A Nondeterministic Algorithm for Satisfiability

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

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

Analysis

- The algorithm decides language $\{\phi : \phi$ is satisfiable$\}$.
  - The computation tree is a complete binary tree of depth $n$.
  - Every computation path corresponds to a particular truth assignment out of $2^n$.
  - $\phi$ is satisfiable if and only if there is a computation path (truth assignment) that results in “yes.”
- General paradigm: Guess a “proof” and then verify it.

The Computation Tree for Satisfiability

- We are given $n$ cities $1, 2, \ldots, n$ and integer distances $d_{ij}$ 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.
- The decision version $TSP$ ($TSP$) asks if there is a tour with a total distance at most $B$, where $B$ is an input.
- Both problems are extremely important but equally hard (p. 308 and p. 370).
A Nondeterministic Algorithm for TSP (D)
1: for \( i = 1, 2, \ldots, n \) do
2: \( \text{Guess} \ x_i \in \{1, 2, \ldots, n\}; \) (The \( i \)th city)
3: end for
4: \( x_{n+1} := x_1; \)
5: \{Verification stage\}
6: if \( x_1, x_2, \ldots, x_n \) are distinct and \( \sum_{i=1}^{n} d_{x_i, x_{i+1}} \leq B \) then
7: \( \text{“yes”;} \)
8: else
9: \( \text{“no”;} \)
10: end if
(The degree of nondeterminism is \( n \).

Time Complexity under Nondeterminism
- Nondeterministic machine \( N \) decides \( L \) in time \( f(n) \), where \( f : \mathbb{N} \rightarrow \mathbb{N} \), 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
- \( \text{NTIME}(f(n)) \) is the set of languages decided by NTMs within time \( f(n) \).
- \( \text{NTIME}(f(n)) \) is a complexity class.

NP
- Define
  \[ \text{NP} = \bigcup_{k>0} \text{NTIME}(n^k). \]
- Clearly \( P \subseteq \text{NP} \).
- Think of \( \text{NP} \) as efficiently verifiable problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
  - Hamiltonian path.
  - Graph colorability.
- The most important open problem in computer science is whether \( P = \text{NP} \).
Simulating Nondeterministic TMs

**Theorem 5** 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 $x$, $M$ goes down every computation path of $N$ using depth-first search (but $M$ does not know $f(n)$).
- If some path leads to “yes,” then $M$ enters the “yes” state.
- If none of the paths leads to “yes,” then $M$ enters the “no” state.

Graph Reachability

- Let $G(V, E)$ be a directed graph (digraph).
- Reachability asks if, 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 the nondeterministic space complexity?

NTIME vs. TIME

**Corollary 6** $\text{NTIME}(f(n)) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

- Does converting an NTM into a TM require exploring all the computation paths of the NTM as done in Theorem 5?
- This is the most important question in theory with practical implications.

The First Try in NSPACE($n \log n$)

1: $x_1 := a$; {Assume $a \neq b$.}
2: for $i = 2, 3, \ldots, n$ do
3:   Guess $x_i \in \{v_1, v_2, \ldots, v_n\}$; {The $i$th node.}
4: end for
5: for $i = 2, 3, \ldots, n$ do
6:   if $(x_{i-1}, x_i) \notin E$ then
7:     “no”;
8: end if
9:   if $x_i = b$ then
10:      “yes”;
11: end if
12: end for
13: “no”;
In Fact \( \text{REACHABILITY} \in \text{NSPACE}(\log n) \)

1: \( x := 0; \)
2: \textbf{for} \( i = 2, 3, \ldots, n \) \textbf{do}
3: \hspace{1em} \text{Guess} \( y \in \{2, 3, \ldots, n\}; \{\text{The next node.}\}
4: \hspace{1em} \textbf{if} \ (x, y) \notin E \textbf{then}
5: \hspace{2em} \text{"no"};
6: \hspace{1em} \textbf{end if}
7: \hspace{1em} \textbf{if} \ y = b \textbf{then}
8: \hspace{2em} \text{"yes"};
9: \hspace{1em} \textbf{end if}
10: \hspace{1em} x := y;
11: \hspace{1em} \textbf{end for}
12: \text{"no"};

---

### Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-to-one correspondence with \( \mathbb{N} \), the set of natural numbers.
  - Set of integers \( \mathbb{Z} \).
    * \( 0 \leftrightarrow 0, 1 \leftrightarrow 1, 2 \leftrightarrow 3, 3 \leftrightarrow 5, \ldots, -1 \leftrightarrow 2, -2 \leftrightarrow 4, -3 \leftrightarrow 6, \ldots \)
  - Set of positive integers \( \mathbb{Z}^+ \): \( i - 1 \leftrightarrow i \).
  - Set of odd integers: \( (i - 1)/2 \leftrightarrow i \).
  - Set of rational numbers: See next page.
  - Set of squared integers: \( i \leftrightarrow \sqrt{i} \).

---

### Space Analysis

- Variables \( i, x, \) 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

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

- \text{REACHABILITY} with more than one terminal node also has the same complexity.
- \text{REACHABILITY} \in P \ (p, 175).
Cardinality

- For any set $A$, define $|A|$ as $A$'s **cardinality** (size).
- Two sets are said to have the same cardinality (written as $|A| = |B|$ or $A \sim B$) if there exists a one-to-one correspondence between their elements.
- $2^A$ denotes set $A$'s **power set**, that is $\{ B : B \subseteq A \}$.
  - If $|A| = k$, then $|2^A| = 2^k$.
  - So $|A| < |2^A|$ when $A$ is finite.

Cardinality and Infinite Sets

- If $A$ and $B$ are infinite sets, it is possible that $A \subseteq B$ yet $|A| = |B|$.
  - The set of integers properly contains the set of odd integers.
  - But the set of integers has the same cardinality as the set of odd integers (p. 94).
- A lot of "paradoxes."

Cardinality (concluded)

- $|A| \leq |B|$ if there is a one-to-one correspondence between $A$ and one of $B$'s subsets.
- $|A| < |B|$ if $|A| \leq |B|$ but $|A| \neq |B|$.
- If $A \subseteq B$, then $|A| \leq |B|$.
- But if $A \subseteq B$, then $|A| < |B|$?

Hilbert's\(^a\) Paradox of the Grand Hotel

- For a hotel with a finite number of rooms with all the rooms occupied, a new guest will be turned away.
- Now let us imagine a hotel with an infinite number of rooms, and all the rooms are occupied.
- A new guest comes and asks for a room.
  - "But of course!" exclaims the proprietor, and he moves the person previously occupying Room 1 into Room 2, the person from Room 2 into Room 3, and so on . . .
- The new customer occupies Room 1.

\(^a\)David Hilbert (1862-1943).
Hilbert’s Paradox of the Grand Hotel (concluded)

- Let us imagine now a hotel with an infinite number of rooms, all taken up, and an infinite number of new guests who come in and ask for rooms.
- “Certainly, gentlemen,” says the proprietor, “just wait a minute.”
- He moves the occupant of Room 1 into Room 2, the occupant of Room 2 into Room 4, and so on,
- Now all odd-numbered rooms become free and the infinity of new guests can be accommodated in them,
- “There are many rooms in my Father’s house, and I am going to prepare a place for you,” (John 14:3)

Cantor’s\textsuperscript{a} Theorem

**Theorem 7** The set of all subsets of \( \mathbb{N} \) (\( 2^{\mathbb{N}} \)) is infinite and not countable.

- Suppose it is countable with \( f : \mathbb{N} \rightarrow 2^{\mathbb{N}} \) being a bijection.
- Consider the set \( B = \{ k \in \mathbb{N} : k \notin f(k) \} \subseteq \mathbb{N} \).
- Suppose \( B = f(n) \) for some \( n \in \mathbb{N} \).

\textsuperscript{a}Georg Cantor (1845-1918).

Galileo’s\textsuperscript{a} Paradox (1638)

- The squares of the positive integers can be placed in one-to-one correspondence with all the positive integers.
- This is contrary to the axiom of Euclid that the whole is greater than any of its proper parts.
- Resolution of paradoxes: Pick the notion that results in “better” mathematics.
- The difference between a mathematical paradox and a contradiction is often a matter of opinion.

\textsuperscript{a}Galileo (1564-1642).

The Proof (concluded)

- If \( n \in f(n) \), then \( n \in B \), but then \( n \notin B \) by \( B \)’s definition,\textsuperscript{a}
- If \( n \notin f(n) \), then \( n \notin B \), but then \( n \in B \) by \( B \)’s definition,
- Hence \( B \neq f(n) \) for any \( n \).
- \( f \) is not a bijection, a contradiction.

\textsuperscript{a}If \( B \) is empty, skip this part. Thanks to a lively class discussion on October 1, 2003.
A Corollary of Cantor’s Theorem

**Corollary 8** For any set $T$, finite or infinite,

$$|T| < |2^T|.$$  

- The inequality holds in the finite $A$ case.
- Assume $A$ is infinite now.
- $|T| \leq |2^T|$: Consider $f(x) = \{x\}$.
- The strict inequality uses the same argument as Cantor’s theorem.

---

A Second Corollary of Cantor’s Theorem

**Corollary 9** The set of all functions on $N$ is not countable.

- Every function $f : \mathbb{N} \to \{0, 1\}$ determines a set
  $$\{n : f(n) = 1\} \subseteq \mathbb{N},$$

- And vice versa.
- So the set of functions from $N$ to $\{0, 1\}$ has cardinality
  $|2^\mathbb{N}|$.
- Corollary 8 (p. 106) then implies the claim.

---

How about ...?

- Consider this subset of $2^\mathbb{N}$:
  $$2^{\mathbb{N}}_{=k} \equiv \{x : x \subseteq \mathbb{N}, \ |x| = k\}.$$  

- Is it still uncountable?
- No.
  - $|2^\mathbb{N}_{=1}| = |\mathbb{N}|$.
  - $|2^\mathbb{N}_{=2}| = |\mathbb{Q}|$.

---

\(^a\)Thanks to a lively class discussion on October 1, 2003.
Existence of Uncomputable Problems

- Every program is a finite sequence of 0s and 1s, thus a nonnegative integer.
- Hence every program corresponds to some integer.
- The set of programs is countable.
- A function is a mapping from integers to integers.
- The set of functions is not countable by Corollary 9 (p. 107).
- So there must exist functions for which there are no programs.

The Halting Problem

- Undecidable problems are problems that have no algorithms or languages that are not recursive.
- We knew undecidable problems exist (p. 108).
- We now define a concrete undecidable problem, the halting problem:
  \[ H = \{ M; x : M(x) \neq \gamma \} , \]
  - Does \( M \) halt on input \( x \)?

Universal Turing Machine

- 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, \( x \).
  - 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.

\( H \) 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 \( H \).
- Membership of \( x \) in any recursively enumerable language accepted by \( M \) can be answered by asking
  \[ M; x \in H ? \]
**H Is Not Recursive**

- Suppose there is a TM $M_H$ that decides $H$.
- Consider the program $D(M)$ that calls $M_H$:
  1. if $M_H(M;M) =$ "yes" then
  2. $\n$; { Writing an infinite loop is easy, right?}
  3. else
  4. "yes";
  5. end if
- Consider $D(D)$:
  - $D(D) =$ */\Rightarrow M_H(D;D) =$ "yes" $\Rightarrow D; D \in H \Rightarrow D(D) = ^*/\$, a contradiction,
  - $D(D) =$ "yes" $\Rightarrow M_H(D;D) =$ "no" $\Rightarrow D; D \notin H \Rightarrow D(D) = ^*/\$, a contradiction.

**Self-Loop Paradoxes**

**Cantor’s Paradox** (1899): Let $T$ be the set of all sets.
- Then $2^T \subseteq T$, but we know $|2^T| > |T|$

**Russell’s Paradox** (1901): Consider $R = \{A : A \notin A\}$.
- If $R \in R$, then $R \notin R$ by definition.
- If $R \notin R$, then $R \in R$ also by definition,

**Eubulides**: The Cretan says, “All Cretans are liars.”

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

---

**Comments**

- In general, we cannot tell if a running program will ever halt.
- Two levels of interpretations of $M$:
  - A sequence of 0s and 1s (data).
  - An encoding of instructions (programs).
- There are no paradoxes,
  - Concepts should be familiar to computer scientists,
  - Supply a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, etc.

**Axiomatic Set Theory**

- Russell’s paradox initiated the effort to axiomatize set theory in 1908 1929.
- The standard theory is the Zermelo-Fraenkel-Skolem (ZFS) system.\(^a\)
- In ZFS, the Axiom of Foundation says that any descending membership chain is finite.
- Then $x \notin x$ for any set $x$.
  - Otherwise, $x \in x \in x \in \ldots$, a contradiction
- Hence Russell’s paradox is avoided.

---

\(^a\)Ernst Friedrich Ferdinand Zermelo (1871 1953); Adolf Abraham Halevi Fraenkel (1891 1965); Albert Thoralf Skolem (1887 1963),
More Undecidability

- \{M : M \text{ halts on all inputs}\},
- Given \(M; x\), we construct the following machine:
  * \(M_x(y)\) : if \(y = x\) then \(M(x)\) else halt,
- \(M_x\) halts on all inputs if and only if \(M\) halts on \(x\).
- So if the said language were recursive, \(H\) would be recursive, a contradiction.
- This technique is called \textit{reduction}.

Reductions in Proving Undecidability

- Suppose we are asked to prove \(L\) is undecidable,
- Language \(H\) is known to be undecidable.
- We try to find a computable transformation (or reduction) \(R\) such that
  \[ R(x) \in L \text{ if and only if } x \in H. \]
- This suffices to prove that \(L\) is undecidable.

More Undecidability (concluded)

- \(\{M; x : \text{there is a } y \text{ such that } M(x) = y\}\).
- \(\{M; x : \text{the computation } M \text{ on input } x \text{ uses all states of } M\}\).
- \(\{M; x; y : M(x) = y\}\).

Complements of Recursive Languages

\textbf{Lemma 10} If \(L\) is recursive, then so is \(L^c\).

- Let \(L\) be decided by \(M\) (which is deterministic).
- Swap the “yes” state and the “no” state of \(M\).
- The new machine decides \(L^c\).
Recursive and Recursively Enumerable Languages

**Lemma 11** \( L \) is recursive if and only if both \( L \) and \( L' \) are recursively enumerable.

- Suppose both \( L \) and \( L' \) are recursively enumerable, accepted by \( M \) and \( M' \), respectively.
- Simulate \( M \) and \( M' \) in an interleaved fashion,
- If \( M \) accepts, then \( x \in L \) and \( M' \) halts on state “yes,”
- If \( M' \) accepts, then \( x \notin L \) and \( M' \) halts on state “no.”

R, RE, and coRE

**RE:** The set of all recursively enumerable languages.

**coRE:** The set of all languages whose complements are recursively enumerable (note that coRE is not RE).

**R:** The set of all recursive languages.

- \( R = \text{RE} \cap \text{coRE} \) (p. 120).
- There exist languages in RE but not in R or coRE (such as \( H \)).
- There are languages in coRE but not in R or RE (such as \( H \)).
- There are languages in neither RE nor coRE.

A Very Useful Corollary and Its Consequences

**Corollary 12** \( L \) is recursively enumerable but not recursive, then \( L \) is not recursively enumerable.

- Suppose \( L \) is recursively enumerable.
- Then both \( L \) and \( L' \) are recursively enumerable.
- By Lemma 11, \( L \) is recursive, a contradiction.

**Corollary 13** \( R \) is not recursively enumerable.
Notations
- Suppose $M$ is a TM accepting $L$.
- Write $L(M) = L$.
- If $M(x)$ is never “yes” nor $\not\nearrow$ (as required by the definition of acceptance), we define $L(M) = \emptyset$.
- Of course, if $M(x) = \not\nearrow$ for all $x$, then $L(M) = \emptyset$, too.

Rice's Theorem

**Theorem 14 (Rice's theorem)** Suppose $C \neq \emptyset$ is a proper subset of the set of all recursively enumerable languages. Then the question “$L(M) \in C$?" is undecidable.

- Assume that $\emptyset \not\in C$ (otherwise, repeat the proof for the class of all recursively enumerable languages not in $C$).
- Let $L \in C$ be accepted by TM $M_L$ (recall that $C \neq \emptyset$).
- Let $M_H$ accept the undecidable language $H$.
  - $M_H$ exists (p. 111).

Nontrivial Properties of Sets in RE

- A property of a set accepted by a TM (a recursively enumerable set) is **trivial** if it is always true or false.
  - Is an RE set accepted by a TM? Always true.
- It can be defined by the set $C$ of RE sets that satisfy it.
- The property is nontrivial if $C \neq \text{RE}$ and $C \neq \emptyset$.
- Up to now, all nontrivial properties of RE sets are undecidable (pp. 116 117).
- In fact, Rice's theorem confirms that.

The Proof (continued)

- Construct machine $M_x(y)$:
  
  $$ \text{if } M_H(x) = \text{"yes" then } M_L(y) \text{ else } \not\nearrow $$

- We next prove that
  
  $$ L(M_x) \in C \text{ if and only if } x \in H. \quad (2) $$

  - The halting problem has been reduced to deciding $L(M_x) \in C$.
  - Hence $L(M_x) \in C$ must be undecidable, and we are done.
The Proof (concluded)

- Suppose $x \in H$, i.e., $M_H(x) = \text{"yes."}$
  - $M_x(y)$ determines this, and it either accepts $y$ or never halts, depending on whether $y \in L$.
  - Hence $L(M_x) = L \in C$.
- Suppose $M_H(x) \Rightarrow$.
  - $M_x$ never halts.
  - $L(M_x) = \emptyset \notin C$.

Consequences of Rice’s Theorem

**Corollary 15** The following properties of recursively enumerative sets are undecidable.

- **Emptiness.**
- **Finiteness.**
- **Regularity.**
- **Context-freedom.**