More Undecidability

- $\{M : M \text{ halts on all inputs}\}$.
  - Given $M; x$, we construct the following machine:
    - $M_x(y)$: if $y = x$ then $M(x)$ else halt.
  - M_x halts on all inputs if and only if M halts on x.
  - So if the said language were recursive, H would be recursive, a contradiction.
  - This technique is called **reduction**.

Reductions in Proving Undecidability

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

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

- This suffices to prove that L is undecidable.

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

Complements of Recursive Languages

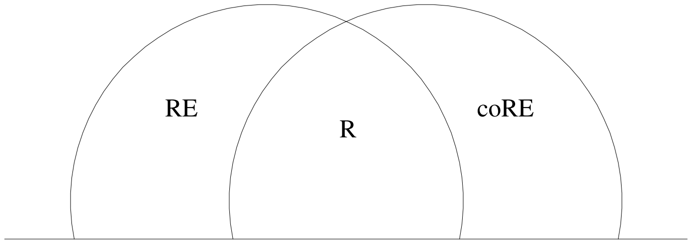
Lemma 6 *If L is recursive, then so is $\bar{L}$.*

- 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 7 *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."



Boolean Logic

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]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 $x_1 \vee x_2$.

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 $T \models \phi_2$.
  - Equivalently, $T \models (\phi_1 \Leftrightarrow \phi_2)$.

Truth Tables

- Suppose ϕ has n boolean variables.
- A **truth table** contains 2^n rows, one for each possible truth assignment of the n variables together with the truth value of ϕ under that truth assignment.
- A truth table can be used to prove if two boolean expressions are equivalent.
  - Check if they give identical truth values under all 2^n 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[a] Laws

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

$$\neg(\phi_1 \wedge \phi_2) = \neg\phi_1 \vee \neg\phi_2,$$
$$\neg(\phi_1 \vee \phi_2) = \neg\phi_1 \wedge \neg\phi_2.$$

- Here is a proof for the first law:

ϕ_1	ϕ_2	$\neg(\phi_1 \wedge \phi_2)$	$\neg\phi_1 \vee \neg\phi_2$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

[a] Augustus DeMorgan (1806–1871).

Conjunctive Normal Forms

- A boolean expression ϕ is in **conjunctive normal form** (**CNF**) if
$$\phi = \bigwedge_{i=1}^{n} C_i,$$
where each **clause** C_i is the disjunction of one or more literals.
- For example, $(x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (x_2 \lor x_3)$ is in CNF.
- Convention: An empty CNF is satisfiable, but a CNF containing an empty clause is not.

Disjunctive Normal Forms

- A boolean expression ϕ 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,
$$(x_1 \land x_2) \lor (x_1 \land \neg x_2) \lor (x_2 \land x_3)$$
is in DNF.

Any Expression ϕ Can Be Converted into CNFs and DNFs

$\phi = x_j$: This is trivially true.

$\phi = \neg\phi_1$ and a CNF is sought: Turn ϕ_1 into a DNF and apply de Morgan's laws to make a CNF for ϕ.

$\phi = \neg\phi_1$ and a DNF is sought: Turn ϕ_1 into a CNF and apply de Morgan's laws to make a DNF for ϕ.

$\phi = \phi_1 \lor \phi_2$ and a DNF is sought: Make ϕ_1 and ϕ_2 DNFs.

$\phi = \phi_1 \lor \phi_2$ and a CNF is sought: Let $\phi_1 = \bigwedge_{i=1}^{n_1} A_i$ and $\phi_2 = \bigwedge_{i=1}^{n_2} B_i$ be CNFs. Set $\phi = \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (A_i \lor B_j)$.

$\phi = \phi_1 \land \phi_2$: Similar to above.

Satisfiability

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

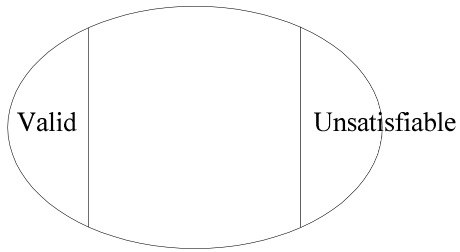
SATISFIABILITY (SAT)

- The **length** of a boolean expression is the length of the string encoding it.
- SATISFIABILITY (SAT): Given a CNF ϕ, is it satisfiable?
- Solvable in time $O(n^2 2^n)$ on a TM by the truth table method.
- Solvable in polynomial time on an NTM, hence in NP (p. 49).
- A most important problem in answering the P = NP problem (p. 149).

UNSATISFIABILITY (UNSAT or SAT COMPLEMENT) and VALIDITY

- UNSAT (SAT COMPLEMENT): Given a boolean expression ϕ, is it unsatisfiable?
- VALIDITY: Given a boolean expression ϕ, is it valid?
  - ϕ is valid if and only if $\neg\phi$ is unsatisfiable.
  - So UNSAT and VALIDITY have the same complexity.
- Both are solvable in time $O(n^2 2^n)$ on a TM by the truth table method.

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.
  - Each of the 2^n truth assignments can make f 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.
  - $\bigvee_{T \models \phi, \text{ literal } y_i \text{ is true under } T} (y_1 \wedge \cdots \wedge y_n)$.
    * $y_1 \wedge \cdots \wedge y_n$ is the **minterm** over $\{x_1, \ldots, x_n\}$ for T.
  - The length[a] is $\leq n2^n \leq 2^{2n}$.
  - In general, the exponential length in n cannot be avoided (p. 94)!

[a]We count the logical connectives here.

Boolean Functions (concluded)

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:

$$(\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \vee (x_1 \wedge x_2).$$

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

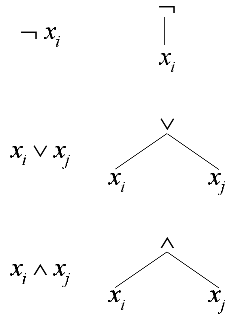
$$\{\texttt{true}, \texttt{false}, \vee, \wedge, \neg, x_1, x_2, \ldots\}.$$

Boolean Circuits (concluded)

- Gates of 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.

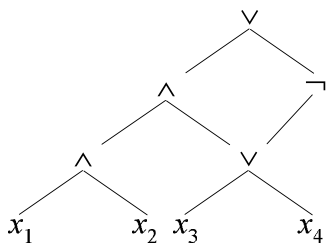
Boolean Circuits and Expressions

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

- CIRCUIT SAT $\in$ NP: Guess a truth assignment and then evaluate the circuit.
- CIRCUIT VALUE $\in$ P: Evaluate the circuit from the input gates gradually towards the output gate.

Some Boolean Functions Need Exponential Circuits[a]

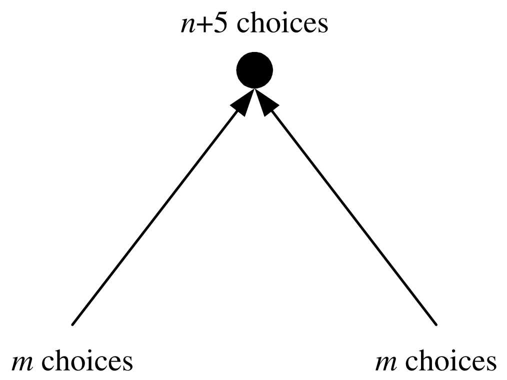
Theorem 8 (Shannon (1949)) *For any $n \geq 2$, there is an n-ary boolean function f such that no boolean circuits with $2^n/(2n)$ or fewer gates can compute it.*

- There are 2^{2^n} different n-ary boolean functions.
- So it suffices to prove that the number of boolean circuits with $2^n/(2n)$ or fewer gates is less than 2^{2^n}.

[a] Can be strengthened to "almost all boolean functions ..."

The Proof (concluded)

- There are at most $((n+5) \times m^2)^m$ boolean circuits with m or fewer gates (see next page).
- But $((n+5) \times m^2)^m < 2^{2^n}$ when $m = 2^n/(2n)$.
  - $m \log_2((n+5) \times m^2) = 2^n(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n}) < 2^n$ for $n \geq 2$.



Relations between Complexity Classes

Proper (Complexity) Functions

- We say that $f : \mathbb{N} \rightarrow \mathbb{N}$ is a **proper (complexity) function** if the following hold:
  - f is nondecreasing.
  - There is a k-string TM M_f such that $M_f(x) = \sqcap^{f(\,|\,x\,|\,)}$ for any x.[a]
  - M_f halts after $O(|\,x\,| + f(|\,x\,|))$ steps.
  - M_f uses $O(f(|\,x\,|))$ space besides its input x.
- M_f's behavior depends only on $|\,x\,|$ not x's contents.
- M_f's running time is basically bounded by $f(n)$.

[a]This point will become clear in Proposition 9 on p. 101.

Examples of Proper Functions

- Most "reasonable" functions are proper: c, $\lceil \log n \rceil$, polynomials of n, 2^n, $\sqrt{n}$, $n!$, etc.
- If f and g are proper, then so are $f + g$, fg, and 2^g.
- Only proper functions f will be used in $\text{TIME}(f(n))$ and $\text{NTIME}(f(n))$.

Precise Turing Machines

- A TM M is **precise** if there are functions f and g such that for every $n \in \mathbb{N}$, for every x of length n, and for every computation path of M,
  - M halts after precise $f(n)$ steps, and
  - All of its strings are of length precisely $g(n)$ at halting.
- M can be deterministic or nondeterministic.

Precise TMs Are General

Proposition 9 *Suppose a TM*[a] *M decides L within time (space) $f(n)$, where f is proper. Then there is a precise TM M' which decides L in time $O(n + f(n))$ (space $O(f(n))$, respectively).*

- M' on input x first simulates the TM M_f associated with the proper function f on x.
- M_f's output of length $f(|x|)$ will serve as a "yardstick" or an "alarm clock."

[a] It can be deterministic or nondeterministic.

The Proof (continued)

- If f is a time bound:
  - The simulation of each step of M on x is matched by advancing the cursor on the "clock" string.
  - M' stops at the moment the "clock" string is exhausted—even if $M(x)$ stops before that time.
  - The time bound is therefore $O(|x| + f(|x|))$.

The Proof (concluded)

- If f is a space bound:
  - M' simulates *on* M_f's output string.
  - The total space, not counting the input string, is $O(f(n))$.

Important Time Complexity Classes

- We write expressions like n^k to denote the union of all complexity classes, one for each value of k.
- For example,

$$\mathrm{NTIME}(n^k) = \bigcup_{j>0} \mathrm{NTIME}(n^j).$$

Important Time Complexity Classes (concluded)

$$\begin{aligned} \mathrm{P} &= \mathrm{TIME}(n^k), \\ \mathrm{NP} &= \mathrm{NTIME}(n^k), \\ \mathrm{E} &= \mathrm{TIME}(2^{kn}), \\ \mathrm{EXP} &= \mathrm{TIME}(2^{n^k}), \end{aligned}$$