Turing Machines
Reading Assignment: Sipser Chapter 3.1, 4.2

4.1 covers algorithms for decidable problems about DFA, NFA, RegExp, CFG, and PDAs, e.g. slides 17 & 18 below. I’ve talked about most of this in class at one point or another, but skimming 4.1 would probably be a good review.
Define $M = (Q, \Sigma, \Gamma, s, q_0, q_{acc}, q_{rej})$

- $Q$: finite state set
- $\Sigma$: finite input alphabet set
- $\Gamma$: finite tape alphabet
- $s$: $Q \times \Gamma \rightarrow Q \times \Gamma \times L \times R$ transition function
- $q_0 \in Q$: start state
- $q_{acc} \in Q$: accept state
- $q_{rej} \in Q$: reject state
Example

$L = \{ w \# w \mid w \in \{0,1\}^* \}$

1. check that there's a single #
2. read & remember left most letter
3. scan to # & compare next letter
4. if ok, cross it off
5. repeat
• $Q = \{q_1, \ldots, q_{14}, q_{\text{accept}}, q_{\text{reject}}\}$,
• $\Sigma = \{0, 1, \#\}$, and $\Gamma = \{0, 1, \#, x, y\}$.
• We describe $\delta$ with a state diagram (see the following figure).
• The start, accept, and reject states are $q_1$, $q_{\text{accept}}$, and $q_{\text{reject}}$.

Example

$L = \{ w#w \mid w \in \{0, 1\}^* \}$

1. check that there’s a single #
2. read & remember & cross off left most letter
3. scan to # & compare next letter
4. if ok, cross it off
5. reject

All other transitions go to $q_{\text{reject}}$
By definition, no transitions \textit{out} of \( q_{\text{acc}} \), \( q_{\text{rej}} \);

\( M \) \textit{halts} if (and only if) it reaches either

\( M \) \textit{loops} if it never halts ("loop" might suggest "simple", but non-halting computations may of course be arbitrarily complex)

\( M \) \textit{accepts} if it reaches \( q_{\text{acc}} \),

\( M \) \textit{rejects} by halting in \( q_{\text{rej}} \) or by looping

The language \textit{recognized} by \( M \):

\( L(M) = \{ \ w \in \Sigma^* \mid M \text{ accepts } w \} \)
L is *Turing recognizable* if \( \exists \text{TM } M \) s.t. \( L = L(M) \)

L is *Turing decidable* if, furthermore, \( M \) halts on all inputs

**A key distinction!**
Church-Turing Thesis

TM’s formally capture the intuitive notion of “algorithmically solvable”

Not provable, since “intuitive” is necessarily fuzzy.

But, give support for it by showing that
  (a) other intuitively appealing (but formally defined) models are precisely equivalent, and
  (b) models that are provably different are unappealing, either because they are too weak (e.g., DFA’s) or too powerful (e.g., a computer with a “solve-the-halting-problem” instruction).
Example: Multi-tape Turing Machines

\[ \delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k \]
Nondeterministic Turing Machines

\[ \delta: Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\}) \]
**Nondeterministic Turing Machines**

\[ \delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L,R\}) \]

Accept if *any* path leads to \( q_{\text{accept}} \); reject otherwise, (i.e., all halting paths lead to \( q_{\text{reject}} \))
Simulating an NTM

Key issue: avoid getting lost on $\infty$ path

Key Idea: breadth-first search

$$\text{tree arity } \leq |Q| \times |\Gamma| \times |\{L, R}\|$$  (3 in example)
Encoding things

CFG $G = (V, \Sigma, R, S)$; $\langle G \rangle =$

(1, 2, 3, 4) ((1, 2), (2, 3), (3, 1), (1, 4))

$\Sigma = ?$

DFA $D = (Q, \Sigma, \delta, q_0, F)$; $\langle D \rangle =$ (...)

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$; $\langle M \rangle =$ (...)

...
Decidability

Recall: L *decidable* means there is a TM recognizing L *that always halts*.

Example:

“The acceptance problem for DFAs”

\[ A_{DFA} = \{ <D,w> \mid D \text{ is a DFA} \& w \in L(D) \} \]
Some Decidable Languages

The following are decidable:

\( A_{\text{DFA}} = \{ <D,w> \mid D \text{ is a DFA} \& w \in L(D) \} \)

pf: simulate \( D \) on \( w \)

\( A_{\text{NFA}} = \{ <N,w> \mid N \text{ is an NFA} \& w \in L(N) \} \)

pf: convert \( N \) to a DFA, then use previous as a subroutine

\( A_{\text{REX}} = \{ <R,w> \mid R \text{ is a regular expr} \& w \in L(R) \} \)

pf: convert \( R \) to an NFA, then use previous as a subroutine
\[
\text{EMPTY}_{\text{DFA}} = \{ <D> \mid D \text{ is a DFA and } L(D) = \emptyset \} \\
\text{pf: is there no path from start state to any final state?}
\]

\[
\text{EQ}_{\text{DFA}} = \{ <A, B> \mid A \& B \text{ are DFAs s.t. } L(A) = L(B) \} \\
\text{pf: equal iff } L(A) \oplus L(B) = \emptyset, \text{ and } x \oplus y = (x \cap y^c) \cup (x^c \cap y), \text{ and regular sets are closed under } \cup, \cap, \text{ complement}
\]

\[
\text{A}_{\text{CFG}} = \{ <G, w> \mid ... \} \\
\text{pf: see book}
\]

\[
\text{EMPTY}_{\text{CFG}} = \{ <G> \mid ... \} \\
\text{pf: see book}
\]
\( E_{\text{CFG}} = \{ \langle A, B \rangle \mid A \ & B \text{ are CFGs s.t. } L(A) = L(B) \} \)

This is \textit{NOT} decidable
**Figure 4.10**
The relationship among classes of languages
The Acceptance Problem for TMs

$A_{TM} = \{ <M,w> \mid M \text{ is a TM } \& w \in L(M) \}$

Theorem: $A_{TM}$ is Turing recognizable

Pf: It is recognized by a TM $U$ that, on input $<M,w>$, simulates $M$ on $w$ step by step. $U$ accepts iff $M$ does. ☐

$U$ is called a *Universal Turing Machine*  
(Ancestor of the stored-program computer)

Note that $U$ is a recognizer, not a decider.
Programming ENIAC, circa 1947

http://en.wikipedia.org/wiki/ENIAC
The Set of Languages in $\Sigma^*$ is Uncountable

Suppose they were
List them in order
Define $L$ so that
$w_i \in L \iff w_i \notin L_i$
Then $L$ is not in the list
Contradiction
“Most” languages are neither Turing recognizable nor Turing decidable

Proof idea:

“⟨ ⟩” maps TMs into \( \Sigma^* \), a countable set, so the set of TMs, and hence of Turing recognizable languages is also countable; Turing decidable is a subset of Turing recognizable, so also countable. But by the previous result, the set of all languages is uncountable.
A specific non-Turing-recognizable language

Let $M_i$ be the TM encoded by $w_i$, i.e. $\langle M_i \rangle = w_i$

($M_i$ = some default machine, if $w_i$ is an illegal code.)

$i, \ j$ entry = 1 $\iff M_i$ accepts $w_j$

$L_D = \{ w_i \mid i, i$ entry = 0 $\}$

Then $L_D$ is not recognized by any TM
Theorem: The class of Turing recognizable languages is not closed under complementation.

Proof:

The *complement* of $D$, is Turing recognizable:

On input $w_i$, run $<M_i>$ on $w_i$ ($= <M_i>$); accept if it does. E.g. use a universal TM on input $<M_i,<M_i>>$

E.g., in previous example, $D^c$ might be $L(M_6)$
Theorem: The class of Turing decidable languages is closed under complementation.

Proof Idea:

Flip $q_{\text{accept}}$, $q_{\text{reject}}$, (just like we did with DFAs)
The Acceptance Problem for TMs

\[ A_{TM} = \{ <M,w> \mid M \text{ is a TM} \& w \in L(M) \} \]

Theorem: \( A_{TM} \) is Turing recognizable

Pf: It is recognized by a TM \( U \) that, on input \( <M,w> \), simulates \( M \) on \( w \) step by step. \( U \) accepts iff \( M \) does. \( \square \)

\( U \) is called a *Universal Turing Machine*
(Ancestor of the stored-program computer)

Note that \( U \) is a recognizer, not a decider.
$A_{TM}$ is Undecidable

$A_{TM} = \{ <M,w> \mid M \text{ is a TM & } w \in L(M) \}$

Suppose it’s decidable, say by TM H. Build a new TM D:

“on input $<M>$ (a TM), run H on $<M,<M>>$; when it halts, halt & do the opposite, i.e. accept if H rejects and vice versa”

D accepts $<M>$ iff H rejects $<M,<M>>$ (by construction)
iff M rejects $<M>$ (H recognizes $A_{TM}$)

D accepts $<D>$ iff D rejects $<D>$ (special case)

Contradiction!
Let $M_i$ be the TM encoded by $w_i$, i.e. $<M_i>$ = $w_i$ ($M_i$ = some default machine, if $w_i$ is an illegal code.)

The $i, j$ entry tells whether $M_i$ accepts $w_j$.

Then $L_D$ is not recognized by any TM.
Decidable $\subsetneq$ Recognizable
Decidable = $\text{Rec} \cap \text{co-Rec}$

L decidable iff both $L$ & $L^c$ are recognizable

Pf: ($\iff$) on any given input, dovetail (run in parallel) a recognizer for $L$ with one for $L^c$; one or the other must halt & accept, so you can halt & accept/reject appropriately.

($\Rightarrow$): from above, decidable languages are closed under complement (flip acc/rej)
The Halting Problem

\[ \text{HALT}_{TM} = \{ <M,w> \mid \text{TM } M \text{ halts on input } w \} \]

Theorem: The halting problem is undecidable

Proof:

Suppose TM R decides \( \text{HALT}_{TM} \).

Consider S:

On input \( <M,w> \), run R on it. If it rejects, halt & reject; if it accepts, run M on w; accept/reject as it does.

Then S decides \( A_{TM} \), which is impossible. R can’t exist.
Programs vs TMs

Everything we’ve done re TMs can be rephrased re programs.
From the Church-Turing thesis, we expect them to be equivalent, and it’s not hard to prove that they are.
Some things are perhaps easier with programs.
Others get harder (e.g., “Universal TM” is a Java interpreter written in Java; “configurations” etc. are much messier).
TMs are convenient to use here since they strike a good balance between simplicity and versatility.
Hopefully you can mentally translate between the two; decidability/undecidability of various properties of programs are obviously more directly relevant.
Programs vs TMs

Fix $\Sigma =$ printable ASCII
Programming language with ints, strings & function calls
“Computable function” = always returns something
“Decider” = computable function always returning 0 / 1
“Acceptor” = accept if return 1; reject if $\neq 1$ or loop
$A_{\text{Prog}} = \{<P,w>| \text{ program } P \text{ returns } 1 \text{ on input } w \}$
$\text{HALT}_{\text{Prog}} = \{<P,w>| \text{ prog } P \text{ returns something on } w \}$
...
Many Undecidable Problems

About Turing Machines

\[ \text{HALT}_{\text{TM}} \quad \text{EQ}_{\text{TM}} \quad \text{EMPTY}_{\text{TM}} \quad \text{REGULAR}_{\text{TM}} \ldots \]

About programs

Ditto! \textit{And}: array-out-of-bounds, unreachability, loop termination, assertion-checking, correctness, ...

About Other Things

\[ \text{EMPTY}_{\text{LBA}} \quad \text{ALL}_{\text{CFG}} \quad \text{EQ}_{\text{CFG}} \quad \text{PCP} \quad \text{DiophantineEqns} \ldots \]
Summary

Turing Machines

A simple model of “mechanical computation”

Church-Turing Thesis

All “reasonable” models are alike in capturing the intuitive notion of “mechanically computable”

Decidable/Recognizable – Key distinction: Does it halt

Undecidability – counting, diagonalization, reduction

\[
A_{\text{T}M} = \{ <M,w> \mid \text{TM M accepts w} \}
\]

\[
\text{HALT}_{\text{T}M} = \{ <M,w> \mid \text{TM M halts on w} \}
\]
Want More?

Check out CSE 431
“Intro Computability & Complexity”