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

- NTIME($f(n)$) is the set of languages decided by NTMs within time $f(n)$.
- NTIME($f(n)$) is a complexity class.

NP

- Define
$$\mathrm{NP} = \bigcup_{k>0} \mathrm{NTIME}(n^k).$$
- Clearly $\mathrm{P} \subseteq \mathrm{NP}$.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (p. 87 and p. 154).
- The most important open problem in computer science is whether $\mathrm{P} = \mathrm{NP}$.

Simulating Nondeterministic TMs

Surprisingly, nondeterminism does not add power to TMs.

Theorem 4 *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.
  - M does *not* need to know $f(n)$.
  - As N is time-bounded, the depth-first search will not run indefinitely.

The Proof (concluded)

- 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.
- Note that every path has a finite length by definition.

Corollary 5 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

NTIME vs. TIME

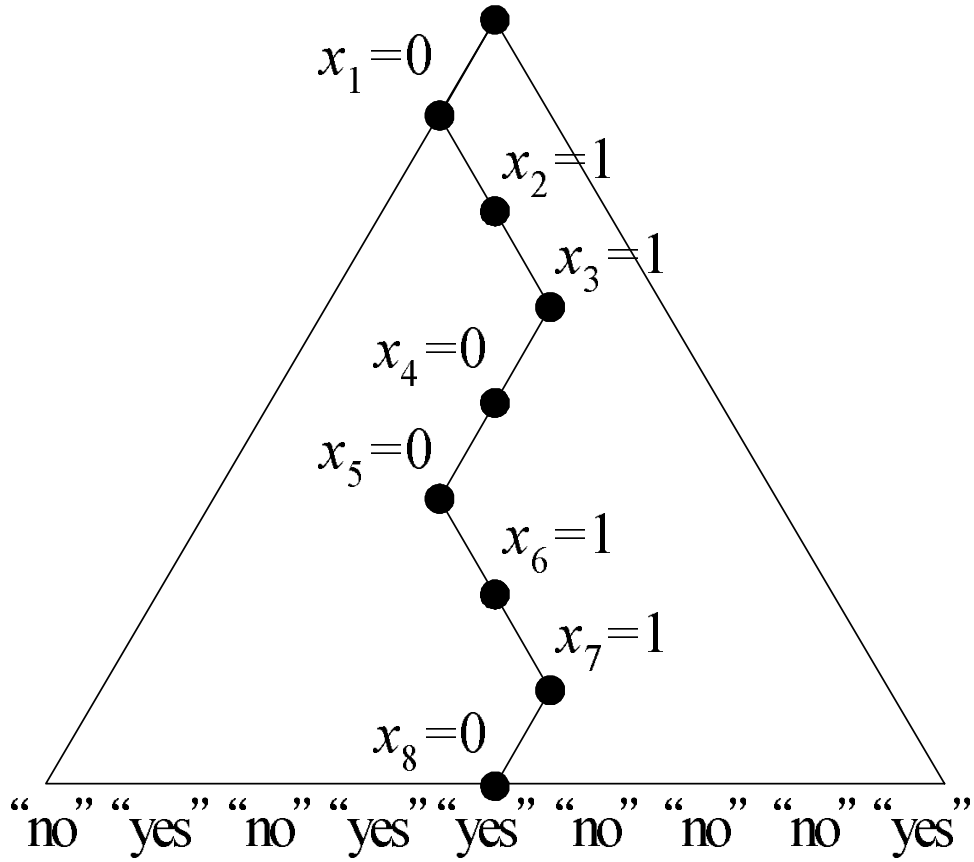
- Does converting an NTM into a TM require exploring all of the computation paths of the NTM as done in Theorem 4 (p. 84)?
- This is the most important question in theory with practical implications.

A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

1: **for** $i = 1, 2, \ldots, n$ **do**
2: 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: "yes";
7: **else**
8: "no";
9: **end if**

The Schematic Computation Tree for Satisfiability



Analysis

- The algorithm decides language $\{\phi : \phi \text{ is satisfiable}\}$.
  - The computation tree is a complete binary tree of depth n.
  - Every computation path corresponds to a particular truth assignment out of 2^n.
  - ϕ is satisfiable iff there is a truth assignment that satisfies ϕ.
  - But there is a truth assignment that satisfies ϕ iff there is a computation path that results in "yes."
- General paradigm: Guess a "proof" and 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.
- The decision version TSP (D) asks if there is a tour with a total distance at most B, where B is an input.
- Both problems are extremely important and are equally hard (p. 333 and p. 428).

A Nondeterministic Algorithm for TSP (D)

1: **for** $i = 1, 2, \ldots, n$ **do**
2: Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}[a]
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: "yes";
8: **else**
9: "no";
10: **end if**

[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 that leads to "yes."[a]
- 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 just means such a computation path exists.

Remarks[a] on $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$

- Verification of password is 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).
- Many practical applications depend on answers to the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question.

---

[a]Contributed by Mr. Kuan-Lin Huang (`B96902079`, `R00922018`) on September 27, 2011.

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 $f(n)$.
- NSPACE$(f(n))$ is a set of languages.
- As in the linear speedup theorem (Theorem 3 on p. 64), constant coefficients do not matter.

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?

The First Try: 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 ith 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 REACHABILITY $\in$ NSPACE$(\log n)$

1: $x := a$;
2: **for** $i = 2, 3, \ldots, n$ **do**
3: Guess $y \in \{v_1, v_2, \ldots, v_n\}$; {The next node.}
4: **if** $(x, y) \notin E$ **then**
5: "no";
6: **end if**
7: **if** $y = b$ **then**
8: "yes";
9: **end if**
10: $x := y$;
11: **end for**
12: "no";

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

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

  - REACHABILITY with more than one terminal node also has the same complexity.
- REACHABILITY $\in$ P (p. 193).

Undecidability

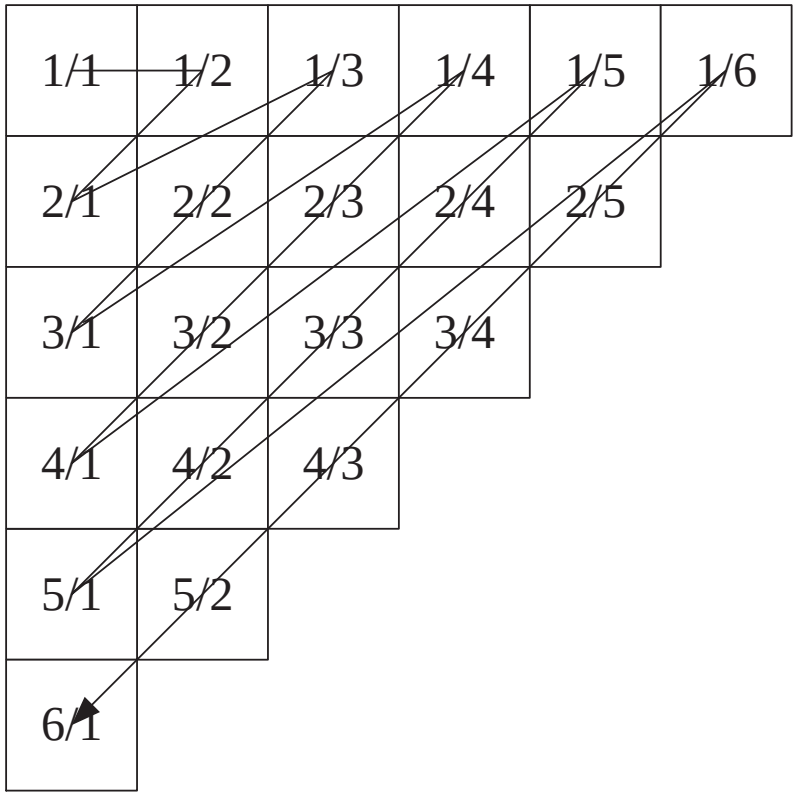
God exists since mathematics is consistent,
and the Devil exists since we cannot prove it.
— André Weil (1906–1998)

Whatsoever we imagine is *finite*.
Therefore there is no idea, or conception
of any thing we call *infinite*.
— Thomas Hobbes (1588–1679), *Leviathan*

Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-one correspondence with $\mathbb{N} = \{0, 1, \ldots\}$, 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: $\sqrt{i} \leftrightarrow i$.

Rational Numbers Are Countable



Cardinality

- For any set A, define $|A|$ as A's **cardinality** (size).
- Two sets are said to have the same cardinality, or

$$|A| = |B| \quad \text{or} \quad A \sim B,$$

  if there exists a one-to-one correspondence between their elements.
- 2^A denotes set A's **power set**, that is $\{B : B \subseteq A\}$.
  - E.g., $\{0, 1\}$'s power set is $2^{\{0,1\}} = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$.
  - If $|A| = k$, then $|2^A| = 2^k$.

Cardinality (concluded)

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

Cardinality and Infinite Sets

- If A and B are infinite sets, it is possible that $A \subsetneq B$ 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. 101).
- A lot of "paradoxes."

Galileo's[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[b] 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 opinions.

---

[a] Galileo (1564–1642).

[b] Euclid (325 B.C.–265 B.C.).

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 imagine a hotel with an infinite number of rooms, all of which are occupied.
- A new guest comes and asks for a room.
- "But of course!" exclaims the proprietor.
- He moves the person previously occupying Room 1 to Room 2, the person from Room 2 to Room 3, and so on.
- The new customer now occupies Room 1.

[a] David Hilbert (1862–1943).

Hilbert's Paradox of the Grand Hotel (concluded)

- Now imagine a hotel with an infinite number of rooms, all taken up.
- An infinite number of new guests come in and ask for rooms.
- "Certainly," says the proprietor.
- He moves the occupant of Room 1 to Room 2, the occupant of Room 2 to 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)

David Hilbert (1862–1943)

Cantor's Theorem

Theorem 6 *The set of all subsets of* $\mathbb{N}$ $(2^{\mathbb{N}})$ *is infinite and not countable.*

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

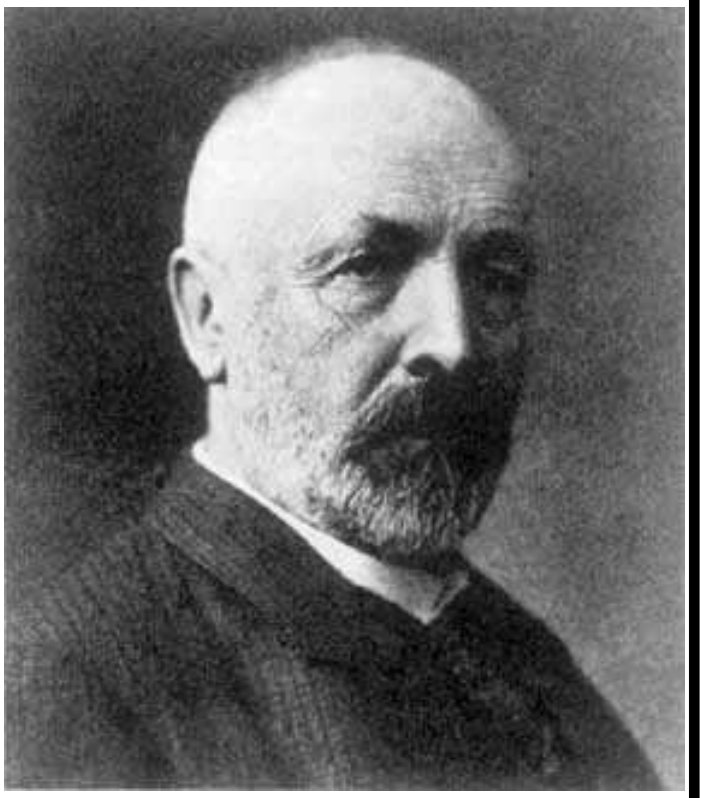
[a] Note that $f(k)$ is a subset of $\mathbb{N}$.

The Proof (concluded)

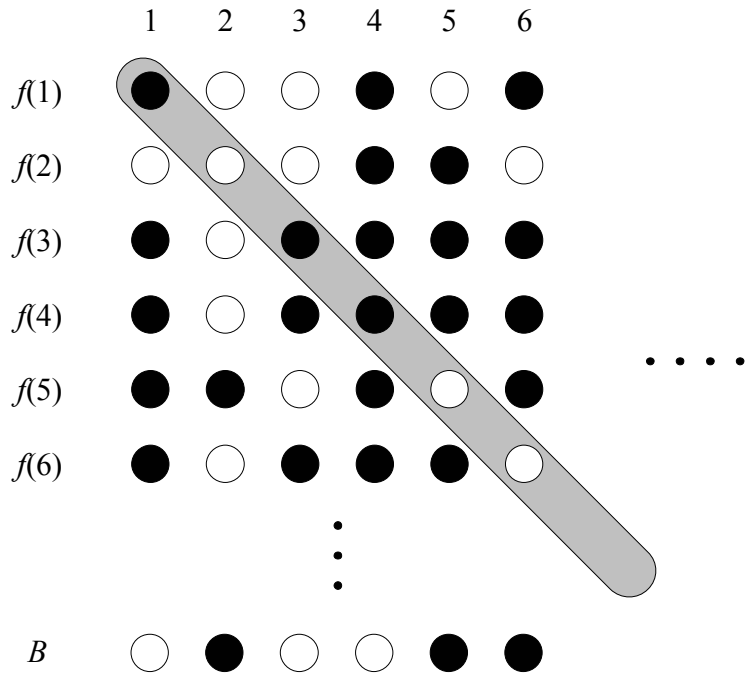
- If $n \in f(n) = B$, then $n \in B$, but then $n \notin B$ by B's definition.
- If $n \notin f(n) = B$, then $n \notin B$, but then $n \in B$ by B's definition.
- Hence $B \neq f(n)$ for any n.
- f is not a bijection, a contradiction.

Georg Cantor (1845–1918)

Kac and Ulam, "[If] one had to name a single person whose work has had the most decisive influence on the present spirit of mathematics, it would almost surely be Georg Cantor."



Cantor's Diagonalization Argument Illustrated



A Corollary of Cantor's Theorem

Corollary 7 *For any set T, finite or infinite,*

$$|\,T\,| < |\,2^T\,|.$$

- The inequality holds in the finite T case as $k < 2^k$.
- Assume T is infinite now.
- To prove $|\,T\,| \leq |2^T|$, simply consider $f(x) = \{x\} \in 2^T$.
  - f maps a member of $T = \{\,a, b, c, \ldots\,\}$ to a corresponding member of $\{\,\{\,a\,\}, \{\,b\,\}, \{\,c\,\}, \ldots\,\} \subseteq 2^T$.
- To prove the strict inequality $|\,T\,| \lneq |2^T|$, we use the same argument as Cantor's theorem.

A Second Corollary of Cantor's Theorem

Corollary 8 *The set of all functions on* $\mathbb{N}$ *is not countable.*

- It suffices to prove it for functions from $\mathbb{N}$ to $\{0, 1\}$.
- Every function $f : \mathbb{N} \to \{0, 1\}$ determines a subset of $\mathbb{N}$:

$$\{n : f(n) = 1\} \subseteq \mathbb{N},$$

  and vice versa.
- So the set of functions from $\mathbb{N}$ to $\{0, 1\}$ has cardinality $|\, 2^{\mathbb{N}} \,|$.
- Corollary 7 (p. 114) then implies the claim.

Existence of Uncomputable Problems

- Every program is a finite sequence of 0s and 1s, thus a nonnegative integer.[a]
- Hence every program corresponds to some integer.
- The set of programs is countable.

[a]Different binary strings may be mapped to the same integer (e.g., "001" and "01"). To prevent it, use the lexicographic order as the mapping or simply insert "1" as the most significant bit of the binary string before the mapping (so "001" becomes "1001"). Contributed by Mr. Yu-Chih Tung (R98922167) on October 5, 2010.

Existence of Uncomputable Problems (concluded)

- A function is a mapping from integers to integers.
- The set of functions is not countable by Corollary 8 (p. 115).
- So there are functions for which no programs exist.[a]

[a] As a nondeterministic program may not compute a function, we consider only deterministic programs for this sentence. Contributed by Mr. Patrick Will (A99725101) on October 5, 2010.

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.
  - 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).

The Halting Problem

- **Undecidable problems** are problems that have no algorithms or languages that are not recursive.
- We knew undecidable problems exist (p. 116).
- We now define a concrete undecidable problem, the **halting problem**:

$$H = \{M; x : M(x) \neq \nearrow\}.$$

  - Does M halt on input x?

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.
  - E.g., membership of x in a recursively enumerative 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:     ↗; {Writing an infinite loop is easy, right?}
3: else
4:     "yes";
5: end if
```

- Consider $D(D)$:
  - $D(D) = \nearrow \Rightarrow M_H(D; D) =$ "yes" $\Rightarrow D; D \in H \Rightarrow$ $D(D) \neq \nearrow$, a contradiction.
  - $D(D) =$ "yes" $\Rightarrow M_H(D; D) =$ "no" $\Rightarrow D; D \notin H \Rightarrow$ $D(D) = \nearrow$, a contradiction.

Comments

- 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.
  - Feed a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, etc.

Self-Loop Paradoxes

Cantor's Paradox (1899): Let T be the set of all sets.[a]

- Then $2^T \subseteq T$ because 2^T is a set.
- But we know[b] $|\,2^T\,| > |\,T\,|$ (p. 114)!
- We got a "contradiction."
- So what gives?
- Are we willing to give up Cantor's theorem?
- If not, what is a set?

---

[a] Recall this ontological argument for the existence of God by St Anselm (–1109) in the 11th century: If something is possible but is not part of God, then God is not the greatest possible object of thought, a contradiction.

[b] Really?

Self-Loop Paradoxes (continued)

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.
- In either case, we have a "contradiction."

Eubulides: The Cretan says, "All Cretans are liars."

Liar's Paradox: "This sentence is false."

Hypochondriac: a patient (like Gödel) with imaginary symptoms and ailments.

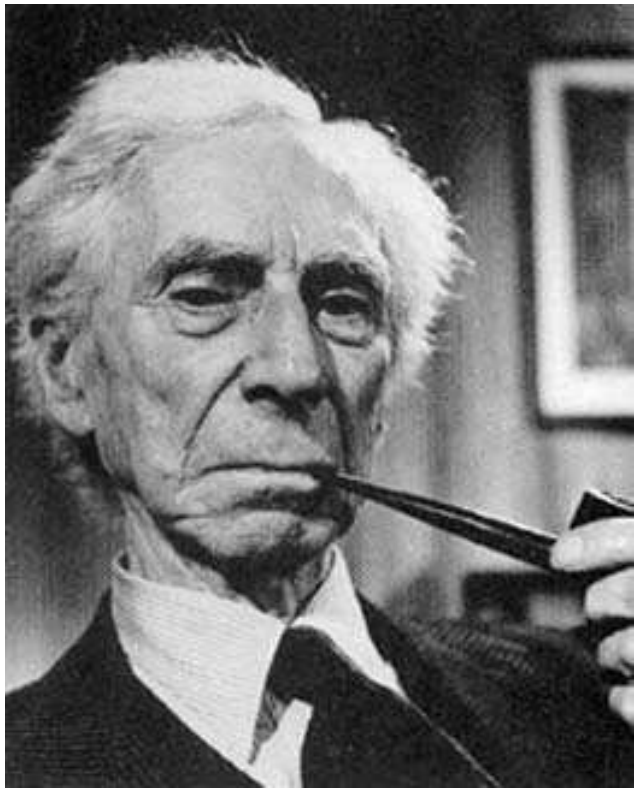
Self-Loop Paradoxes (concluded)

Sharon Stone in *The Specialist* (1994): "I'm not a woman you can trust."

Spin City (1996–2002): "I am not gay, but my boyfriend is."

Numbers 12:3, Old Testament: "Moses was the most humble person in all the world [$\cdots$]" (attributed to Moses).

Bertrand Russell (1872–1970)



Reductions in Proving Undecidability

- Suppose we are asked to prove L is undecidable.
- Language H is known to be undecidable.
- We then try to find a computable transformation (called **reduction**) R such that[a]

$$\forall x \ \{R(x) \in L \text{ if and only if } x \in H\}.$$

- Now we can answer "$x \in H$?" for *any* x by asking "$R(x) \in L$?" instead.
- If L were decidable, H would be decidable, a contradiction!
- So L must be undecidable.

[a] Contributed by Mr. Tai-Dai Chou (J93922005) on May 19, 2005.