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 such 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 $\mathbb{N}$ to $\{0,1\}$ has cardinality $|\, 2^{\mathbb{N}} \,|$.
- Corollary 7 (p. 116) 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] Use lexicographic order or other tricks to prevent two binary strings from being mapped to the same integer. 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. 117).
- So there are functions for which no programs exist.[a]

---

[a] As a nondeterministic program may not be said to compute a function, we consider only deterministic programs here. 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. 118).
- 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:

```
if M_H(M; M) = "yes" then
    ↗; {Writing an infinite loop is easy, right?}
else
    "yes";
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.

©2010 Prof. Yuh-Dauh Lyuu, National Taiwan University

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 $| 2^T | > | T |$ (p. 116)!
- 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.

©2010 Prof. Yuh-Dauh Lyuu, National Taiwan University

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.

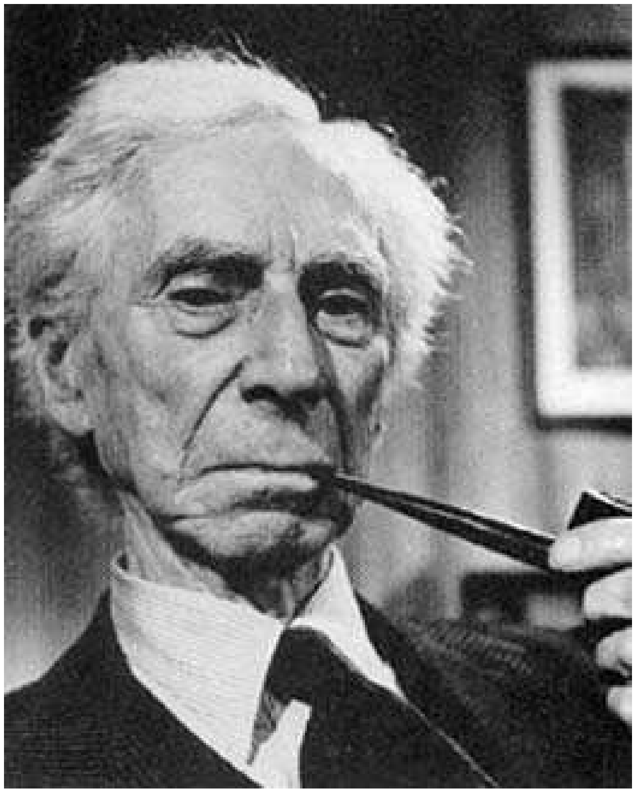
Self-Loop Paradoxes (concluded)

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]

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

- We can answer "$x \in H$?" for *any* x by asking "$R(x) \in L$?" instead.
- This suffices to prove that L is undecidable.

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

More Undecidability

- $H^* = \{M : M \text{ halts on all inputs}\}$.
  - Given the question "$M; x \in H$?" we construct the following machine:[a]

$$M_x(y) : M(x).$$

  - M_x halts on all inputs if and only if M halts on x.
  - In other words, $M_x \in H^*$ if and only if $M; x \in H$.
  - So if H^* were recursive, H would be recursive, a contradiction.

---

[a]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 (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

Lemma 9 *If L is recursive, then so is $\bar{L}$.*

- Let L be decided by M (which is deterministic).
- Swap the "yes" state and the "no" state of M.
- The new machine decides $\bar{L}$.

Recursive and Recursively Enumerable Languages

Lemma 10 *L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.*

- Suppose both L and $\bar{L}$ are recursively enumerable, accepted by M and $\bar{M}$, respectively.
- Simulate M and $\bar{M}$ in an *interleaved* fashion.
- If M accepts, then $x \in L$ and M' halts on state "yes."
- If $\bar{M}$ accepts, then $x \notin L$ and M' halts on state "no."

A Very Useful Corollary and Its Consequences

Corollary 11 *L is recursively enumerable but not recursive, then $\bar{L}$ is not recursively enumerable.*

- Suppose $\bar{L}$ is recursively enumerable.
- Then both L and $\bar{L}$ are recursively enumerable.
- By Lemma 10 (p. 132), L is recursive, a contradiction.

Corollary 12 *$\bar{H}$ is not recursively enumerable.*

©2010 Prof. Yuh-Dauh Lyuu, National Taiwan University

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 $\overline{\mathrm{RE}}$).

- $\mathrm{coRE} = \{ L : \overline{L} \in \mathrm{RE} \}$.
- $\overline{\mathrm{RE}} = \{ L : L \notin \mathrm{RE} \}$.

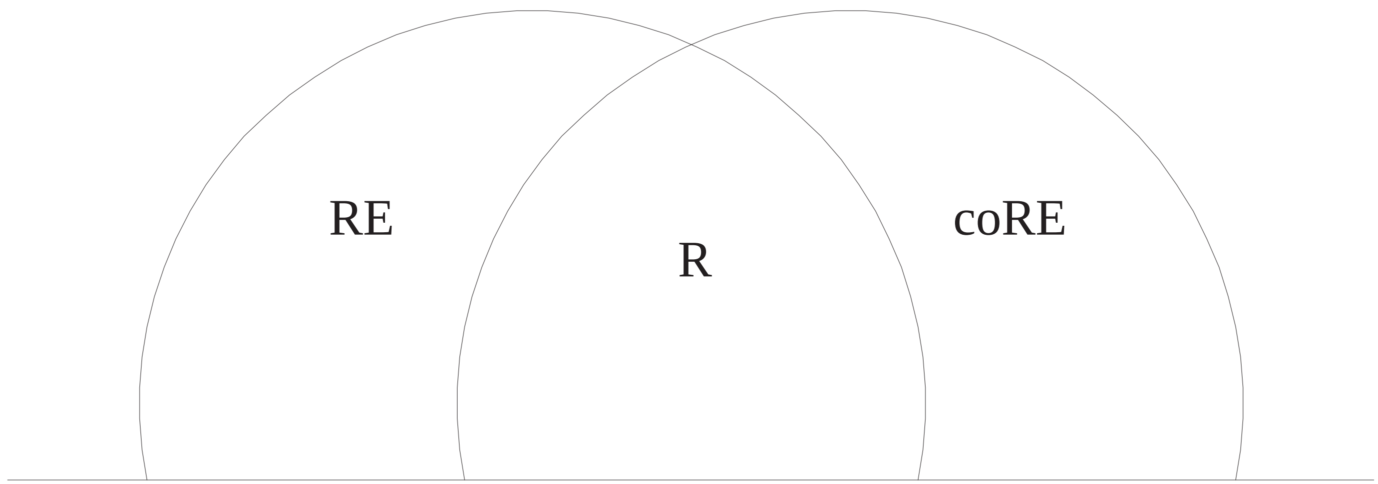
R: The set of all recursive languages.

©2010 Prof. Yuh-Dauh Lyuu, National Taiwan University

R, RE, and coRE (concluded)

- R = RE $\cap$ coRE (p. 132).
- There exist languages in RE but not in R and not in coRE.
  - Such as H (p. 121, p. 122, and p. 133).
- There are languages in coRE but not in RE.
  - Such as $\bar{H}$ (p. 133).
- There are languages in neither RE nor coRE.

©2010 Prof. Yuh-Dauh Lyuu, National Taiwan University


RE
R
coRE