## $H$ 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
    ↗; {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 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.

## Cantor's Paradox (1899[a])

- 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. 128)!
- We got a "contradiction."
- Are we willing to give up Cantor's theorem?
- If not, what is a set?

---

[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?

## Self-Loop Paradoxes[a]

**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."[b]

**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.

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

[b]Gottlob Frege (1848–1925) to Bertrand Russell in 1902, "Your discovery of the contradiction [$\cdots$] has shaken the basis on which I intended to build arithmetic."

## Self-Loop Paradoxes (continued)

**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).

## Self-Loop Paradoxes (concluded)

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

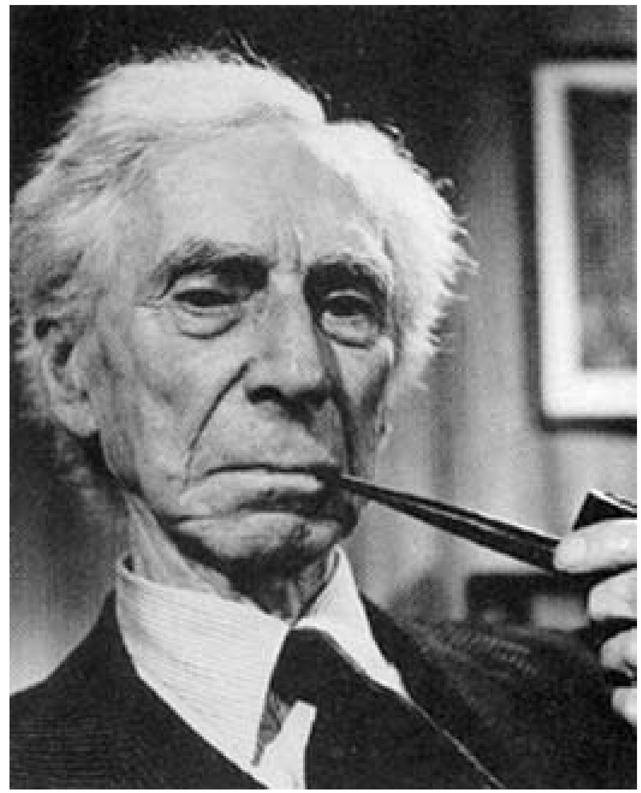
***John* 17:21, New Testament:** "just as you are in me and I am in you."

***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.

## 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 9** *Suppose language $L_1$ can be reduced to language $L_2$. If $L_1$ is not recursive, then $L_2$ is not recursive.*

# 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, $H$ 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 10** *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 11 (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, halt on state "yes" because $x \in L$.
- If $\bar{M}$ accepts, halt on state "no" because $x \notin L$.
- Note that either $M$ or $\bar{M}$ must accept the input.

## A Very Useful Corollary and Its Consequences

**Corollary 12** *$L$ 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 11 (p. 148), $L$ is recursive, a contradiction.

**Corollary 13** *$\bar{H}$ 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.

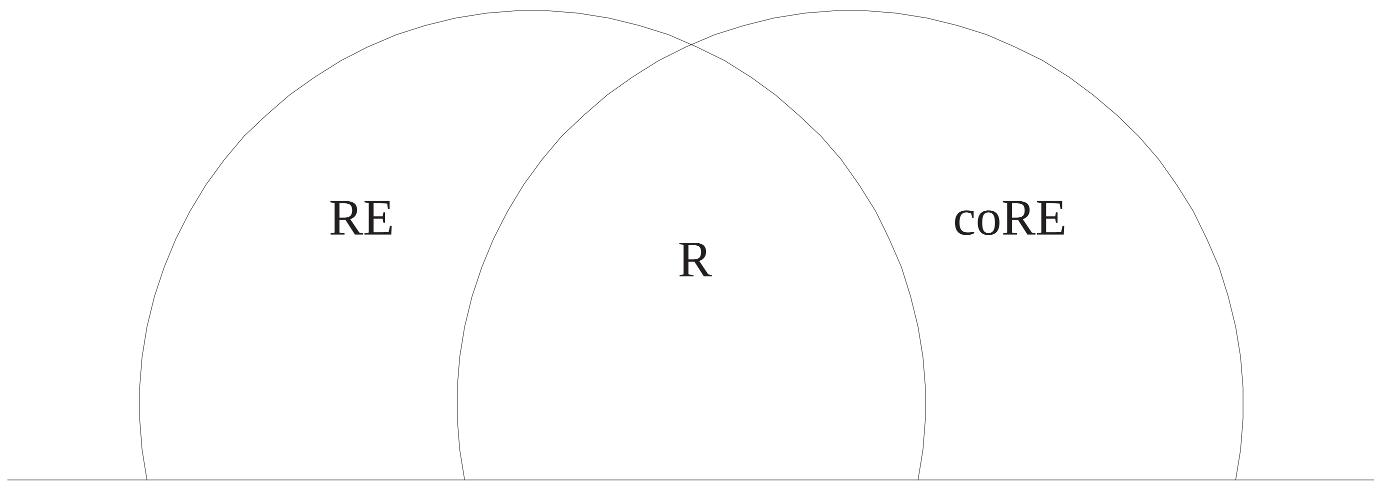
**coRE:** The set of all languages whose complements are recursively enumerable.

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

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

## R, RE, and coRE (concluded)

- $\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$ (p. 148).
- There exist languages in RE but not in R and not in coRE.
  - Such as $H$ (p. 134, p. 135, and p. 149).
- There are languages in coRE but not in RE.
  - Such as $\bar{H}$ (p. 149).
- 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 "*Entscheidungsproblem*" (1928)).[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:** $\lor, \land, \neg$.

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

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

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

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

---

[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 $\phi$ if it defines the truth value for every variable in $\phi$.
  - $\{x_1 = \texttt{true}, x_2 = \texttt{false}\}$ is appropriate to $x_1 \lor x_2$.
  - $\{x_2 = \texttt{true}, x_3 = \texttt{false}\}$ is not appropriate to $x_1 \lor x_2$.

# Satisfaction

- $T \models \phi$ means boolean expression $\phi$ is true under $T$; in other words, $T$ **satisfies** $\phi$.
- $\phi_1$ and $\phi_2$ 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 $\phi$ 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 $\phi$ 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 |

## De Morgan's Laws[a]

- **De Morgan's laws** say that

$$\begin{aligned}\neg(\phi_1 \land \phi_2) &\equiv \neg\phi_1 \lor \neg\phi_2,\\ \neg(\phi_1 \lor \phi_2) &\equiv \neg\phi_1 \land \neg\phi_2.\end{aligned}$$

- Here is a proof of the first law:

| $\phi_1$ | $\phi_2$ | $\neg(\phi_1 \land \phi_2)$ | $\neg\phi_1 \lor \neg\phi_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

[a] Augustus DeMorgan (1806–1871) or William of Ockham (1288–1348).

## Conjunctive Normal Forms

- A boolean expression $\phi$ is in **conjunctive normal form** (**CNF**) if

$$\phi = \bigwedge_{i=1}^{n} C_i,$$

where each **clause** $C_i$ is the disjunction of zero or more literals.[a]

  – For example,

$$(x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (x_2 \lor x_3).$$

- Convention: An empty CNF is satisfiable, but a CNF containing an empty clause is not.

---

[a] Improved by Mr. Aufbu Huang (R95922070) on October 5, 2006.

## Disjunctive Normal Forms

- A boolean expression $\phi$ is in **disjunctive normal form** (**DNF**) if

$$\phi = \bigvee_{i=1}^{n} D_i,$$

where each **implicant** $D_i$ is the conjunction of one or more literals.

  – For example,

$$(x_1 \land x_2) \lor (x_1 \land \neg x_2) \lor (x_2 \land x_3).$$

## Any Expression $\phi$ Can Be Converted into CNFs and DNFs

$\phi = x_j$**:**

- This is trivially true.

$\phi = \neg\phi_1$ **and a CNF is sought:**

- Turn $\phi_1$ into a DNF.
- Apply de Morgan's laws to make a CNF for $\phi$.

$\phi = \neg\phi_1$ **and a DNF is sought:**

- Turn $\phi_1$ into a CNF.
- Apply de Morgan's laws to make a DNF for $\phi$.

## Any Expression $\phi$ Can Be Converted into CNFs and DNFs (continued)

**$\phi = \phi_1 \vee \phi_2$ and a DNF is sought:**

- Make $\phi_1$ and $\phi_2$ DNFs.

**$\phi = \phi_1 \vee \phi_2$ and a CNF is sought:**

- Turn $\phi_1$ and $\phi_2$ into CNFs,[a]

$$\phi_1 = \bigwedge_{i=1}^{n_1} A_i, \quad \phi_2 = \bigwedge_{j=1}^{n_2} B_j.$$

- Set

$$\phi = \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (A_i \vee B_j).$$

[a] Corrected by Mr. Chun-Jie Yang (`R99922150`) on November 9, 2010.

# Any Expression $\phi$ Can Be Converted into CNFs and DNFs (concluded)

**$\phi = \phi_1 \wedge \phi_2$ and a CNF is sought:**

- Make $\phi_1$ and $\phi_2$ CNFs.

**$\phi = \phi_1 \wedge \phi_2$ and a DNF is sought:**

- Turn $\phi_1$ and $\phi_2$ into DNFs,

$$\phi_1 = \bigvee_{i=1}^{n_1} A_i, \quad \phi_2 = \bigvee_{j=1}^{n_2} B_j.$$

- Set

$$\phi = \bigvee_{i=1}^{n_1} \bigvee_{j=1}^{n_2} (A_i \wedge B_j).$$

## An Example: Turn $\neg((a \wedge y) \vee (z \vee w))$ into a DNF

$$
\begin{aligned}
& \neg((a \wedge y) \vee (z \vee w)) \\
\stackrel{\neg(\mathrm{CNF} \vee \mathrm{CNF})}{=} \quad & \neg(((a) \wedge (y)) \vee ((z \vee w))) \\
\stackrel{\neg(\mathrm{CNF})}{=} \quad & \neg((a \vee z \vee w) \wedge (y \vee z \vee w)) \\
\stackrel{\text{de Morgan}}{=} \quad & \neg(a \vee z \vee w) \vee \neg(y \vee z \vee w) \\
\stackrel{\text{de Morgan}}{=} \quad & (\neg a \wedge \neg z \wedge \neg w) \vee (\neg y \wedge \neg z \wedge \neg w).
\end{aligned}
$$

# Satisfiability

- A boolean expression $\phi$ is **satisfiable** if there is a truth assignment $T$ appropriate to it such that $T \models \phi$.
- $\phi$ is **valid** or a **tautology**,[a] written $\models \phi$, if $T \models \phi$ for all $T$ appropriate to $\phi$.
- $\phi$ is **unsatisfiable** if and only if $\phi$ is false under all appropriate truth assignments if and only if $\neg\phi$ is valid.

---

[a] Wittgenstein (1889–1951) in 1922. Wittgenstein is one of the most important philosophers of all time. "God has arrived," the great economist Keynes (1883–1946) said of him on January 18, 1928. "I met him on the 5:15 train." Russell (1919), "The importance of 'tautology' for a definition of mathematics was pointed out to me by my former pupil Ludwig Wittgenstein, who was working on the problem. I do not know whether he has solved it, or even whether he is alive or dead."

# Ludwig Wittgenstein (1889–1951)

Wittgenstein (1922), "Whereof one cannot speak, thereof one must be silent."

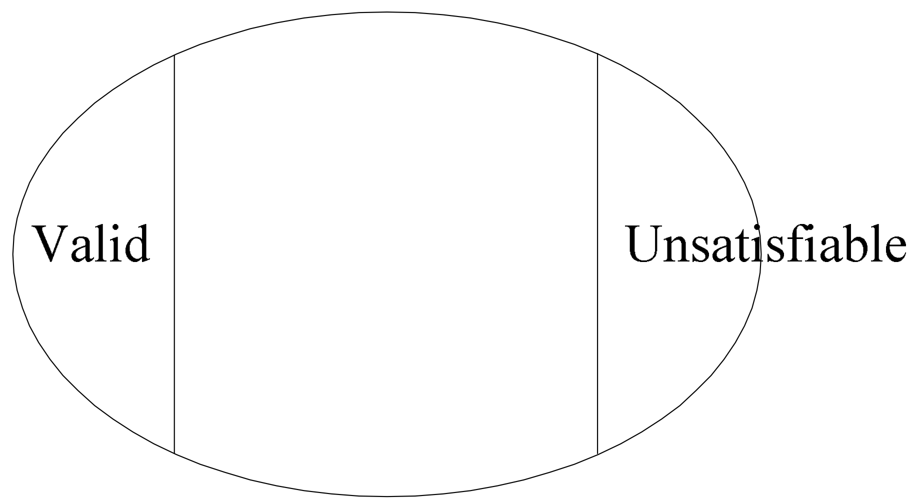
# SATISFIABILITY (SAT)

- The **length** of a boolean expression is the length of the string encoding it.
- SATISFIABILITY (SAT): Given a CNF $\phi$, is it satisfiable?
- Solvable in exponential time on a TM by the truth table method.
- Solvable in polynomial time on an NTM, hence in NP (p. 100).
- A most important problem in settling the "P $\stackrel{?}{=}$ NP" problem (p. 282).

## UNSATISFIABILITY (UNSAT or SAT COMPLEMENT) and VALIDITY

- UNSAT (SAT COMPLEMENT): Given a boolean expression $\phi$, is it unsatisfiable?
- VALIDITY: Given a boolean expression $\phi$, is it valid?
  - $\phi$ is valid if and only if $\neg\phi$ is unsatisfiable.
  - $\phi$ and $\neg\phi$ are basically of the same length.
  - So UNSAT and VALIDITY have the same complexity.
- Both are solvable in exponential time on a TM by the truth table method.
- Can we do better?

# Relations among SAT, UNSAT, and VALIDITY




- The negation of an unsatisfiable expression is a valid expression.
- None of the three problems—satisfiability, unsatisfiability, validity—are known to be in P.

# Boolean Functions

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

$$f : \{\texttt{true}, \texttt{false}\}^n \to \{\texttt{true}, \texttt{false}\}.$$

- It can be represented by a truth table.
- There are $2^{2^n}$ such boolean functions.
  - We can assign `true` or `false` to $f$ under each of the $2^n$ truth assignments.

# Boolean Functions (continued)

| Assignment | Truth value |
|---|---|
| 1 | `true` or `false` |
| 2 | `true` or `false` |
| $\vdots$ | $\vdots$ |
| $2^n$ | `true` or `false` |

## Boolean Functions (continued)

- A boolean expression expresses a boolean function.
  - Think of its truth value under all truth assignments.
- A boolean function expresses a boolean expression.
  - $\bigvee_{T \,\models\, \phi,\ \text{literal } y_i \text{ is true in ``row'' } T} (y_1 \wedge \cdots \wedge y_n)$.
    - $y_1 \wedge \cdots \wedge y_n$ is called the **minterm** over $\{x_1, \ldots, x_n\}$ for $T$.[a]
  - The size[b] is $\leq n2^n \leq 2^{2n}$.

---

[a] Similar to **programmable logic array**.

[b] We count only the literals here.

## Boolean Functions (continued)

| $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The corresponding boolean expression:

$$(\neg x_1 \land \neg x_2) \lor (\neg x_1 \land x_2) \lor (x_1 \land x_2).$$

## Boolean Functions (concluded)

**Corollary 14** *Every $n$-ary boolean function can be expressed by a boolean expression of size $O(n2^n)$.*

- In general, the exponential length in $n$ cannot be avoided (p. 186).
- The size of the truth table is also $O(n2^n)$.

# Boolean Circuits

- A **boolean circuit** is a graph $C$ whose nodes are the **gates**.
- There are no cycles in $C$.
- All nodes have indegree (number of incoming edges) equal to 0, 1, or 2.
- Each gate has a **sort** from

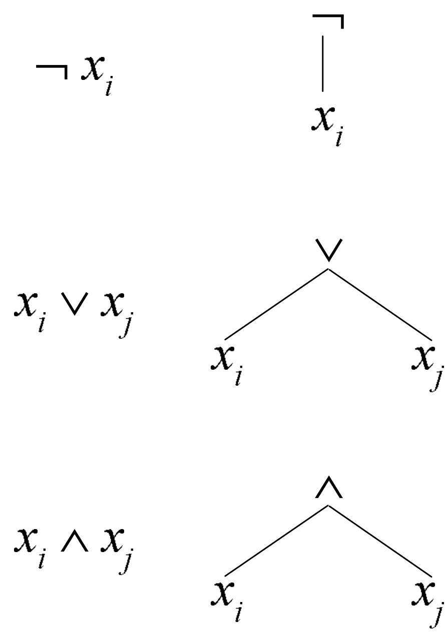
$$\{\texttt{true}, \texttt{false}, \lor, \land, \neg, x_1, x_2, \ldots\}.$$

## Boolean Circuits (concluded)

- Gates with a sort from $\{\texttt{true}, \texttt{false}, x_1, x_2, \ldots\}$ are the **inputs** of $C$ and have an indegree of zero.
- The **output gate**(s) has no outgoing edges.
- A boolean circuit computes a boolean function.
- The same boolean function can be computed by infinitely many boolean circuits.

# Boolean Circuits and Expressions

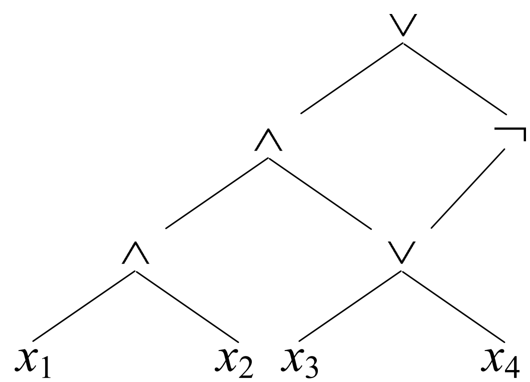
- They are equivalent representations.
- One can construct one from the other:



# An Example

$$((x_1 \wedge x_2) \wedge (x_3 \vee x_4)) \vee (\neg (x_3 \vee x_4))$$




- Circuits are more economical because of the possibility of sharing.

## CIRCUIT SAT and CIRCUIT VALUE

CIRCUIT SAT: Given a circuit, is there a truth assignment such that the circuit outputs true?

- CIRCUIT SAT $\in$ NP: Guess a truth assignment and then evaluate the circuit.

CIRCUIT VALUE: The same as CIRCUIT SAT except that the circuit has no variable gates.

- CIRCUIT VALUE $\in$ P: Evaluate the circuit from the input gates gradually towards the output gate.