							
D						?	
…							

Notice that if A_{TM} D accepts (A, <D>) then we had "accept", but now it has been switched to "reject. However, this doesn't make sense. If D accepts the encoding of D, then D doesn't accept the encoding of D. If D does accept the encoding of D, then D does accept the encoding of D. This is contradictory, thus our original assumption must be wrong. A_{TM} must not be decidable.

This is the same argument as: