Corollary 5 $\text{TIME}^{O^c(f)} \subseteq ((\text{TIME})^c)^<_{\text{TIME}}$.

If none of the paths leads to "yes", then $M$ enters the "no" state.

If some path leads to "yes", then $M$ enters the "yes" state.

Using depth-first search, $M$ does not know $N$.

On input $x$, $M$ goes down every computation path of $N$.

Constant depending on $N$.

Deterministic TM in time $O^c(f)$, where $c < 1$, is some deterministic TM in time $f(n)$. Then it is decided by a $3$-string TM.

Suppose that language $L$ is decided by an NTM.

Simulating Nondeterministic TMs.
A Nondeterministic Algorithm for Graph Reachability

1: if "no" then 14: return

13: end for

12: end if

11: if "no" then

10: else

9: end if

8: \( \hat{h} = x \)

7: else

6: \( \text{Yes, } u \text{ is reached from node } I. \) \{ Node \}

5: if \( u = \hat{h} \) then

4: if \( \langle \hat{h}, x \rangle \in E \) then

3: \{ The next node. \}

2: for \( i = 2 \) to \( \hat{h} \) do

1: \( u = 1 \)
Reachability is in P.

- Reachability also has the same complexity.
- Reachability with more than one terminal node

Hence \( \text{REACHABILITY} \in \text{SPACE}(\log n) \).

- In input string with counters of \( O(\log n) \) bits long.
- Testing if \( (x, y) \in S \) is accomplished by consulting the

Variables \( x, y \) and \( y \) each require \( O(\log n) \) bits.

Space Analysis
Set of squared integers

\( \{1/1, 1/2, 2/1, 1/3, 3/2, 2/3, 3/4, 4/3, 4/2, 2/4, 4/1, \ldots \} \)

- Set of rational numbers
- Set of odd integers
- Set of positive integers
- Set of integers \( \mathbb{N} \)

A set is countable (countably infinite) or

Infinite Sets
\[ |B| > |A| \text{ or } |B| \geq |A| \text{ or } |B| \neq |A| \]

If \( B \subseteq A \), then \( |B| \leq |A| \) but if \( |B| \geq |A| \) and one of \( B \)'s subsets,

\[ |B| \neq |A| \text{ or } |B| \geq |A| \text{ if } |B| > |A| \]

\[ (B \sim A) \text{ if there exists a one-to-one correspondence between } A \text{ and } B \]

Two sets are said to have the same cardinality (written as \( |C| = \) cardinality (size)).

For any set \( C \), define \( |C| \) as \( C \)'s cardinality (size).

\[ |\mathcal{P} A| > |A| \text{ or } |\mathcal{P} A| \geq |A| \text{ or } |\mathcal{P} A| \neq |A| \]

If \( \mathcal{P} A \) is \( \mathcal{P} \) denoted the \( \mathcal{P} \) power set, that is \( \mathcal{P} \subseteq B \), then \( \mathcal{P} A \) denotes \( \mathcal{P} \) power set, that is \( \mathcal{P} A \subseteq B \).

\[ \text{Let } \mathcal{P} A \text{ denote a set } \]

Cardinality
A lot of "paradoxes.

- The set of odd integers.

  - But the set of integers has the same cardinality as the set of odd integers.

  - The set of integers, properly contains the set of odd integers.

\[ |B| = |A| \]

If \( A \) and \( B \) are infinite sets, it is possible that \( A \nsubseteq B \).

Cardinality and Infinite Sets
The new customer occupies Room I.

... the person from Room 2 into Room 3, and so on...

... the person previously occupying Room I into Room 2,

... "But of course" exclaims the proprietor, and he moves...

A new guest comes and asks for a room.

rooms, and all the rooms are occupied.

Now let us imagine a hotel with an infinite number of...

rooms occupied, a new guest will be turned away.

For a hotel with a finite number of rooms with all the...

Hilbert’s Paradox of the Grand Hotel.
Going to prepare a place for you." John 14:3.

There are many rooms in my Father's house, and I am

infinity of new guests can be accommodated in them.

Now all odd-numbered rooms become free and the

occupant of Room 2 into Room 4, and so on.

He moves the occupant Room 1 into Room 2, the

minute.

"Certainly, Gentlemen," says the proprietor, "Just wait a

guests who come in and ask for rooms.

all taken up, and an infinite number of new

rooms, let us imagine now a hotel with an infinite number of

Hilbert's Paradox of the Grand Hotel (continued)
Galileo (1564–1642)'s Paradox (1638)

Resolution of paradoxes: Which notion results in better

- greater than any of its proper parts.

- This is contrary to the axiom of Euclid that the whole is
  one-to-one correspondence with all the positive integers.

- The squares of the positive integers can be placed in
George Cantor (1845-1918)

f is not a bijection, a contradiction.

Hence $B$ for any $n$.

$B$

If $n \in B$, then $n \in B$, but then $n \not\in B$ by the definition of $B$.

Suppose that $B = (u)f$ for some $n$.

Consider the set $B$ of $n \in N$ for which $(u)f \not\in B$.

Suppose it is countable with $f : N \rightarrow N \subset N$ being a bijection.

The set of all subsets of $N$ is infinite and not countable.

Theorem 6

Cantor’s Theorem
\[ |\mathbb{N}| \text{ cardinality} \]

So the set of functions from \( \mathbb{N} \) to \( \{0, 1\} \) has \( 2^\mathbb{N} \) elements.

That \( u \) iff \( \mathcal{N} \supseteq \mathcal{M} \) and only if \( \mathcal{N} \subseteq \mathcal{M} \) determine an \( f \) \( \mathcal{N} : f \) \( |\mathbb{N}| \text{ function} \) is not countable.

The set of all functions on \( \mathbb{N} \) is not countable.

Theorem.

The inequality uses the same proof as Cantor's.

\[ \mathcal{L} \text{ maps } \{x\} = (x) \text{ as } |\mathcal{L}| \geq |\mathcal{L}| \]

\[ |\mathcal{L}| > |\mathcal{L}| \]

For any set \( \mathcal{L} \), finite or infinite,

Two Corollaries:
Existence of Uncomputable Problems

Programs by the second corollary above. So there must exist functions for which there are no
A function is a mapping from integers to integers.
The set of programs is countable.
Every program corresponds to some integer.

Every program is a sequence of 0s and 1s.
We skip the details of $\mathcal{N}$. 

execute any valid Java bytecode.

Think of $\mathcal{M}$ as a modern computer, which can execute any valid machine code, or a Java Virtual Machine, which can simulate any $\mathcal{M}$ on $x$ so that $\mathcal{M}$ simulates $\mathcal{N}$.

Both $\mathcal{M}$ and $x$ are over the alphabet of $\mathcal{N}$. 

An input to that machine, $x$, an input to that machine, with the description of a TM $\mathcal{M}$ concatenated with the description of a universal Turing machine $\mathcal{M}$ interprets the input as the

Universal Turing Machine.
- Does $\mathcal{M}$ halts on input $x$?

\[ \{ \forall \not= \ (x) \mathcal{M} : x \not\in \mathcal{M} \} = H \]

**Halting problem:**

We now define a concrete undecidable problem, the halting problem. We already know undecidable problems must exist. Algorithms or languages that are not recursive.

**Undecidable problems** are problems that have no

**The Halting Problem**
By asking, \( \mathcal{H} \in \mathcal{W} \), the question can be answered.

Comment: Membership of \( x \) in any recursively enumerable

- \( \mathcal{H} \) accepts \( \mathcal{W} \).
- When \( \mathcal{W} \) is about to halt, \( \mathcal{W} \) enters a "yes" state.
- Use the universal \( \mathcal{T} \) M to simulate \( \mathcal{W} \) on \( x \).

Proposition: \( \mathcal{H} \) is recursively enumerable.
\(\sqrt{\alpha \gamma} = (\alpha \gamma)\gamma \)

\(\vdash H \not\ni \alpha \gamma \leftrightarrow \text{"\textcolor{red}{\alpha \gamma}}\text{"} = (\alpha \gamma \not\ni \alpha \gamma \) - a contradiction.

\(\neg = (\alpha \gamma \not\ni \alpha \gamma \) - a contradiction.

\(\vdash H \ni \alpha \gamma \leftrightarrow \text{"\textcolor{green}{\alpha \gamma}}\text{"} = (\alpha \gamma \ni \alpha \gamma \) - a contradiction.

\(\vdash \) Consider now \(\alpha \gamma \)
Java compiler, etc.

- Interpret to a Lisp interpreter, a Java compiler to a
  Lisp
- Supply a C compiler to a C compiler, a Lisp

Concepts are familiar to computer scientists (but not

There are no paradoxes.

- An encoding of instructions (programs).
- A sequence of 0s and 1s (data).

Two levels of interpretation of $M$: Comments
Sharon Stone, *The Specialist*: "I am not a woman. You can trust.

Eubulides: *The Greek says*, "All Cretans are liars."

- If $S$, then $S \not \in S$ also by definition.
- If $S \not \in S$, then $S \not \in S$ by definition.

$\{ v \not \in v : v \} = S$.

Russell's Paradox (1901): Consider $\exists ! \mathcal{L}$

$|\mathcal{L}| < |\mathcal{P}|$.

But we know $|\mathcal{P}| \geq |\mathcal{L}|$.

Let $\mathcal{L}$ be the set of all sets.

Cantor's Paradox (1899):

Self-Loop Paradoxes
\{\tilde{h} = (x)_{W : \tilde{h}}, x : W\} \Rightarrow \\
\{W \text{ the computation on input } x \text{ uses all states of } W\} \Rightarrow \\
\{\tilde{h} = (x)_{W \text{ there is a such that } \tilde{h} : x \text{ W}}\} \Rightarrow \\

This technique is called reduction.

Recursively, a contradiction.

So if the said language were recursive, \( \mathcal{H} \) would be

\( \mathcal{H} \text{ halts on all inputs if and only if it halts on } x \).

\( (x)_{W \text{ if then } x = \tilde{h} : (\tilde{h})_{W} \text{ else halt.}} \)

Given \( x \), we construct the following machine:

\{W \text{ halts on all inputs : W}\} \Rightarrow \\

More Undecidability
If \( M \) accepts, then \( M \) halts on state "no."

- If \( M \) accepts, then \( M \) halts on state "yes."

Simultaneously and in an interleaved fashion, accepted by \( M \) and \( M' \), respectively, \( M \) and \( M' \) are recursively enumerable,

- Suppose both \( M \) and \( M' \) are recursively enumerable.

\( \mathcal{L} \) is recursively if and only if both \( M \) and \( M' \) are recursively enumerable.

(Prop. 60)

Can’t work for recursively enumerable languages.

- The "no" state of \( M \) results in a TM that decides \( \mathcal{L} \).

- \( \mathcal{L} \) is decided by \( M \), swapping the "yes" state and

- \( \mathcal{L} \) is recursively, then so is \( \mathcal{L} \).
There are languages in $\text{RE}$ but not in $R$ or $\overline{R}$.

\(\text{RE} \cap \overline{R} \neq \emptyset\).

There exist languages in $\overline{\text{RE}}$ but not in $R$ or $\overline{R}$.

$\text{RE} \cap \overline{\text{RE}} = \emptyset$.

$R$: The set of all recursive languages.

$\text{RE}$: The set of all recursively enumerable languages.

$\overline{R}$: $R$'s complement.

$\overline{\text{RE}}$: The set of all recursively enumerable languages whose complements are not in $\text{RE}$.

$\text{COR}$: The set of all languages whose complements are not in $\text{RE}$.

$H$: Such as $H$. (such as $H$.)
Then the question "\( T \in \mathcal{C} \)" is undecidable.

Theorem 8 (Rice's Theorem) Suppose that \( \mathcal{C} \) is an undecidable

Rice's theorem says any nontrivial property of TMs is

\( \emptyset = (\mathcal{W} T) (x) \) if neither "yes" nor "no" is required by the

Write \( T = (\mathcal{W} T) \) if \( \mathcal{W} \) is a TM accepting \( T \).

Suppose \( \mathcal{W} \) is a TM accepting \( T \).
reduced to deciding $T(x) \in C$.

Then we are done because the halting problem has been

\[ (\exists x \in T) \quad H \in C \iff \langle x \rangle \in C \quad \text{and only if} \quad x \in T \]

If we can prove that

\[ \exists \text{ yes } \text{ if } \langle x \rangle \in T \quad \text{and yes } \text{ if } \langle x \rangle \notin T \]

Consider machine $H$.

Let $M$ accept the undecidable language $H$.

\[ \emptyset \neq T \quad \text{let } T \in \text{TM} \quad \text{and that } (\forall C \in \text{TM} \text{ such that } C \text{ is decidable in class } C) \]

Assume that $C \not\in \emptyset$.

The Proof
\[ \mathcal{C} \not\supset \emptyset = (xW)T \]

- \( xW \) never halts.

\[ \checkmark = (x)H_W \text{ Suppose that } \]

\[ \mathcal{C} \ni T = (xW)T \text{ Hence } \]

- \( \mathcal{C} \equiv \emptyset \text{ depending on whether } \checkmark \text{ \( \in \) } T. \)

- \( H \) determines this, and it either accepts orㆍ

\[ \text{"Yes."} = (x)H_W \text{ Suppose that } x \in \mathcal{H}, \text{ i.e., } \]

\[ \text{We proceed to prove claim (1).} \]

The proof (continued)
\[
\begin{align*}
\neg \text{Boolean Logic (1815-1864)}, 1847. \\
& (\forall \phi \iff \exists \phi) \lor (\exists \phi \iff \forall \phi) \\
& \text{Boolean Conjunctions: } \forall \exists \phi \iff \exists \forall \phi \\
& \text{Impllications: } \forall \exists \phi \iff \exists \forall \phi \\
& \forall \exists \phi \land \exists \forall \phi \iff \exists \forall \phi \\
& \exists \forall \phi \land \forall \exists \phi \iff \forall \exists \phi \\
& \text{Boolean Expressions: } \exists \forall \phi \iff \forall \exists \phi \\
& \text{Boolean Conjunctions: } \land, \lor, \iff \\
& \text{Literals: } x, \neg x, \exists x, \forall x \\
& \text{Boolean Variables: } x_1, x_2, \ldots \\
\end{align*}
\]
\[ (\phi \iff \phi_1) = \models \text{Equivalently, } \]
\[ \phi \iff \phi_2 = \models \text{if and only if } \]
\[ \phi = \models \text{true assignment appropriate to both of them, } \]
\[ \phi_1 = \models \phi_2, \text{ if for any } \]
\[ \text{truth assignment } \models \text{ are equivalent, written } \phi \equiv \phi_1 \text{ and } \phi_2, \text{ if for any } \]
\[ \phi \text{ satisfies } \models \text{ in other words, } \]
\[ \phi \text{ satisfies } \models \text{ in } \text{if it defines the truth value for every } \]
\[ \phi \text{ true assignment is appropriate to boolean expression } \]
\[ \text{A truth assignment is a mapping from boolean variables to truth values true and false. } \]
\[ \text{A truth assignment is } \models \text{ from boolean variables. } \]

**Truth Assignments**
\( \varphi \land \phi \land \psi = (\varphi \land \psi) \land \phi \land \psi \)

\( \varphi \land \phi \land \psi = (\varphi \land \psi) \land \phi \land \psi \)

**De Morgan's Laws** say that expressions are equivalent.

A truth table can be used to prove if two boolean expressions are equivalent. A truth table contains \( 2^n \) rows, one for each possible truth assignment of the \( n \) variables together with the truth value of \( \phi \) under that truth assignment.

Suppose \( n \) boolean variables.

**Truth Tables**
\[ (\exists x \lor \exists x) \land (\exists x \lor \exists x) \land (\exists x \lor \exists x) - \]

- conjunction of one or more literals.

\[ \Gamma \text{ where each } \exists \text{ is the } DNF \]

- boolean expression is in disjunctive normal form.

\[ (\exists x \land \exists x) \lor (\exists x \land \exists x) \lor (\exists x \land \exists x) - \]

- the disjunction of one or more literals.

\[ \text{form (CNF) if } \Gamma \text{ where each clause } \exists \text{ is} \]

- boolean expression is in conjunctive normal form.

**Normal Forms**
Similiarly: \( \phi \lor \bot \phi = \phi \)

\[
\begin{align*}
\forall x \in \{0,1\} \forall y \in \{0,1\} \forall z \in \{0,1\} & \quad \phi \lor \top \phi = \phi \\
\text{Since } \phi \text{ is a CNF, set } \phi \text{ equal to } \phi \text{ and a DNF is sought: Make } \phi \text{ and } \phi \\
\phi \text{ is a DNF, let } \phi \text{ and apply de Morgan's laws to make a CNF for } \phi \\
\text{Turn } \phi \text{ into a CNF and apply de Morgan's laws to make a DNF for } \phi \\
\text{Turn } \phi \text{ into a DNF and apply de Morgan's laws to make a CNF for } \phi \\
\text{This is trivially true.}
\end{align*}
\]

Any expression \( \phi \) can be converted into CNFs and DNFs.
appropriate truth assignments if and only if \( \phi \) is valid.

\( \phi \) is \textit{unsatisfiable} if and only if \( \phi \) is false under all

\[ \phi \text{ is valid or a tautology} \]

\[ \phi \text{ is a truth assignment} \]

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

A boolean expression is \textit{satisfiable} if there is a truth
A most important problem in answering the $P \neq NP$ question is the SAT

$\text{SATISFIABILITY} \ (\text{SAT})$.
unsatisfiability, validity— are known to be in $P$.

None of the three problems—satisfiability, expression, and SAT— are valid.

The negation of an unsatisfiable expression is a valid.

Relations among SAT, unsAT, and Validity.
If a Horn clause has no positive literals, we keep its non-implication form, \( \neg x \land \cdots \land \neg x \land \neg x \land \neg x \land \forall \).

- If \( m = 0 \), use \( \forall \) also in implication form, where \( \forall \) is the positive literal.

\[
\forall \iff (\forall x \lor \cdots \lor \forall x \lor \forall x)
\]

rewritten as an implication

\[
\forall \iff (\forall x \lor \cdots \lor \forall x \lor \forall x \land \forall x
\]

A Horn clause can be \( \forall \) as an implication

\[
\forall \iff (\forall x \land \cdots \land \forall x \land \forall x \land \forall x)
\]

A Horn clause is a clause with at most one positive literal

Horn Clauses
Satisfiability of CNFs with Horn Clauses Is in P

- Interpret a truth assignment as a set $T$ of those variables that are assigned true.
- Let $\phi$ be a conjunction of Horn clauses.

$T \models x_i$ if and only if $x_i \in T$. 
The Algorithm

1: While not all implications are satisfied do
2: \( \forall \) variables are false.
3: Pick an unsatisfied \((x_1 \lor \cdots \lor x_m)\).
4: Add \( \phi \) to \( \mathcal{L} \).
\( \forall \) \( \phi \) \text{ true}.
5: \text{end while}
6: If \( \mathcal{L} \) then \( \phi = \mathcal{L} \).
7: Return \( \phi \) is satisfiable.
8: Else
9: Return \( \phi \) is unsatisfiable.
10: end if
Analytic of the Algorithm

\[ L \subseteq \{ w_x \mid \cdots \} \text{ then } w_x \land \cdots \land \exists x \land \exists x \not\models L \]

If no supersets of \( L \) can satisfy this clause, which means

\[ L \equiv \{ w_x \mid \cdots \} \text{ then } w_x \land \cdots \land \exists x \land \exists x \not\models L \]

Then \( L \) cannot be satisfied by \( L \)

This implies that the implication that causes insertion of a new fact at which the first time in the execution of the algorithm

Otherwise, the execution of the algorithm must be such that

\[ L \equiv \{ \} \]

is satisfied.

By the time the while loop exits, all implications are

Implications (but not necessarily all Horn clauses) are

Eventually, it will be large enough to make all size and eventually it will be large enough to make all

It will terminate, because \( L \) is monotonically increasing in

Analytic of the Algorithm
A boolean function expresses a boolean expression.

A boolean expression expresses a boolean function.

Each of the $2^n$ truth assignments can be true or false.

There are $2^n$ such boolean functions.

It can be represented by a truth table.

An $n$-ary boolean function is a function of

Boolean Functions
A boolean circuit computes a boolean function.

The output gate(s) has no outgoing edges.

Of C and have an indegree of zero.

C Gates of sort from True, False, \( x_1, x_2, \ldots \) are the inputs.

\{True, False\} \( \land, \lor, \neg, x_1, x_2, \ldots \).

Each gate has a sort from 0, 1, 0, or 2.

All nodes have indegree (number of incoming edges) equal to 0, 1, or 2.

There can be no cycles in C.

A boolean circuit is a graph C whose nodes are the gates.
One can construct one from the other.

They are equivalent representations.

Boolean Circuits and Expressions
Circuits are more economical because of sharing.

\[

((\neg x \land \neg x) \implies ((\neg x \land \neg x) \lor (\neg x \lor x))
\]

An Example
Given a circuit, is there a truth assignment such that the circuit outputs true?

Circuit SAT: Given a circuit, is there a truth assignment such that the circuit satisfies the circuit value?

Circuit SAT and Circuit Value

Circuit SAT and Circuit Value

...
can be improved to "almost all boolean functions..."

\[ \forall u \in \mathbb{Z} \quad u > \left( \frac{u}{2^\lfloor \log_2 u \rfloor} - 1 \right) u \mathbb{Z} = (\mathbb{Z} \times (\mathbb{Z} + u)) \mathbb{Z} \quad \text{for } u \in \mathbb{Z} \]

But \( u \mathbb{Z} / u \mathbb{Z} = u \) when \( u \mathbb{Z} > u(\mathbb{Z} \times (\mathbb{Z} + u)) \)

There are at most \( \mathbb{Z}^n \) boolean circuits with \( n \)-ary boolean functions.

There are \( \mathbb{Z} \) different \( n \)-ary boolean functions.

Theorem 9 (Shannon, 1949) For any \( n \geq 2 \), there is an 

Some Boolean Functions Need Exponential Circuits