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. 119).

- A most important problem in settling the “$P \overset{?}{=} NP$” problem (p. 312).
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

$$f : \{\text{true, false}\}^n \rightarrow \{\text{true, false}\}.$$ 

• It can be represented by a truth table.

• There are $2^{2^n}$ such boolean functions.
  - We can assign true or false to $f$ for each of the $2^n$ truth assignments.

• How about $\{\text{true, false}\}^n \rightarrow \{\text{true, false}\}^m$?
Boolean Functions (continued)

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Truth value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true or false</td>
</tr>
<tr>
<td>2</td>
<td>true or false</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$2^n$</td>
<td>true or false</td>
</tr>
</tbody>
</table>
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.
  - $\bigvee_T \models \phi$, literal $y_i$ is true in “row” $T(y_1 \land \cdots \land y_n)$.
    * $y_1 \land \cdots \land y_n$ 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)

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$f(x_1, x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The corresponding boolean expression:

$$(\neg x_1 \land \neg x_2) \lor (\neg x_1 \land x_2) \lor (x_1 \land x_2).$$
Corollary 15  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. 211).
- 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

\[
\{\text{true, false, } \lor, \land, \neg, x_1, x_2, \ldots\}.
\]

- There are $n + 5$ sorts.
Boolean Circuits (concluded)

• Gates with a sort from \{true, false, \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 equivalent boolean circuits.
Boolean Circuits and Expressions

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

\[ \neg x_i \quad \neg \]

\[ x_i \quad x_i \]

\[ x_i \lor x_j \quad \lor \]

\[ x_i \quad x_j \quad x_i \quad x_j \]

\[ x_i \land x_j \quad \land \]

\[ x_i \quad x_j \quad x_i \quad x_j \]
An Example

\(( (x_1 \land x_2) \land (x_3 \lor x_4)) \lor (\neg (x_3 \lor x_4))\)

- 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 ∈ 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 ∈ P: Evaluate the circuit from the input gates gradually towards the output gate.
Some Boolean Functions Need Exponential Circuits\(^a\)

**Theorem 16 (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 (p. 201).
- 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 \(m = 2^n/(2n)\):

\[
m \log_2 ((n + 5) \times m^2) = 2^n \left( 1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) < 2^n
\]

for \(n \geq 2\).
$n+5$ choices

$m$ choices

$m$ choices
Claude Elwood Shannon (1916–2001)

Howard Gardner, “[Shannon’s master’s thesis is] possibly the most important, and also the most famous, master’s thesis of the century.”
Comments

• The lower bound $2^n/(2n)$ is rather tight because an upper bound is $n2^n$ (p. 203).

• The proof counted the number of circuits.
  – Some circuits may not be valid at all.
  – Different circuits may also compute the same function.

• Both are fine because we only need an upper bound on the number of circuits.

• We do not need to consider the outdoing edges because they have been counted as incoming edges.
Relations between Complexity Classes
It is, I own, not uncommon to be wrong in theory and right in practice.

— Edmund Burke (1729–1797),

*A Philosophical Enquiry into the Origin of Our Ideas of the Sublime and Beautiful* (1757)
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 bounded by $f(n)$.

\(^a\)The textbook calls “\(\sqcap\)” the quasi-blank symbol. The use of $M_f(x)$ will become clear in Proposition 17 (p. 221).
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 $2^g$.\(^a\)

• Nonproper functions when serving as the time bounds for complexity classes spoil “the theory building.”
  
  – For example, $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$ for some recursive function $f$ (the gap theorem).\(^b\)

• Only proper functions $f$ will be used in $\text{TIME}(f(n))$, $\text{SPACE}(f(n))$, $\text{NTIME}(f(n))$, and $\text{NSPACE}(f(n))$.

\(^a\)For $f(g)$, we need to add $f(n) \geq n$.

\(^b\)Trakhtenbrot (1964); Borodin (1972).
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 precisely $f(n)$ steps, and
  - All of its strings are of length precisely $g(n)$ at halting.

* Recall that if $M$ is a TM with input and output, we exclude the first and last strings.

• $M$ can be deterministic or nondeterministic.
Precise TMs Are General

**Proposition 17** 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)

• Then $M'$ simulates $M(x)$.

• $M'(x)$ halts when and only when the alarm clock runs out—even if $M$ halts earlier.

• 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.
  – Because $M'$ stops at the moment the “clock” string is exhausted—even if $M(x)$ stops earlier, it is precise.
  – The time bound is therefore $O(|x| + f(|x|))$. 
The Proof (concluded)

• If $f$ is a space bound (sketchy):
  – $M'$ simulates $M$ on the quasi-blanks of $M_f$’s output string.
  – The total space, not counting the input string, is $O(f(n))$.
  – But we still need a way to make sure there is no infinite loop.\(^a\)

\(^a\)See the proof of Theorem 24 on p. 239.
Important Complexity Classes

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

- For example,

\[ \text{NTIME}(n^k) = \bigcup_{j>0} \text{NTIME}(n^j). \]
Important Complexity Classes (concluded)

\[
\begin{align*}
P &= \text{TIME}(n^k), \\
\text{NP} &= \text{NTIME}(n^k), \\
\text{PSPACE} &= \text{SPACE}(n^k), \\
\text{NPSPACE} &= \text{NSPACE}(n^k), \\
\text{E} &= \text{TIME}(2^{kn}), \\
\text{EXP} &= \text{TIME}(2^{n^k}), \\
\text{L} &= \text{SPACE}(\log n), \\
\text{NL} &= \text{NSPACE}(\log n).
\end{align*}
\]
Complements of Nondeterministic Classes

- Recall that the complement of $L$, denoted by $\bar{L}$, is the language $\Sigma^* - L$.
  - SAT complement is the set of unsatisfiable boolean expressions.

- We knew that $R$, $RE$, and $coRE$ are distinct (p. 172).
  - Again, $coRE$ contains the complements of languages in $RE$, not the languages not in $RE$.

- How about $coC$ when $C$ is a complexity class?
The Co-Classes

- For any complexity class $C$, $\text{co}C$ denotes the class $\{L: \overline{L} \in C\}$.

- Clearly, if $C$ is a deterministic time or space complexity class, then $C = \text{co}C$.
  - They are said to be closed under complement.
  - A deterministic TM deciding $L$ can be converted to one that decides $\overline{L}$ within the same time or space bound by reversing the “yes” and “no” states (p. 169).

- Whether nondeterministic classes for time are closed under complement is not known (p. 111).
Comments

• As

$$\text{co} \mathcal{C} = \{ L : \bar{L} \in \mathcal{C} \},$$

$L \in \mathcal{C}$ if and only if $\bar{L} \in \text{co} \mathcal{C}$.

• But it is not true that $L \in \mathcal{C}$ if and only if $L \notin \text{co} \mathcal{C}$.
  
  – $\text{co} \mathcal{C}$ is not defined as $\bar{\mathcal{C}}$.

• For example, suppose $\mathcal{C} = \{\{2, 4, 6, 8, 10, \ldots\}\}$.

• Then $\text{co} \mathcal{C} = \{\{1, 3, 5, 7, 9, \ldots\}\}$.

• But $\bar{\mathcal{C}} = 2^{\{1,2,3,\ldots\}} - \{\{2, 4, 6, 8, 10, \ldots\}\}$.
The Quantified Halting Problem

• Let $f(n) \geq n$ be proper.

• Define

$$H_f = \{ M; x : M \text{ accepts input } x \text{ after at most } f(|x|) \text{ steps} \},$$

where $M$ is deterministic.

• Assume the input is binary.
\[ H_f \in \text{TIME}(f(n)^3) \]

- For each input \( M; x \), we simulate \( M \) on \( x \) with an alarm clock of length \( f(|x|) \).
  - Use the single-string simulator (p. 87), the universal TM (p. 153), and the linear speedup theorem (p. 96).
  - Our simulator accepts \( M; x \) if and only if \( M \) accepts \( x \) before the alarm clock runs out.

- From p. 94, the total running time is \( O(\ell_M k_M^2 f(n)^2) \), where \( \ell_M \) is the length to encode each symbol or state of \( M \) and \( k_M \) is \( M \)'s number of strings.

- As \( \ell_M k_M^2 = O(n) \), the running time is \( O(f(n)^3) \), where the constant is independent of \( M \).
$H_f \notin \text{TIME}(f(\lfloor n/2 \rfloor))$

- Suppose TM $M_{H_f}$ decides $H_f$ in time $f(\lfloor n/2 \rfloor)$.
- Consider machine:

  $$D_f(M) \begin{cases} 
  \text{if } M_{H_f}(M; M) = \text{“yes”} \\
  \text{then “no”;} \\
  \text{else “yes”;}
  \end{cases}$$

- $D_f$ on input $M$ runs in the same time as $M_{H_f}$ on input $M; M$, i.e., in time $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$, where $n = |M|$.

---

\[a\] A student pointed out on October 6, 2004, that this estimation omits the time to write down $M; M$. Regression Error Estimation
The Proof (concluded)

• First,

\[ D_f(D_f) = "yes" \]
\[ \Rightarrow D_f; D_f \not\in H_f \]
\[ \Rightarrow D_f \text{ does not accept } D_f \text{ within time } f(|D_f|) \]
\[ \Rightarrow D_f(D_f) \neq "yes" \]
\[ \Rightarrow D_f(D_f) = "no" \]

a contradiction

• Similarly, \( D_f(D_f) = "no" \Rightarrow D_f(D_f) = "yes." \)
The Time Hierarchy Theorem

**Theorem 18** If \( f(n) \geq n \) is proper, then

\[
\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n + 1)^3).
\]

- The quantified halting problem makes it so.

**Corollary 19** \( P \subsetneq E \).

- \( P \subseteq \text{TIME}(2^n) \) because \( \text{poly}(n) \leq 2^n \) for \( n \) large enough.
- But by Theorem 18,

\[
\text{TIME}(2^n) \subsetneq \text{TIME}((2^{2n+1})^3) \subseteq E.
\]
- So \( P \subsetneq E \).
The Space Hierarchy Theorem

Theorem 20 (Hennie and Stearns (1966)) If \( f(n) \) is proper, then

\[
\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n) \log f(n)).
\]

Corollary 21 \( \text{L} \subsetneq \text{PSPACE} \).
Nondeterministic Time Hierarchy Theorems

Theorem 22 (Cook (1973)) \( \text{NTIME}(n^r) \subsetneq \text{NTIME}(n^s) \) whenever \( 1 \leq r < s \).

Theorem 23 (Seiferas, Fischer, and Meyer (1978)) If \( T_1(n), T_2(n) \) are proper, then

\[
\text{NTIME}(T_1(n)) \subsetneq \text{NTIME}(T_2(n))
\]

whenever \( T_1(n + 1) = o(T_2(n)) \).
The Reachability Method

- The computation of a time-bounded TM can be represented by a directed graph.
- The TM’s configurations constitute the nodes.
- Two nodes are connected by a directed edge if one yields the other in one step.
- The start node representing the initial configuration has zero in degree.
The Reachability Method (concluded)

• When the TM is nondeterministic, a node may have an out degree greater than one.
  – The graph is the same as the computation tree earlier except that identical configuration nodes are merged into one node.

• So $M$ accepts the input if and only if there is a path from the start node to a node with a “yes” state.

• It is the reachability problem.
Illustration of the Reachability Method

Initial configuration

yes

yes
Relations between Complexity Classes

Theorem 24 Suppose \( f(n) \) is proper. Then

1. \( \text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n)) \),
   \( \text{TIME}(f(n)) \subseteq \text{NTIME}(f(n)) \).

2. \( \text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n)) \).

3. \( \text{NSPACE}(f(n)) \subseteq \text{TIME}(k \log n + f(n)) \).

\[ \bullet \] Proof of 2:

- Explore the computation tree of the NTM for “yes.”
- Specifically, generate an \( f(n) \)-bit sequence denoting the nondeterministic choices over \( f(n) \) steps.
Proof of Theorem 24(2)

• (continued)
  – Simulate the NTM based on the choices.
  – Recycle the space and repeat the above steps.
  – Halt with “yes” when a “yes” is encountered or “no” if the tree is exhausted.
  – Each path simulation consumes at most $O(f(n))$ space because it takes $O(f(n))$ time.
  – The total space is $O(f(n))$ because space is recycled.
Proof of Theorem 24(3)

- Let $k$-string NTM

$$M = (K, \Sigma, \Delta, s)$$

with input and output decide $L \in \text{NSPACE}(f(n))$.

- Use the reachability method on the configuration graph of $M$ on input $x$ of length $n$.

- A configuration is a $(2k + 1)$-tuple

$$(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).$$
Proof of Theorem 24(3) (continued)

• We only care about

\[(q, i, w_2, u_2, \ldots, w_{k-1}, u_{k-1}),\]

where \(i\) is an integer between 0 and \(n\) for the position of the first cursor.

• The number of configurations is therefore at most

\[|K| \times (n + 1) \times |\Sigma|^{(2k-4)f(n)} = O(c_1^{\log n + f(n)})\]  \hspace{1cm} (2)

for some \(c_1\), which depends on \(M\).

• Add edges to the configuration graph based on \(M\)’s transition function.
Proof of Theorem 24(3) (concluded)

• $x \in L \iff$ there is a path in the configuration graph from the initial configuration to a configuration of the form (“yes”, i, …).\textsuperscript{a}

• This is REACHABILITY on a graph with $O(c_1 \log n + f(n))$ nodes.

• It is in $\text{TIME}(c_1 \log n + f(n))$ for some $c$ because $\text{REACHABILITY} \in \text{TIME}(n^j)$ for some $j$ and

$$\left[c_1 \log n + f(n)\right]^j = (c_1^j) \log n + f(n).$$

\textsuperscript{a}There may be many of them.
Space-Bounded Computation and Proper Functions

- In the definition of space-bounded computations earlier (p. 110), the TMs are not required to halt at all.

- When the space is bounded by a proper function $f$, computations can be assumed to halt:
  - Run the TM associated with $f$ to produce a quasi-blank output of length $f(n)$ first.
  - The space-bounded computation must repeat a configuration if it runs for more than $c \log n + f(n)$ steps for some $c$ (p. 242).
Space-Bounded Computation and Proper Functions (concluded)

- (continued)
  - So an infinite loop occurs during simulation for a computation path longer than $c \log n + f(n)$ steps.
  - Hence we only simulate up to $c \log n + f(n)$ time steps per computation path.
A Grand Chain of Inclusions\textsuperscript{a}

- It is an easy application of Theorem 24 (p. 239) that

\[ L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP. \]

- By Corollary 21 (p. 234), we know \( L \subsetneq PSPACE \).

- So the chain must break somewhere between \( L \) and \( EXP \).

- It is suspected that all four inclusions are proper.

- But there are no proofs yet.

\textsuperscript{a}With input from Mr. Chin-Luei Chang (R93922004, D95922007) on October 22, 2004.
Nondeterministic Space and Deterministic Space

• By Theorem 5 (p. 116),

\[ \text{NTIME}(f(n)) \subseteq \text{TIME}(c^f(n)), \]

an exponential gap.

• There is no proof yet that the exponential gap is inherent.

• How about NSPACE vs. SPACE?

• Surprisingly, the relation is only quadratic—a polynomial—by Savitch’s theorem.