The point of philosophy is
to start with something so simple
as not to seem worth stating,
and to end with something
so paradoxical that no one will believe it.
— Bertrand Russell (1872–1970)

Cantor's Theorem (1895)

Theorem 9 *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 $n \in \mathbb{N}$.

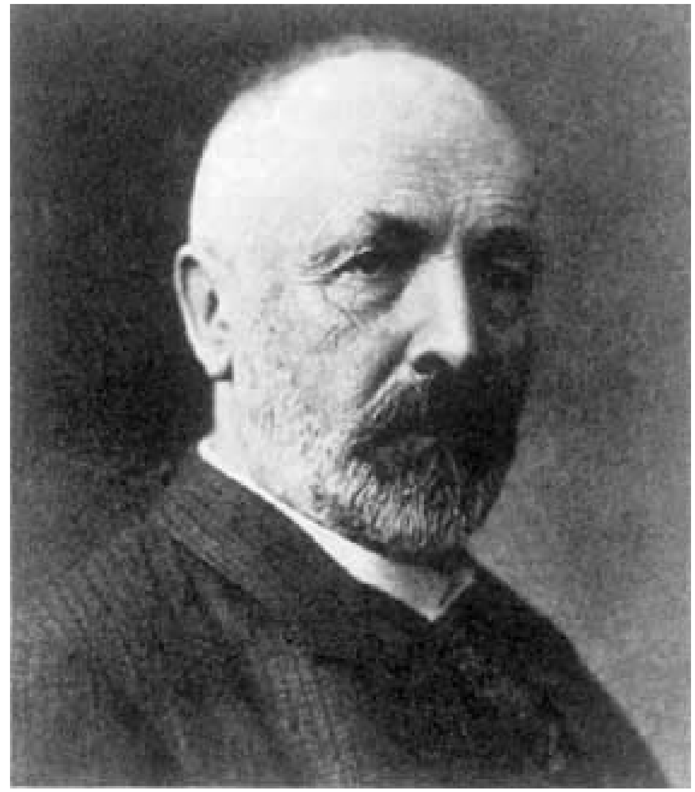
[a]Note that $f(k)$ is a subset of $\mathbb{N}$.

The Proof (concluded)

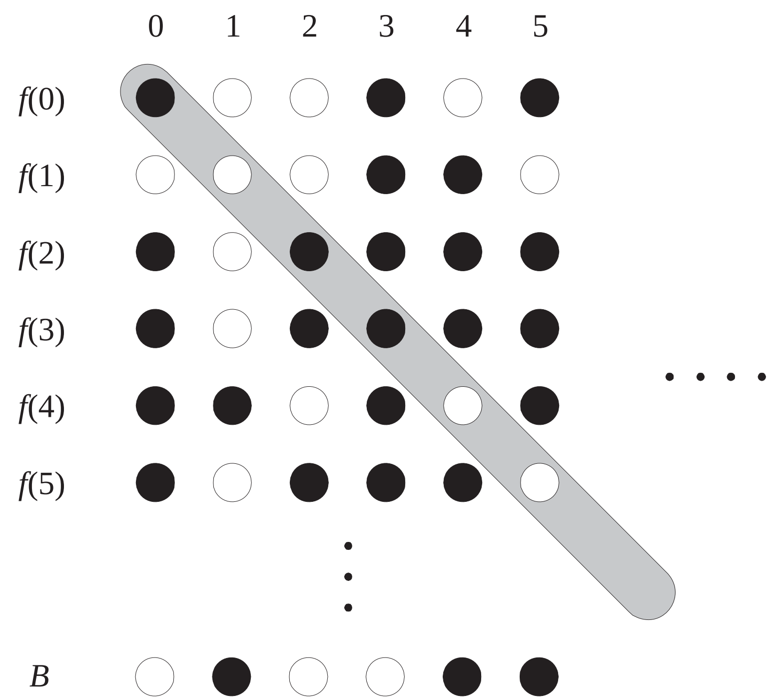
- If $n \in f(n) = B$, then $n \in B$, but then $n \notin B$ by B's definition.
- If $n \notin f(n) = B$, then $n \notin B$, but then $n \in B$ by B's definition.
- Hence $B \neq f(n)$ for any n.
- f is not a bijection, a contradiction.

Georg Cantor (1845–1918)

Kac and Ulam (1968), "[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 10 *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.[a]

[a] Mr. Kai-Yuan Hou (`B99201038`, `R03922014`) on October 13, 2015: Should we limit T to be countable?

The Proof (concluded)

- $|\, T\,| \leq |2^T|$.
  - Consider $f(x) = \{x\} \in 2^T$.
  - f maps a member of $T = \{\, a, b, c, \ldots\}$ to the corresponding member of $\{\, \{\, a\,\}, \{\, b\,\}, \{\, c\,\}, \ldots\} \subseteq 2^T$.
- $|\, T\,| \neq |2^T|$.
  - Use the same argument as Cantor's theorem.

A Second Corollary of Cantor's Theorem

Corollary 11 *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 $f : \mathbb{N} \rightarrow \{0, 1\}$ 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 $|\, 2^{\mathbb{N}} \,|$.
- Cantor's theorem (p. 139) 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 therefore 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 11 (p. 145).
- 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.

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.
  - 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.
  - Equivalently, they are languages that are not recursive.
- We knew undecidable problems exist (p. 146).
- 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.

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:

  if $M_H(M; M) =$ "yes" **then**
      $\nearrow$; {Writing an infinite loop is easy.}
  else
      "yes";
  end if

---

[a] Turing (1936).

H Is Not Recursive (concluded)

- 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]
  - 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.

[a] Eckert and Mauchly (1943); von Neumann (1945); Turing (1946).

Cantor's Paradox[a] (1899)

- Let T be the set of all sets.[b]
- Then $2^T \subseteq T$ because 2^T is a set.
- But we know[c] $|\,2^T\,| > |\,T\,|$ (p. 143)!
- We got a "contradiction."
- Are we willing to give up Cantor's theorem?
- If not, what is a set?[d]

[a] In a letter to Richard Dedekind. First published in Russell (1903).

[b] Recall this ontological argument for the existence of God by St Anselm (1033–1109) 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.

[c] Really?

[d] It partially answers the question on p. 143n.

Self-Loop Paradoxes[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."[b]

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

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

---

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

[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."

Self-Loop Paradoxes (continued)

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

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

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

Self-Loop Paradoxes (concluded)

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

***John* 14:10, New Testament:** "Don't you believe that I am in the Father, and that the Father is in me?"

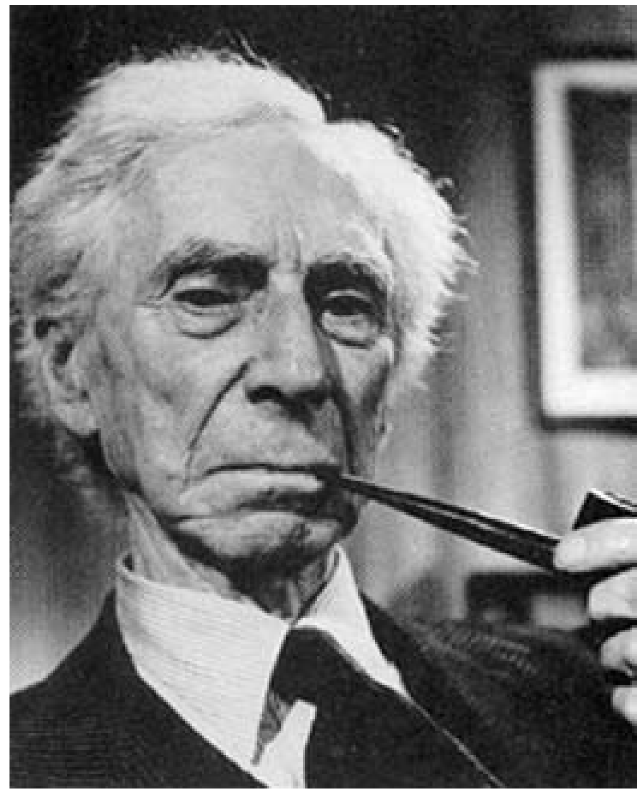
***John* 17:21, New Testament:** "just as you are in me and I am in you."

***Pagan & Christian Creeds* (1920):** "I was moved to Odin, myself to myself."

Soren Kierkegaard in *Fear and Trembling* (1843): "to strive against the whole world is a comfort, to strive with oneself is dreadful."

Bertrand Russell (1872–1970)

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



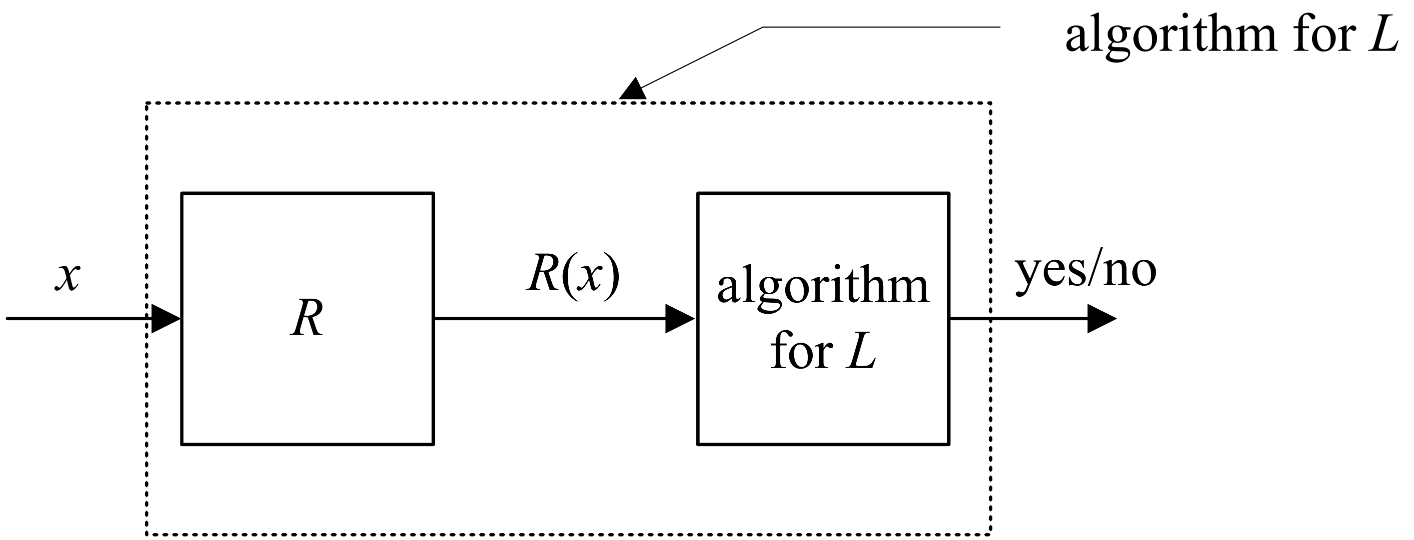
Reductions in Proving Undecidability

- 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**) from L' to L such that[a]

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

- Now we can answer "$x \in L'$?" for *any* x by asking "$R(x) \in L$?" because they have the same answer.
- L' is said to be **reduced** to L.

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


algorithm for L'
x
R
$R(x)$
algorithm for L
yes/no

Reductions in Proving Undecidability (concluded)

- 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 12 *Suppose language L_1 can be reduced to language L_2. If L_1 is undecidable, then L_2 is undecidable.*

Undecidability: Special Cases and Subsets

- Suppose L_1 can be reduced to L_2.
- As the reduction R maps members of L_1 to a *subset* of L_2,[a] we may say L_1 is a "special case" of L_2.[b]
- Now suppose L_1 is undecidable and $L_1 \subseteq L_2$.
- Iis L_2 then undecidable?[c]

---

[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.

[c] Contributed by Ms. Mei-Chih Chang (D03922022) on October 13, 2015.

Undecidability: Special Cases and Subsets (concluded)

- 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,

$$\forall x \; \{x \in L_1 \text{ if and only if } x \notin L_2 - L_1 \text{ and } x \in L_2\}.$$

  - Therefore if L_2 were decidable, then L_1 would be.

More Undecidability

- $H^* = \{M : M \text{ halts on all inputs}\}$.
  - We will reduce H to H^*.
  - Given the question "$M; x \in H$?", construct the following machine (this is the reduction):[a]

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

  - 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. 161), 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

The **complement** of L, denoted by $\bar{L}$, is the language $\Sigma^* - L$.

Lemma 13 *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 $\bar{L}$.[a]

---

[a]Recall p. 105.

Recursive and Recursively Enumerable Languages

Lemma 14 (Kleene's theorem) *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 halt on state "yes" because $x \in L$.
- If $\bar{M}$ accepts, then halt on state "no" because $x \notin L$.
- Note that either M or $\bar{M}$ (but not both) must accept the input and halt.

A Very Useful Corollary and Its Consequences

Corollary 15 *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 14 (p. 168), L is recursive, a contradiction.

Corollary 16 *$\bar{H}$ is not recursively enumerable.*[a]

[a] Recall that $\bar{H} = \{M; x : M(x) = \nearrow\}$.

R, RE, and coRE

RE: The set of all recursively enumerable languages.

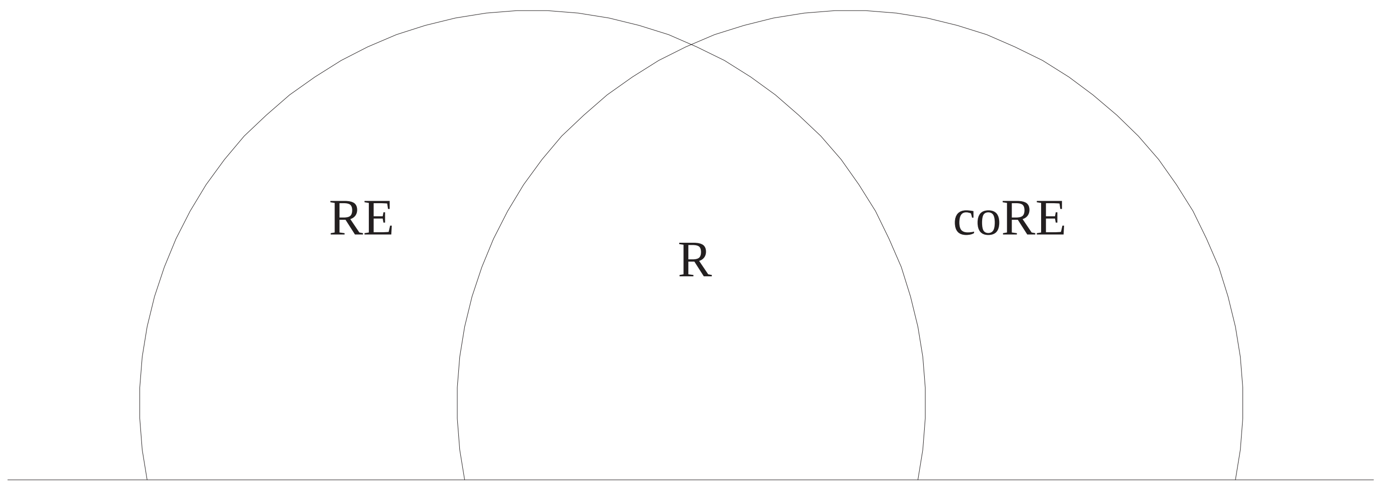
coRE: The set of all languages whose complements are recursively enumerable.

R: The set of all recursive languages.

- Note that coRE is not $\overline{\mathrm{RE}}$.
  - $\mathrm{coRE} = \{ L : \overline{L} \in \mathrm{RE} \} = \{ \overline{L} : L \in \mathrm{RE} \}$.
  - $\overline{\mathrm{RE}} = \{ L : L \notin \mathrm{RE} \}$.

R, RE, and coRE (concluded)

- R = RE $\cap$ coRE (p. 168).
- There exist languages in RE but not in R and not in coRE.
  - Such as H (p. 151, p. 152, and p. 169).
- There are languages in coRE but not in RE.
  - Such as $\bar{H}$ (p. 169).
- There are languages in neither RE nor coRE.


RE
R
coRE

Undecidability in Logic and Mathematics

- First-order logic is undecidable (answer to Hilbert's (1928) *Entscheidungsproblem*).[a]
- Natural numbers with addition and multiplication is undecidable.[b]
- Rational numbers with addition and multiplication is undecidable.[c]

---

[a] Church (1936).
[b] Rosser (1937).
[c] Robinson (1948).

Undecidability in Logic and Mathematics (concluded)

- Natural numbers with addition and equality is decidable and complete.[a]
- Elementary theory of groups is undecidable.[b]

---

[a]Presburger's Master's thesis (1928), his only work in logic. The direction was suggested by Tarski. Mojžesz Presburger (1904–1943) died in a concentration camp during World War II.

[b]Tarski (1949).

Julia Hall Bowman Robinson (1919–1985)

Alfred Tarski (1901–1983)

Boolean Logic

It seemed unworthy of a grown man
to spend his time on such trivialities,
but what was I to do? $[\cdots]$
The whole of the rest of my life might be
consumed in looking at
that blank sheet of paper.
— Bertrand Russell (1872–1970),
Autobiography, Vol. I (1967)

Boolean Logic[a]

Boolean variables: $x_1, x_2, \ldots$.

Literals: x_i, $\neg x_i$.

Boolean connectives: $\vee, \wedge, \neg$.

Boolean expressions: Boolean variables, $\neg\phi$ (**negation**), $\phi_1 \vee \phi_2$ (**disjunction**), $\phi_1 \wedge \phi_2$ (**conjunction**).

- $\bigvee_{i=1}^{n} \phi_i$ stands for $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$.
- $\bigwedge_{i=1}^{n} \phi_i$ stands for $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$.

Implications: $\phi_1 \Rightarrow \phi_2$ is a shorthand for $\neg\phi_1 \vee \phi_2$.

Biconditionals: $\phi_1 \Leftrightarrow \phi_2$ is a shorthand for $(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$.

---

[a] George Boole (1815–1864) in 1847.

Truth Assignments

- A **truth assignment** T is a mapping from boolean variables to **truth values** `true` and `false`.
- A truth assignment is **appropriate** to boolean expression ϕ if it defines the truth value for every variable in ϕ.
  - $\{x_1 = \texttt{true}, x_2 = \texttt{false}\}$ is appropriate to $x_1 \vee x_2$.
  - $\{x_2 = \texttt{true}, x_3 = \texttt{false}\}$ is not appropriate to $x_1 \vee x_2$.

Satisfaction

- $T \models \phi$ means boolean expression ϕ is true under T; in other words, T **satisfies** ϕ.
- ϕ_1 and ϕ_2 are **equivalent**, written

$$\phi_1 \equiv \phi_2,$$

  if for any truth assignment T appropriate to both of them, $T \models \phi_1$ if and only if $T \models \phi_2$.

Truth Tables

- Suppose ϕ has n boolean variables.
- A **truth table** contains 2^n rows.
- Each row corresponds to one truth assignment of the n variables and records the truth value of ϕ under that truth assignment.
- A truth table can be used to prove if two boolean expressions are equivalent.
  - Just check if they give identical truth values under all appropriate truth assignments.

A Truth Table

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

A Second Truth Table

p	q	$p \lor q$
0	0	0
0	1	1
1	0	1
1	1	1

A Third Truth Table

p	$\neg p$
0	1
1	0

Proof of Equivalency: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1