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 and 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 and apply de Morgan’s laws to make a DNF for $\phi$.

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

Satisifiability

- 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, 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 and if and only if $\neg \phi$ is valid.

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 $\phi$, 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. 84).
  - A most important problem in answering the P = NP problem (p. 237).

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.
    - 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 (continued)

- A boolean expression expresses a boolean function.
  - Think of its truth value under all truth assignments.
- A boolean function expresses a boolean expression.
  - \( \lor_T \phi \), literal \( y_i \) is true under \( T(y_1 \land \cdots \land y_n) \).
  - \( y_1 \land \cdots \land y_n \) 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. 156)!

\(^a\)We count the logical connectives here.

Boolean Functions

- An \( n \)-ary boolean function is a function

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

- 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.

<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:

\[ (\neg x_1 \land \neg x_2) \lor (\neg x_1 \land x_2) \lor (x_1 \land x_2). \]
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 $\{true, false, \lor, \land, \neg, x_1, x_2, \ldots\}$.

Boolean Circuits (concluded)

- Gates of sort from $\{true, false, x_1, x_2, \ldots\}$ 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:

\[
\neg x_j \\
\downarrow \\
x_j \\
\uparrow \\
x_j \lor x_j \\
\downarrow \\
x_j \\
\downarrow \\
x_j \\
\land x_j \\
\downarrow \\
x_j \\
\uparrow \\
x_j
\]

An Example

\[
((x_j \land x_2) \land (x_3 \lor x_4)) \lor (\neg (x_3 \lor x_4))
\]

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

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

Some Boolean Functions Need Exponential Circuits

Theorem 15 (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.
- There are at most \((n + 5) \times m^2\) boolean circuits with \( m \) or fewer gates (see next page).
- But \((n + 5) \times m^2 < 2^n\) when \( m = 2^n/(2n)\).
- \( m \log_2((n + 5) \times m^2) = 2^n(1 - \frac{\log_2 \frac{4n^2}{2n}}{2n}) < 2^n\) for \( n \geq 2 \).

Comments

- The lower bound is rather tight because an upper bound is \( n2^n \) (p. 149).
- In the proof, we counted the number of circuits.
- Some circuits may not be valid at all.
- Others may compute the same boolean functions.
- Both are fine because we only need an upper bound.
- We do not need to consider the outdoing edges because they have been counted in the incoming edges.
Proper (Complexity) Functions

- We say that $f : \mathbb{N} \to \mathbb{N}$ is a proper (complexity) function if the following hold:
  - $f$ is nondecreasing.
  - There is a $k$-string TM $M_f$ such that $M_f(x) = \lceil f(|x|) \rceil$ for any $x$.
  - $M_f$ 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 $f(n)$.

This point will become clear in Proposition 16 on p. 164.

Examples of Proper Functions

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

Trakhtenbrot (1964); Borodin (1972).

Space-Bounded Computation and Proper Functions

- In the definition of space-bounded computations, 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 an output of length $f(n)$ first.
  - The space-bound computation must repeat a configuration if it runs for more than $c^n + f(n)$ steps for some $c$ (p. 183).
  - So we can count steps to prevent infinite loops.
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.
  * If $M$ is a TM with input and output, we exclude the first and the last strings.

• $M$ can be deterministic or non-deterministic.

Proposition 16 Suppose a TM $M$ 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.”

*It can be deterministic or non-deterministic.

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 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)

P = TIME($n^k$),
NP = NTIME($n^k$),
PSPACE = SPACE($n^k$),
NPSPACE = NSPACE($n^k$),
E = TIME($2^{kn}$),
EXP = TIME($2^{n^k}$),
L = SPACE($\log n$),
NL = NSPACE($\log n$).

Complements of Nondeterministic Classes

• From p. 128, we know R, RE, and coRE are distinct.
  – coRE contains the complements of languages in RE, not the languages not in RE.

• Recall that the complement of $L$, denoted by $\bar{L}$, is the language $\Sigma^* - L$.
  – SAT COMPLEMENT is the set of unsatisfiable boolean expressions.
  – HAMILTONIAN PATH COMPLEMENT is the set of graphs without a Hamiltonian path.

The Co-Classes

• For any complexity class $C$, co$C$ denotes the class

$$\{\bar{L} : L \in C\}.$$

• 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 $\bar{L}$ within the same time or space bound by reversing the “yes” and “no” states.

• Whether nondeterministic classes for time are closed under complement is not known (p. 82).
Comments

- Then coC is the class 
  \[ \{ L : L \in C \} \].
- So \( L \in C \) if and only if \( \bar{L} \in \text{co}C \).
- But it is not true that \( L \in C \) if and only if \( L \not\in \text{co}C \).
- \( \text{co}C \) is not defined as \( \bar{C} \).
- For example, suppose \( C = \{ \{ 2, 4, 6, 8, 10, \ldots \} \} \).
- Then \( \text{co}C = \{ \{ 1, 3, 5, 7, 9, \ldots \} \} \).
- But \( \bar{C} = 2^{\{1,2,3,\ldots\}^*} - \{\{2,4,6,8,10,\ldots\}\} \).

The Quantified Halting Problem

- Let \( f(n) \geq n \) be proper.
- Define 
  \[ H_f = \{ M; x : M \text{ accepts input } x \} \]
  after at most \( f(|x|) \) steps, 
where \( M \) is deterministic.
- Assume the input is binary.

- For each input \( M; x \), we simulate \( M \) on \( x \) with an alarm clock of length \( f(|x|) \).
  - Use the single-string simulator (p. 66), the universal TM (p. 116), and the linear speedup theorem (p. 71).
  - Our simulator accepts \( M; x \) if and only if \( M \) accepts \( x \) before the alarm clock runs out.
- From p. 70, 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 \( M \).

\( H_f \not\in \text{TIME}(f(n)^3) \)

- Suppose TM \( M_{H_f} \) decides \( H_f \) in time \( f(\lfloor n/2 \rfloor) \).
- Consider machine \( D_f(M) \):
  \[
  \text{if } M_{H_f}(M; M) = \text{“yes” then “no” else “yes”}
  \]
- \( D_f \) 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 \( M; M \).
The Proof (concluded)

• First,

\[ D_f(D_f) = \text{“yes”} \]
\[ \Rightarrow D_f \notin H_f \]
\[ \Rightarrow D_f \text{ does not accept } D_f \text{ within time } f(|D_f|) \]
\[ \Rightarrow D_f(D_f) = \text{“no”} \]

a contradiction

• Similarly, \( D_f(D_f) = \text{“no”} \Rightarrow D_f(D_f) = \text{“yes.”} \)

The Time Hierarchy Theorem

**Theorem 17** If \( f(n) \geq n \) is proper, then

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

• The quantified halting problem makes it so.

**Corollary 18** \( \text{P} \subset \text{EXP}. \)

• \( \text{P} \subset \text{TIME}(2^n) \) because \( \text{poly}(n) \leq 2^n \) for \( n \) large enough.

• But by Theorem 17,

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

The Space Hierarchy Theorem

**Theorem 19** (Hennie and Stearns (1966)) If \( f(n) \) is proper, then

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

**Corollary 20** \( \text{L} \subset \text{PSPACE}. \)