Complementing a TM’s Halting States

- Let $M$ decide $L$, and $M'$ be $M$ after “yes” $\leftrightarrow$ “no”.
- If $M$ is deterministic, then $M'$ decides $\bar{L}$.\(^a\)
  - So $M$ and $M'$ decide languages that complement each other.
- But if $M$ is an NTM, then $M'$ may not decide $\bar{L}$.
  - It is possible that $M$ and $M'$ accept the same input $x$ (see next page).
  - So $M$ and $M'$ may accept languages that are not even disjoint.

\(^a\)By the definition on p. 53, $M$ must halt on all inputs.
A Nondeterministic Algorithm for Satisfiability

$\phi$ is a boolean formula with $n$ variables.

1: for $i = 1, 2, \ldots, n$ do
2:     Guess $x_i \in \{0, 1\}$; {Nondeterministic choices.}
3: end for
4: {Verification:}
5: if $\phi(x_1, x_2, \ldots, x_n) = 1$ then
6:     “yes”;
7: else
8:     “no”;
9: end if
Computation Tree for Satisfiability

- $x_1 = 0$
- $x_2 = 1$
- $x_3 = 1$
- $x_4 = 0$
- $x_5 = 0$
- $x_6 = 1$
- $x_7 = 1$
- $x_8 = 0$

Branches:
- "no", "yes", "no", "yes", "yes", "no", "no", "no", "yes"
Analysis

• Recall that $\phi$ is satisfiable if and only if there is a truth assignment that satisfies $\phi$.

• The computation tree is a complete binary tree of depth $n$.

• Every computation path corresponds to a particular truth assignment\(^a\) out of $2^n$.

\(^a\)Equivalently, a sequence of nondeterministic choices.
Analysis (concluded)

- The algorithm decides language

\[ \{ \phi : \phi \text{ is satisfiable} \} . \]

- Suppose \( \phi \) is satisfiable.
  * There is a truth assignment that satisfies \( \phi \).
  * So there is a computation path that results in “yes.”

- Suppose \( \phi \) is not satisfiable.
  * That means every truth assignment makes \( \phi \) false.
  * So every computation path results in “no.”

- General paradigm: Guess a “proof” then verify it.
The Traveling Salesman Problem

- We are given \( n \) cities 1, 2, \ldots, \( n \) and integer distance \( d_{ij} \) between any two cities \( i \) and \( j \).
- Assume \( d_{ij} = d_{ji} \) for convenience.
- The traveling salesman problem (TSP) asks for the total distance of the shortest tour of the cities.\(^a\)
- The decision version TSP (D) asks if there is a tour with a total distance at most \( B \), where \( B \) is an input.\(^b\)

\(^a\)Each city is visited exactly once.

\(^b\)Both problems are extremely important. They are equally hard (p. 415 and p. 516).
A Nondeterministic Algorithm for TSP (D)

1: \textbf{for} $i = 1, 2, \ldots, n \textbf{ do}$
2: \hspace{1em} Guess $x_i \in \{1, 2, \ldots, n\}$; \{The $i$th city.\} \textsuperscript{a}
3: \textbf{end for}
4: \{Verification:\}
5: \hspace{1em} \textbf{if} $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n-1} d_{x_i, x_{i+1}} \leq B$ \textbf{then}
6: \hspace{1em} \hspace{1em} “yes”;
7: \hspace{1em} \hspace{1em} \textbf{else}
8: \hspace{1em} \hspace{1em} “no”;
9: \hspace{1em} \textbf{end if}

\textsuperscript{a}Can be made into a series of $\log_2 n$ binary choices for each $x_i$ so that the next-state count (2) is a constant, independent of input size. Contributed by Mr. Chih-Duo Hong (R95922079) on September 27, 2006.
Analysis

• Suppose the input graph contains at least one tour of the cities with a total distance at most $B$.
  – Then there is a computation path for that tour.\(^a\)
  – And it leads to “yes.”

• Suppose the input graph contains no tour of the cities with a total distance at most $B$.
  – Then every computation path leads to “no.”

---

\(^a\)It does not mean the algorithm will follow that path. It merely means such a computation path (i.e., a sequence of nondeterministic choices) exists.
Time Complexity under Nondeterminism

- Nondeterministic machine $N$ decides $L$ in time $f(n)$, where $f : \mathbb{N} \to \mathbb{N}$, if
  - $N$ decides $L$, and
  - for any $x \in \Sigma^*$, $N$ does not have a computation path longer than $f(|x|)$.

- We charge only the “depth” of the computation tree.
Time Complexity Classes under Nondeterminism

- \text{NTIME}(f(n)) \text{ is the set of languages decided by NTMs within time } f(n).
- \text{NTIME}(f(n)) \text{ is a complexity class.}
NP ("Nondeterministic Polynomial")

- Define

\[ NP \triangleq \bigcup_{k > 0} \text{NTIME}(n^k). \]

- Clearly \( P \subseteq NP \).

- Think of NP as efficiently *verifiable* problems (see p. 343).
  - Boolean satisfiability (p. 120 and p. 201), e.g.

- The most important open problem in computer science is whether \( P = NP \).
Remarks on the $P \neq NP$ Open Problem\textsuperscript{a}

- Many practical applications depend on answers to the $P \neq NP$ question.

- Verification of password should be easy (so it is in NP).
  - A computer should not take a long time to let a user log in.

- A password system should be hard to crack (loosely speaking, cracking it should not be in P).

- It took 63 years to settle the Continuum Hypothesis; how long will it take for this one?

\textsuperscript{a}Contributed by Mr. Kuan-Lin Huang (B96902079, R00922018) on September 27, 2011.
Simulating Nondeterministic TMs

Nondeterminism does not add power to TMs.

**Theorem 6** Suppose language $L$ is decided by an NTM $N$ in time $f(n)$. Then it is decided by a 3-string deterministic TM $M$ in time $O(c^{f(n)})$, where $c > 1$ is some constant depending on $N$.

- On input $x$, $M$ explores the computation tree of $N(x)$ using depth-first search.
  - $M$ does not need to know $f(n)$.
  - As $N$ is time-bounded, the depth-first search will halt.
The Proof (concluded)

- If any path leads to “yes,” then $M$ immediately enters the “yes” state.

- If none of the paths lead to “yes,” then $M$ enters the “no” state.

- The simulation takes time $O(c^f(n))$ for some $c > 1$ because the computation tree has that many nodes.

**Corollary 7** \( \text{NTIME}(f(n)) \subseteq \bigcup_{c>1} \text{TIME}(c^f(n)). \)

\(^a\)Mr. Kai-Yuan Hou (B99201038, R03922014) on October 6, 2015: \( \bigcup_{c>1} \text{TIME}(c^f(n)) \subseteq \text{NTIME}(f(n))) \)?
NTIME vs. TIME

- Does converting an NTM into a TM require exploring all computation paths of the NTM in the worst case as done in Theorem 6 (p. 132)?

- This is a key question in theory with important practical implications.
Nondeterministic Space Complexity Classes

- Let \( L \) be a language.

- Then

\[
L \in \text{NSPACE}(f(n))
\]

if there is an NTM with input and output that decides \( L \) and operates within space bound \( f(n) \).

- \( \text{NSPACE}(f(n)) \) is a set of languages.

- As in the linear speedup theorem,\(^a\) constant coefficients do not matter.

\(^a\)Theorem 5 (p. 96).
Graph Reachability

- Let $G(V, E)$ be a directed graph (digraph).
- REACHABILITY asks, given nodes $a$ and $b$, does $G$ contain a path from $a$ to $b$?
- Can be easily solved in polynomial time by breadth-first search.
- How about its nondeterministic space complexity?
The First Try: NSPACE \( (n \log n) \)

1: Determine the number of nodes \( m \); \{Note \( m \leq n. \}\)
2: \( x_1 := a \); \{Assume \( a \neq b. \}\)
3: \textbf{for} \( i = 2, 3, \ldots, m \) \textbf{do}
4: \hspace{1em} Guess \( x_i \in \{ v_1, v_2, \ldots, v_m \} \); \{The \( i \)th node.\}
5: \textbf{end for}
6: \textbf{for} \( i = 2, 3, \ldots, m \) \textbf{do}
7: \hspace{1em} \textbf{if} \( (x_{i-1}, x_i) \notin E \) \textbf{then}
8: \hspace{2em} “no”;
9: \hspace{1em} \textbf{end if}
10: \hspace{1em} \textbf{if} \( x_i = b \) \textbf{then}
11: \hspace{2em} “yes”;
12: \hspace{1em} \textbf{end if}
13: \textbf{end for}
14: “no”;
In Fact, \textsc{reachability} $\in \text{NSPACE}(\log n)$

1: Determine the number of nodes $m$; \{Note $m \leq n$.\}
2: $x := a$;
3: \textbf{for} $i = 2, 3, \ldots, m$ \textbf{do}
4: \hspace{1em} Guess $y \in \{v_1, v_2, \ldots, v_m\}$; \{The next node.\}
5: \hspace{1em} \textbf{if} $(x, y) \notin E$ \textbf{then}
6: \hspace{2em} “no”;
7: \hspace{1em} \textbf{end if}
8: \hspace{1em} \textbf{if} $y = b$ \textbf{then}
9: \hspace{2em} “yes”;
10: \hspace{1em} \textbf{end if}
11: \hspace{1em} $x := y$; \{Recycle the space.\}
12: \hspace{1em} \textbf{end for}
13: “no”;

Space Analysis

- Variables $m$, $i$, $x$, and $y$ each require $O(\log n)$ bits.
- Testing $(x, y) \in E$ is accomplished by consulting the input string with counters of $O(\log n)$ bits long.
- Hence

\[ \text{REACHABILITY} \in \text{NSPACE}(\log n). \]

- REACHABILITY with more than one terminal node also has the same complexity.
- In fact, REACHABILITY for undirected graphs is in $\text{SPACE}(\log n)$.\(^a\)

- It is well-known that REACHABILITY $\in \text{P}$.\(^b\)

\(^a\)Reingold (2004).
\(^b\)See, e.g., p. 246.
Undecidability
He [Turing] invented the idea of software, essentially.[.]
It’s software that’s really the important invention.
— Freeman Dyson (2015)
Universal Turing Machine\(^a\)

- A universal Turing machine \(U\) interprets the input as the description of a TM \(M\) concatenated with the description of an input to that machine, \(x\).\(^b\)
  - Both \(M\) and \(x\) are over the alphabet of \(U\).
- \(U\) simulates \(M\) on \(x\) so that
  \[
  U(M; x) = M(x).
  \]
- \(U\) is like a modern computer, which executes any valid machine code, or a Java virtual machine, which executes any valid bytecode.

\(^a\)Turing (1936) calls it “universal computing machine.”
\(^b\)See pp. 57–58 of the textbook.
The Halting Problem

• **Undecidable problems** are problems that have no algorithms.
  – Equivalently, they are languages that are not recursive.

• We now define a concrete undecidable problem, the **halting problem**:

\[ H \triangleq \{ M; x : M(x) \neq \uparrow \}. \]

  – Does \( M \) halt on input \( x \)?

• \( H \) is called the **halting set**.
**H** Is Recursively Enumerable

- Use the universal TM $U$ to simulate $M$ on $x$.
- When $M$ is about to halt, $U$ enters a “yes” state.
- If $M(x)$ diverges, so does $U$.
- This TM accepts $H$. 
$H$ Is Not Recursive$^a$

- Suppose $H$ is recursive.
- Then there is a TM $M_H$ that decides $H$.
- Consider the program $D(M)$ that calls $M_H$:
  1. if $M_H(M; M) = \text{“yes”}$ then
  2. ↗; \{Inserting an infinite loop here.\}
  3. else
  4. “yes”;
  5. end if

$^a$Turing (1936).
$H$ Is Not Recursive (concluded)

- Consider $D(D)$:
  - $D(D) = \uparrow \Rightarrow M_H(D; D) = \text{“yes”} \Rightarrow D \in H \Rightarrow D(D) \neq \uparrow$, a contradiction.
  - $D(D) = \text{“yes”} \Rightarrow M_H(D; D) = \text{“no”} \Rightarrow D \notin H \Rightarrow D(D) = \uparrow$, a contradiction.
Comments

• Two levels of interpretations of $M$:\footnote{Eckert & Mauchly (1943); von Neumann (1945); Turing (1946).}
  - A sequence of 0s and 1s (data).
  - An encoding of instructions (programs).

• There are no paradoxes with $D(D)$.
  - Concepts should be familiar to computer scientists.
  - Feed a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, a sorting program to a sorting program, etc.
It seemed unworthy of a grown man to spend his time on such trivialities, but what was I to do? […] The whole of the rest of my life might be consumed in looking at that blank sheet of paper.

Self-Loop Paradoxes\textsuperscript{a}

Russell’s Paradox (1901): Consider $R = \{A : A \not\in A\}$.

- If $R \in R$, then $R \not\in R$ by definition.
- If $R \not\in R$, then $R \in R$ also by definition.
- In either case, we have a “contradiction.”\textsuperscript{b}

Epimenides and Eubulides: The Cretan says, “All Cretans are liars.”\textsuperscript{c}

\textsuperscript{a}E.g., Quine (1966), *The Ways of Paradox and Other Essays* and Hofstadter (1979), *Gödel, Escher, Bach: An Eternal Golden Braid*.

\textsuperscript{b}Gottlob Frege (1848–1925) to Bertrand Russell in 1902, “Your discovery of the contradiction [...] has shaken the basis on which I intended to build arithmetic.”

\textsuperscript{c}Also quoted in *Titus* 1:12.
Self-Loop Paradoxes (continued)

Liar’s Paradox: “This sentence is false.”

Hypochondriac: a patient with imaginary symptoms and ailments.\(^a\)

Sharon Stone in *The Specialist* (1994): “I’m not a woman you can trust.”

*Numbers 12:3*: “Moses was the most humble person in all the world [⋯]” (attributed to Moses).

*Psalms 116:11*: “Everyone is a liar.”

\(^a\)Like Gödel and the pianist Glenn Gould (1932–1982).
Self-Loop Paradoxes (continued)

A restaurant in Boston: No Name Restaurant (1917–2020).


The Egyptian Book of the Dead: “ye live in me and I would live in you.”

See also John 14:10 and 17:21.

---

aSee also John 14:10 and 17:21.
Self-Loop Paradoxes (concluded)

Jerome K. Jerome (1887), *Three Men in a Boat*: “How could I wake you, when you didn’t wake me?”

Winston Churchill (January 23, 1948): “For my part, I consider that it will be found much better by all parties to leave the past to history, especially as I propose to write that history myself.”

Norbert Wiener (1953), “It is impossible to describe Bertrand Russell except by saying that he looks like the Mad Hatter.”

Karl Popper (1974), “perhaps the greatest philosopher since Kant.”

*Nobel Prize in Literature (1950).*
Reductions in Proving Undecidability

• Suppose we are asked to prove that $L$ is undecidable.

• Suppose $L'$ (such as $H$) is known to be undecidable.

• Find a computable transformation $R$ (called reduction\(^a\)) from $L'$ to $L$ such that\(^b\)

\[
\forall x \{ x \in L' \text{ if and only if } R(x) \in L \}.
\]

• Now we can answer “$x \in L'$?” for any $x$ by answering “$R(x) \in L$?” because it has the same answer.

\(^a\)Post (1944).
\(^b\)Contributed by Mr. Tai-Dai Chou (J93922005) on May 19, 2005.
Algorithm for $L'$

$x \xrightarrow{R} R(x) \xrightarrow{\text{algorithm for } L} \text{yes/no}$
Reductions in Proving Undecidability (concluded)

• $L'$ is said to be reduced to $L$.\textsuperscript{a}

• If $L$ were decidable, "$R(x) \in L?$" becomes computable and we have an algorithm to decide $L'$, a contradiction!

• So $L$ must be undecidable.

Theorem 8 \textit{Suppose language $L_1$ can be reduced to language $L_2$. If $L_1$ is undecidable, then $L_2$ is undecidable.}

\textsuperscript{a}Intuitively, $L$ can be used to solve $L'$.  

\footnotesize{©2021 Prof. Yuh-Dauh Lyuu, National Taiwan University}
Special Cases and Reduction

- Suppose $L_1$ can be reduced to $L_2$.

- As the reduction $R$ maps members of $L_1$ to a *subset* of $L_2$, we *may* say $L_1$ is a “special case” of $L_2$.

- That is one way to understand the use of the somewhat confusing term “reduction.”

---

*a* Because $R$ may not be onto.

*b* Contributed by Ms. Mei-Chih Chang (D03922022) and Mr. Kai-Yuan Hou (B99201038, R03922014) on October 13, 2015.
Subsets and Decidability

• Suppose $L_1$ is undecidable and $L_1 \subseteq L_2$.

• Is $L_2$ undecidable?

• It depends.

• When $L_2 = \Sigma^*$, $L_2$ is decidable: Just answer “yes.”

• If $L_2 - L_1$ is decidable, then $L_2$ is undecidable.
  
  – Clearly,

  $$x \in L_1 \text{ if and only if } x \in L_2 \text{ and } x \not\in L_2 - L_1.$$  

  – Therefore, if $L_2$ were decidable, then $L_1$ would be.

\[ ^a \text{Contributed by Ms. Mei-Chih Chang (D03922022) on October 13, 2015.} \]
Subsets and Decidability (concluded)

- Suppose \( L_2 \) is decidable and \( L_1 \subseteq L_2 \).
- Is \( L_1 \) decidable?
- It depends again.
- When \( L_1 = \emptyset \), \( L_1 \) is decidable: Just answer “no.”
- But if \( L_2 = \Sigma^* \), then \( L_1 = H \), then \( L_1 \) is undecidable.
The Universal Halting Problem

- The universal halting problem:
  \[ H^* \triangleq \{ M : M \text{ halts on all inputs} \}. \]

- It is also called the totality problem.
$H^*$ Is Not Recursive$^a$

- We will reduce $H$ to $H^*$.
- Given the question “$M; x \in H?$”, construct the following machine (this is the reduction):$^b$

  $$M_x(y) \{ M(x); \}$$

- $M$ halts on $x$ if and only if $M_x$ halts on all inputs.
- In other words, $M; x \in H$ if and only if $M_x \in H^*$.
- So if $H^*$ were recursive (recall the box for $L$ on p. 155), $H$ would be recursive, a contradiction.

---

$^a$Kleene (1936).

$^b$Simplified by Mr. Chih-Hung Hsieh (D95922003) on October 5, 2006. $M_x$ ignores its input $y$; $x$ is part of $M_x$’s code but not $M_x$’s input.
More Undecidability

• \( \{ M; x : \text{there is a } y \text{ such that } M(x) = y \} \).

• \( \{ M; x : \text{the computation } M \text{ on input } x \text{ uses all states of } M \} \).

• \( \{ M; x; y : M(x) = y \} \).