Recognizability via Decidability

Def. A decidable predicate \( R(x,y) \) is a proposition about the input strings \( x \) and \( y \), such that some TM \( M \) implements \( R \). That is,

- for all \( x, y \), \( R(x,y) \) is TRUE \( \Rightarrow \) \( M(x,y) \) accepts
- \( R(x,y) \) is FALSE \( \Rightarrow \) \( M(x,y) \) rejects

Can think of \( R \) as a function

\[ R : \Sigma^* \times \Sigma^* \to \{ \text{True}, \text{False} \} \]

**EXAMPLES:**

- \( R(x,y) = \) “\( xy \) has at most 100 zeroes”
- \( R(N,y) = \) “TM \( N \) halts on \( y \) in at most 99 steps”
Theorem: A language $A$ is **recognizable** if and only if there is a decidable predicate $R(x, y)$ such that:

$$A = \{ x \mid (\exists y \in \Sigma^*)[R(x, y) \text{ is true}] \}$$

Proof: (1) If $A = \{ x \mid \exists y \ R(x,y) \} \,$ then $A$ is recognizable

Define the TM $M(x)$: For all strings $y \in \Sigma^*$,

- If $R(x,y)$ is true, accept.

Then, $M$ accepts exactly those $x$ s.t. $\exists y \ R(x,y)$ is true

(2) If $A$ is recognizable, then $A = \{ x \mid \exists y \ R(x,y) \}$

Suppose TM $M$ recognizes $A$.

Let $R(x,y)$ be TRUE iff $M$ accepts $x$ in $|y|$ steps

Then, $M$ accepts $x$ $\iff$ $\exists y \ R(x,y)$ is true
Example:  \( L = \{ \langle M \rangle \mid \text{TM } M \text{ accepts at least one string} \} \)

is recognizable.

Want: decidable predicate \( R \) such that

\( L = \{ \langle M \rangle \mid \exists y \in \Sigma^* \ R(\langle M \rangle, y) \text{ is true} \} \)

Define \( R(\langle M \rangle, \langle x, y \rangle) = \text{“TM M accepts string x in } |y| \text{ steps”} \)

Note that \( R \) is decidable!

Just run a universal TM on \( \langle M, x \rangle \) for \( |y| \) steps

Then:  \( L = \{ \langle M \rangle \mid \exists \langle x, y \rangle \in \Sigma^* \ R(\langle M \rangle, \langle x, y \rangle) \text{ is true} \} \)

Therefore, \( L \) is recognizable!

Can always recognize \( L \) by

“guessing \( \langle x,y \rangle \) and verifying in finite time”
Deterministic Computation

- Decidable
- accept or reject

Non-Deterministic Computation

- "Massive Parallelism"
- "Perfect Guessing"
- Recognizable
- reject
- reject
- reject
- accept

Are these equally powerful???

YES for finite automata, NO for Turing machines!
Time-Bounded Complexity Classes

Definition:
\[ \text{TIME}(t(n)) = \{ L' \mid \text{there is a Turing machine } M \text{ with time complexity } O(t(n)) \text{ so that } L' = L(M) \} \]
\[ = \{ L' \mid L' \text{ is a language decided by a Turing machine with running time } \leq c t(n) + c, \text{ for some } c \geq 1 \} \]

We showed: \( A = \{ 0^k1^k \mid k \geq 0 \} \in \text{TIME}(n \log n) \)

Puzzle: Show \( A \notin \text{TIME}( (n \log n)/\log\log\log n) \)
An Efficient Universal TM

**Theorem:** There is a (one-tape) Turing machine $U$ which takes as input:
- the code of an arbitrary TM $M$
- an input string $w$
- and a string of $t$ 1s, $t > |w|$

such that $U$ on $\langle M, w, 1^t \rangle$ halts in $O(|M|^2 \cdot t^2)$ steps and $U$ accepts $\langle M, w, 1^t \rangle$ if and only if $M$ accepts $w$ in $t$ steps.

**The Universal TM with a Clock**

**Idea:** Make a multi-tape TM $U'$ that does the above, and runs in $O(|M| \cdot t)$ steps. Each step of $M$ on $w$ is $O(|M|)$ steps of $U'$. 
The Time Hierarchy Theorem

**Intuition:** If you get more time to compute, then you can solve strictly more problems.

**Theorem:** For all “reasonable” \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) where for all \( n \), \( g(n) > n^2 f(n)^2 \), \( \text{TIME}(f(n)) \nsubseteq \text{TIME}(g(n)) \)

**Proof Idea:** Diagonalization with a clock

Make TM \( \mathcal{N} \) that on input \( \langle \mathcal{M} \rangle \), simulates the TM \( \mathcal{M} \) on input \( \langle \mathcal{M} \rangle \) for \( f(|\mathcal{M}|) \) steps, *then* flips the answer.

We showed \( L(\mathcal{N}) \) cannot have time complexity \( f(n) \)

And there is a TM running in \( O(g(n)) \) time for \( L(\mathcal{N}) \)
$P = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$

Polynomial Time

The analogue of “decidability” in the world of complexity theory
The EXTENDED Church-Turing Thesis

Everyone’s Intuitive Notion of Efficient Algorithms = Polynomial-Time Turing Machines

A controversial (dead?) thesis!

Counterexamples include \( n^{100} \) time algorithms, randomized algorithms, quantum algorithms, ...
Nondeterminism and NP
Nondeterministic Turing Machines

...are just like standard TMs, except:

1. The machine may proceed according to several possible transitions (like an NFA)

2. The machine *accepts* an input string if there *exists* an accepting computation history for the machine on the string
read write move

0 → 0, R

0 → 0, R

□ → □, R

□ → □, R

q_accept

q_reject
Definition: A nondeterministic TM is a 7-tuple $T = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where:

- $Q$ is a finite set of states
- $\Sigma$ is the input alphabet, where $\square \notin \Sigma$
- $\Gamma$ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow 2^{(Q \times \Gamma \times \{L,R\})}$
- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state, and $q_{\text{reject}} \neq q_{\text{accept}}$
Let $N$ be a nondeterministic Turing machine

An **accepting computation history** for $N$ on $w$ is a sequence of configurations $C_0, C_1, \ldots, C_t$ where

1. $C_0$ is the start configuration $q_0w$,
2. $C_t$ is an accepting configuration,
3. Each configuration $C_i$ yields $C_{i+1}$

Def. $N(w)$ accepts in $t$ time $\iff$ Such a history exists

$N$ has **time complexity** $T(n)$ if for all $n$, for all inputs of length $n$ and for all histories, $N$ halts in $T(n)$ time
Definition: \( \mathrm{NTIME}(t(n)) = \{ L \mid L \text{ is decided by a } O(t(n)) \text{ time nondeterministic Turing machine} \} \)

Note: \( \mathrm{TIME}(t(n)) \subseteq \mathrm{NTIME}(t(n)) \)

Is \( \mathrm{TIME}(t(n)) = \mathrm{NTIME}(t(n)) \) for all \( t(n) \)?

**THIS IS AN OPEN QUESTION!**

What can be done in “short” \( \mathrm{NTIME} \) that cannot be done in “short” \( \mathrm{TIME} \)?
Boolean Formulas

A satisfying assignment is a setting of the variables that makes the formula true.

\[ \phi = (\neg x \land y) \lor z \]

x = 1, y = 1, z = 1 is a satisfying assignment for \( \phi \)

Boolean variables (0 or 1)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Logical operations

Parentheses
A Boolean formula is **satisfiable** if there exists a true/false setting to the variables that makes the formula true.

**YES** \( a \land b \land c \land \neg d \)

**NO** \( \neg(x \lor y) \land x \)

\[ SAT = \{ \phi \mid \phi \text{ is a satisfiable Boolean formula} \} \]

(Q: How are we encoding formulas? A: In a “reasonable” way!)

**Encoding:** takes formula \( \phi \) of \( n \) symbols, and outputs \( O(n^c) \) bits.

**Decoding:** takes \( O(n^c) \) bits and \( i \), and outputs \( i \)-th symbol of \( \phi \).
A 3cnf-formula has the form:

\[(x_1 \lor \neg x_2 \lor x_3) \land (x_4 \lor x_2 \lor x_5) \land (x_3 \lor \neg x_2 \lor \neg x_1)\]

Ex: \((x_1 \lor \neg x_2 \lor x_1)\)

\((x_3 \lor x_1) \land (x_3 \lor \neg x_2 \lor \neg x_1)\)

\((x_1 \lor x_2 \lor x_3) \land (\neg x_4 \lor x_2 \lor x_1) \lor (x_3 \lor x_1 \lor \neg x_1)\)

\((x_1 \lor \neg x_2 \lor x_3) \land (x_3 \land \neg x_2 \land \neg x_1)\)

3SAT = \{ \emptyset | \emptyset \text{ is a satisfiable 3cnf-formula} \}
3SAT = \{ \phi \mid \phi \text{ is a satisfiable 3cnf-formula} \}

Theorem: 3SAT ∈ NTIME(n^c) for some constant c > 1

Proof Idea: On input \phi:

1. Check if the formula is in 3cnf
2. For each variable v in \phi, nondeterministically substitute either 0 or 1 in place of v
3. Evaluate the formula with 0-1s all plugged in, accept iff \phi \text{ is true}
NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)

Nondeterministic Polynomial Time

The analogue of "recognizability" in complexity theory
Theorem: $L \in NP \iff$ There is a constant $k$ and polynomial-time TM $V$ such that

$$L = \{ x \mid \exists y \in \Sigma^* \ [ |y| \leq k|x|^k \text{ and } V(x,y) \text{ accepts} \}$$

Proof: (1) If $L = \{ x \mid \exists y \ |y| \leq k|x|^k \text{ and } V(x,y) \text{ accepts} \}$ then $L \in NP$

Given the poly-time TM $V$, our NP machine for $L$ is:

$N(x)$: Nondeterministically guess $y$. 
Run $V(x,y)$ and output its answer.

(2) If $L \in NP$ then

$$L = \{ x \mid \exists y \ |y| \leq k|x|^k \text{ and } V(x,y) \text{ accepts} \}$$

Let $N$ be a nondet. poly-time TM that decides $L$. Define a TM $V(x,y)$ which accepts
$\iff y$ encodes an accepting computation history of $N$ on $x$
Moral: A language $L$ is in $\text{NP}$ if and only if there are polynomial-length proofs for membership in $L$.

$3\text{SAT} = \{ \phi \mid \exists y \text{ such that } \phi \text{ is in 3cnf and } y \text{ is a satisfying assignment to } \phi \}$

$\text{SAT} = \{ \phi \mid \exists y \text{ such that } \phi \text{ is a Boolean formula and } y \text{ is a satisfying assignment to } \phi \}$

$\text{NP} = \text{“Nifty Proofs”}$
**NP** ~ Problems with the property that, once you *have* a *solution*, it is “easy” to verify the solution

SAT is in NP because a satisfying assignment is a polynomial-length proof that a formula is satisfiable

When $\phi \in \text{SAT}$, I can prove that fact to you with a short proof you can quickly verify
The Hamiltonian Path Problem

A Hamiltonian path traverses through each node exactly once.
Assume a reasonable encoding of graphs (example: the adjacency matrix is reasonable)

\[
\text{HAMPATH} = \{ (G, s, t) \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t \}
\]

**Theorem:** \( \text{HAMPATH} \in \text{NP} \)

A Hamiltonian path \( P \) in \( G \) from \( s \) to \( t \) is a **proof** that \((G, s, t)\) is in HAMPATH

Given \( P \) (as a permutation on the nodes) can easily check that it is a path through all nodes exactly once
The k-Clique Problem

k-clique = complete subgraph on k nodes
CLIQUE = \{ (G,k) \mid G \text{ is an undirected graph with a } k\text{-clique} \}

**Theorem:** CLIQUE $\in$ NP

A $k$-clique in $G$ is a proof that $(G, k)$ is in CLIQUE

Given a subset $S$ of $k$ nodes from $G$, we can efficiently check that all possible edges are present between the nodes in $S$
A language is in NP if and only if there are “polynomial-length proofs” for membership in the language.

\( P \approx \) the problems that can be \textit{efficiently solved}

\( NP \approx \) the problems where \textit{proposed solutions can be efficiently verified}

Is \( P = NP \)?

Can problem solving be automated?
$\textbf{P = NP?}$
If P = NP...

Mathematicians/creators may be out of a job
This problem is in NP:
Short-Provability\_F
= \{ (T, 1^k) | T has a proof in F of length ≤ k \}

Cryptography as we know it may be impossible – there are no “one-way” functions!
Machines could effectively learn *any concept with a short description*

In principle, every aspect of daily life could be efficiently and globally optimized...
... life as we know it would be different

**Conjecture:**  P ≠ NP
Are these equally powerful???

YES for FAs, NO for TMs, OPEN for Polynomial Time