A TM is a **decider** if it halts (i.e. accepts or rejects) on all inputs. A language $L$ is **Turing-recognizable** if there exists a TM $M$ such that $L = L(M) = \{w \mid M \text{ accepts } w\}$. $L$ is **decidable** if there exists a decider $D$ such that $L = L(D)$.

The **Church-Turing thesis** tells us that any algorithm or program can be represented by a Turing machine. Even though Turing machines aren’t practical to actually program, they’re a relatively simple model about which we can abstractly reason.

**Topics Covered**

1. TM ↔ NTM Equivalence
2. Enumerators
3. Decidable Languages
4. Undecidability of $A_{TM}$

### 1 TM ↔ NTM Equivalence

A deterministic TM already satisfies the conditions to be considered an NTM. For the other direction, we need to show that an NTM $N$ can be simulated by a deterministic TM $D$. Given $N$, we can construct a four-tape TM $D$, where $D$ writes on each tape with the following alphabets:

1. Input tape: input alphabet $\Sigma$ of $N$
2. Work tape: tape alphabet $\Gamma$ of $N$
3. Nondeterminism tape: $\{1, 2, \ldots, b\}$ where $b$ is the maximum number of transitions $N$ has for any $(q, a)$ pair
4. Counter: binary (0s and 1s)

The TM $D$ operates as follows:
On input $w$, written on the input tape:

1. Initialize the tapes. Copy $w$ to the work tape, set the nondeterminism tape to $\varepsilon$, and set the counter to 0.

2. Simulate $N$ using the work tape.
   
   (a) If $N$ has more than one transition for the current $(q,a)$, use the next symbol $t$ on the nondeterminism tape to pick which transition to follow.
   
   (b) If $t$ is too high, increment the counter and go to step 3.
   
   (c) If $t$ is $\omega$, set the counter to 0 and go to step 3.
   
   (d) If $N$ accepted, accept.
   
   (e) If $N$ rejected, increment the counter and go to step 3.

   (f) Else, go to step 2 (to simulate the next step).

3. If the nondeterminism tape contains $k$ symbols and the counter is $b^k$, reject.
   
   (a) Else, replace the contents of the nondeterminism tape with the next string in lexicographic order over \{1, 2, \ldots, b\}. If the next string is longer than the previous one, set the counter to 0.
   
   (b) Reinitialize the work tape. Copy the input tape to the work tape, erase all else, and return the head to the start of the work tape.
   
   (c) Go to step 2.

**Theorem** If $L = L(N)$ for an NTM $N$, then there exists a TM $M$ such that $L = L(M)$.

**Theorem** If $L = L(N)$ for an NTM $N$ such that $N$ is a decider, then there exists a TM $M$ such that $M$ is a decider and $L = L(M)$.

**Proof of Theorems** We can prove both of these theorems with one construction. Given an NTM $N$, we will construct a TM $M$. First, construct the multi-tape TM $D$, as described by the previous algorithm. Let $M$ be the resulting TM when converting $D$ to a single-tape TM. An analysis of these constructions (which we don’t include here) needs to show the equality of the languages of $M$ and $N$, as well as the preservation of decidability; i.e. that if $N$ is a decider then so is $M$. 

Lecture 10: Decidability

Page 2 / 6
2 Enumerators

- An **enumerator** is a TM with two tapes:
  1. A work tape with tape alphabet \(\Gamma\), where the tape is initialized to \(\varepsilon\)
  2. An output tape on which the TM can only move right or stay put, and write symbols in input alphabet \(\Sigma\) or a special delimiter \# not in \(\Sigma\)

If \(E\) is an enumerator, then its language \(L(E) = \{w \mid w\text{ appears on }E\text{'s output tape}\}\). \(E\) enumerates members of its language. If the language is finite, \(E\) halts; otherwise, \(E\) just keeps enumerating. The delimiter symbol \# indicates where one string ends and the next begins. If \(L = L(E)\) for an enumerator \(E\), then we say \(L\) is **recursively enumerable**.

**Theorem**  \(L\) is recursively enumerable if and only if it is Turing-recognizable.

**Proof**  First suppose that \(L\) is recursively enumerable, and let \(E\) be an enumerator that enumerates \(L\). We can construct a TM \(M\) to recognize \(L\) as follows:

\[
M, \text{ on input } w:
\]

1. Run \(E\).
2. If \(E\) outputs \(w\), accept.

For the other direction, suppose we are given \(M\) recognizing \(L\). Then an enumerator \(E\) enumerating \(L\) operates as follows:

\[E:\]

1. For \(i\) from 0 to \(\infty\):
   1. For \(w\) from the first to \(i\)th strings in lexicographic order over \(\Sigma\):
      i. Run \(M\) on input \(w\) for \(i\) steps.
      ii. If \(M\) accepts, print \(w\) on the output tape.

3 Decidable Languages

The language \(A_{DFA} = \{\langle D, w \rangle \mid D \text{ is a DFA and } w \in L(D)\}\). Note that \(\langle D, w \rangle\) is a binary string encoding of a DFA \(D\) and an input \(w\). We encode a DFA \(D = (Q, \Sigma, \delta, q_0, F)\) as a
binary string by mapping components of $D$ to 0/1 strings (as you saw more generally in the last homework). For example, if $D$ has fifteen states, we might encode $D$ as follows:

1. 15 states: 1111
2. Binary alphabet: 01
3. Transition function: $(0000, 0) \rightarrow 0100, \ldots$
4. Start state: 0000
5. Accept states: $(0001, 1010, \ldots)$
6. Done with description #

**Claim** $A_{DFA}$ is decidable.

**Proof** Here is a TM $M$ that decides $A_{DFA}$:

$M$, on input $s$:

1. Check if $s$ is a valid encoding of a DFA $D$ and input $w$. If not, reject.
2. Simulate $D$ on input $w$.
   
   (a) Write down all states on the work tape, and the current location in $w$.
   
   (b) Keep track of transitions.
3. Return what $D$ would return.

Note that $M$ is a decider because $D$ is a DFA; every transition is accounted for in $D$ and all inputs either accept or reject. There is no input on which $M$ will enter an infinite loop.

4 **Undecidability of $A_{TM}$**

The language $A_{TM} = \{ (M, w) \mid M \text{ is a TM and } M \text{ accepts } w \}$. (In general, $A$ is short for “accepts”.) $A_{TM}$ is Turing-recognizable but not decidable.

**Claim** $A_{TM}$ is Turing-recognizable.
Proof Here is a TM $M'$ that recognizes $A_{TM}$:

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
$M'$, on input $s$: & \quad & \\
\hline
1. Check that $s$ is an encoding of the form $\langle M, w \rangle$. & \quad & \\
2. Simulate $M$ on input $w$, and accept if $M$ accepts. & \quad & \\
\hline
\end{tabular}
\end{center}

From now on, we will generally interpret the input $s$ as being of the form $\langle M, w \rangle$ when this is the expected form of the input. Even if it’s not intentionally of the form $\langle M, w \rangle$, the binary string $s$ can correspond to some construction of a TM, which will simply fail later in the computation.

Even though $A_{TM}$ is Turing-recognizable, it’s not decidable. In particular, $M'$, as we constructed above, is not a decider. If $M$ loops then $M'$ also loops. The proof that $A_{TM}$ is undecidable is similar to our previous reductions to the Halting problem.

**Proof that $A_{TM}$ is Undecidable** Suppose for a contradiction that $A_{TM}$ were decidable. Note that every TM can be encoded as a binary string, and we have an absolute lexicographic ordering on binary strings. Then we can use a “diagonalization” method to construct a new TM. Consider the following table, in which cell $(i,j)$ corresponds to $M_i$ accepting $\langle M_j \rangle$ or not:

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
$\langle M_1 \rangle$ & $\langle M_2 \rangle$ & $\ldots$ & $\langle M_i \rangle$ \\
\hline
$M_1$ & acc & not & $\ldots$ & acc \\
$M_2$ & not & acc & $\ldots$ & acc \\
$\ldots$ & $\ldots$ & $\ldots$ & $\ldots$ & $\ldots$ \\
$M_i$ & not & acc & $\ldots$ & not \\
\hline
\end{tabular}
\end{center}

If $A_{TM}$ were decidable, then you could look up any entry of this table in finite time. However, we can construct a TM $D$ that does the opposite of any TM in the table. Specifically, $D$ will not correspond to any $M_i$ because its output behavior is always different from $M_i$’s behavior on input $\langle M_i \rangle$. If we could decide $A_{TM}$, then we could construct $D$ with this property. But $D$ cannot exist because every TM must appear in the table somewhere, so we will reach a contradiction at $D$, implying that $A_{TM}$ is not decidable.

Consider the contradiction we reach. Suppose $A_{TM}$ were decidable and let $H$ be its decider. Construct $D$ as follows:

\begin{center}
\begin{tabular}{|c|c|c|c|c|}
\hline
$\langle M_1 \rangle$ & $\langle M_2 \rangle$ & $\ldots$ & $\langle M_i \rangle$ \\
\hline
$M_1$ & acc & not & $\ldots$ & acc \\
$M_2$ & not & acc & $\ldots$ & acc \\
$\ldots$ & $\ldots$ & $\ldots$ & $\ldots$ & $\ldots$ \\
$M_i$ & not & acc & $\ldots$ & not \\
\hline
\end{tabular}
\end{center}
What does $D$ do on input $\langle D \rangle$? In step 1, $D$ runs $H$ on input $\langle D, \langle D \rangle \rangle$. If $H$ accepts, $D$ rejects (a contradiction). If $H$ doesn’t accept, $D$ accepts (a contradiction). By construction, if $H$ is a decider then $D$ is a decider. However, $D$ cannot be a decider, so neither can $H$. It follows that our initial assumption was incorrect, and $A_{TM}$ is undecidable.