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 zero or more literals.[a]

- For example, $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3)$ is in CNF.

- 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 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,

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

is a 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.

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

$\phi = \phi_1 \vee \phi_2$ **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

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

Any Expression ϕ Can Be Converted into CNFs and DNFs (concluded)

$\phi = \phi_1 \land \phi_2$ **and a CNF is sought:** Make ϕ_1 and ϕ_2 CNFs.

$\phi = \phi_1 \land \phi_2$ **and a DNF is sought:** Let $\phi_1 = \bigvee_{i=1}^{n_1} A_i$ and $\phi_2 = \bigvee_{i=1}^{n_2} B_i$ be DNFs. Set

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

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

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

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."

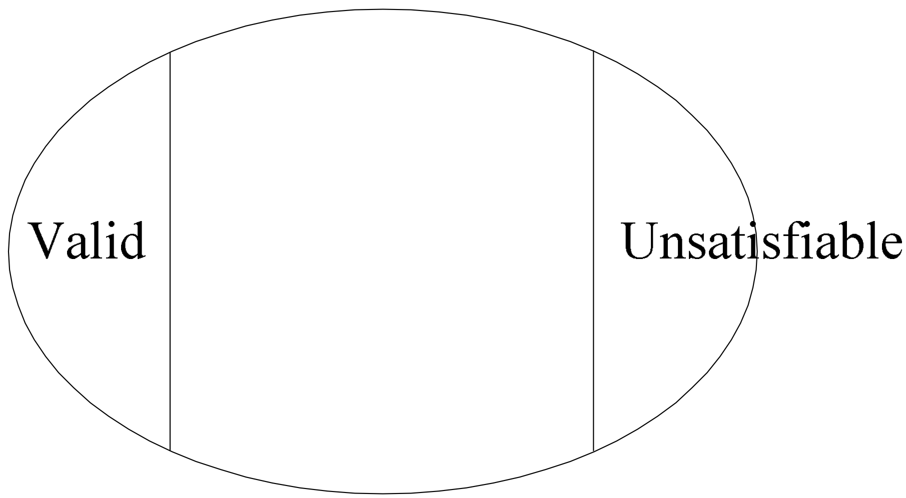
SATISFIABILITY (SAT)

- 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 exponential time on a TM by the truth table method.
- Solvable in polynomial time on an NTM, hence in NP (p. 80).
- A most important problem in answering the $\mathrm{P} = \mathrm{NP}$ problem (p. 242).

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 exponential time 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

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

- 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. 153)!

[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:

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

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

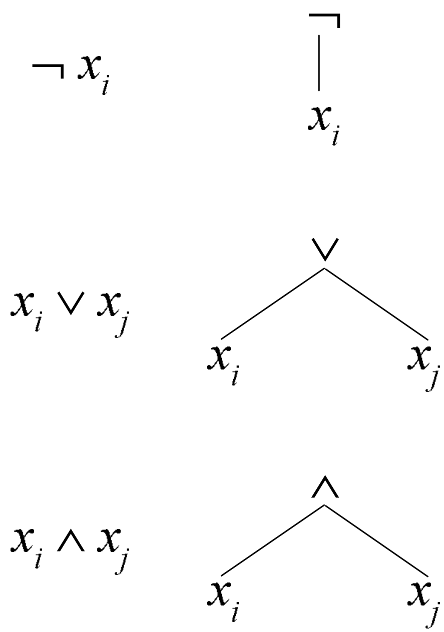
$$\{\texttt{true}, \texttt{false}, \vee, \wedge, \neg, x_1, x_2, \ldots\}.$$

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.

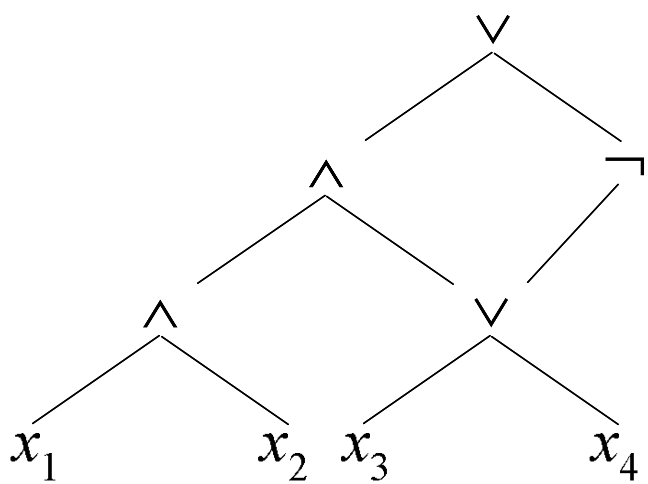
Boolean Circuits and Expressions

- 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 14 (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}.

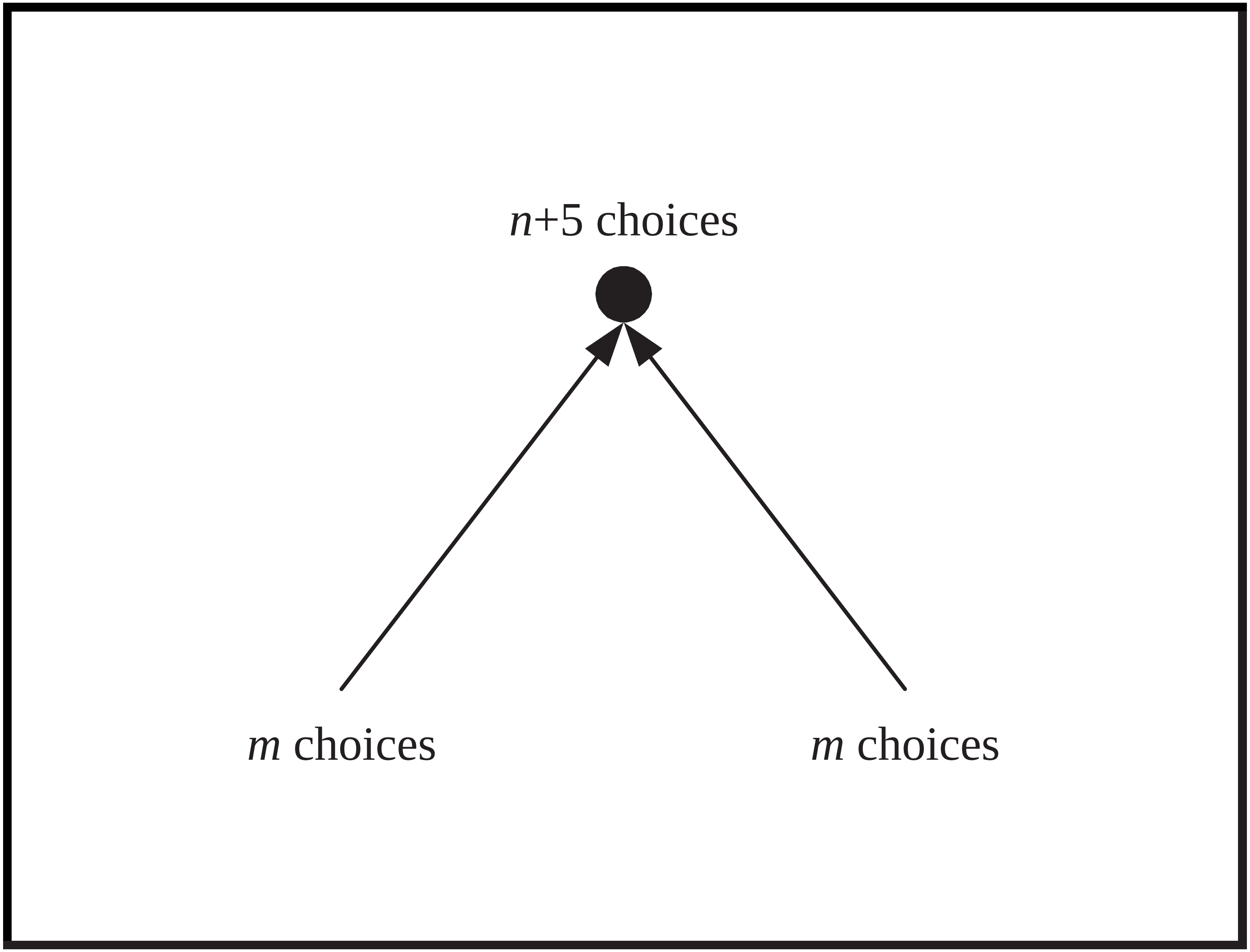
[a] Can be strengthened to "almost all boolean functions ..."

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 $m = 2^n/(2n)$:

$$
\begin{aligned}
& m \log_2((n + 5) \times m^2) \\
= \quad & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\
< \quad & 2^n
\end{aligned}
$$

for $n \geq 2$.


n+5 choices
m choices
m choices

Comments

- The lower bound is rather tight because an upper bound is $n2^n$ (p. 146).
- 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.

Relations between Complexity Classes

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 $M_f(x) = \sqcap^{f(\,|\,x\,|\,)}$ 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 15 on p. 162.

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.
- 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**).[a]
- 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] 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. 179).
  - 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 nondeterministic.

Precise TMs Are General

Proposition 15 *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.

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)

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

Complements of Nondeterministic Classes

- From p. 126, 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 $\mathcal{C}$, co$\mathcal{C}$ denotes the class

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

- Clearly, if $\mathcal{C}$ is a *deterministic* time or space *complexity class*, then $\mathcal{C} = \text{co}\mathcal{C}$.
  - 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. 78).

Comments

- Then co$\mathcal{C}$ is the class

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

  - So $L \in \mathcal{C}$ 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 $L \notin \text{co}\mathcal{C}$.
  - 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

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

where M is deterministic.

- Assume the input is binary.

$H_f \in \mathsf{TIME}(f(n)^3)$

- For each input $M; x$, we simulate M on x with an alarm clock of length $f(|\,x\,|)$.
  - Use the single-string simulator (p. 60), the universal TM (p. 112), and the linear speedup theorem (p. 66).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 65, the total running time is $O(\ell_M k_M^2 f(n)^2)$, where ℓ_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.

$H_f \notin \mathsf{TIME}(f(\lfloor n/2 \rfloor))$

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

$$\textbf{if } M_{H_f}(M;M) = \text{“yes” } \textbf{then } \text{“no” } \textbf{else } \text{“yes”}$$

- 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] A student pointed out on October 6, 2004, that this estimation omits the time to write down $M;M$.

The Proof (concluded)

- First,

$$\begin{aligned} & D_f(D_f) = \text{“yes”} \\ \Rightarrow\ & D_f; D_f \not\in H_f \\ \Rightarrow\ & D_f \text{ does not accept } D_f \text{ within time } f(\mid D_f \mid) \\ \Rightarrow\ & D_f(D_f) = \text{“no”} \end{aligned}$$

  a contradiction

- Similarly, $D_f(D_f) =$ “no” $\Rightarrow D_f(D_f) =$ “yes.”

The Time Hierarchy Theorem

Theorem 16 *If $f(n) \geq n$ is proper, then*

$$\mathrm{TIME}(f(n)) \subsetneq \mathrm{TIME}(f(2n+1)^3).$$

- The quantified halting problem makes it so.

Corollary 17 $\mathrm{P} \subsetneq \mathrm{EXP}$.

- $\mathrm{P} \subseteq \mathrm{TIME}(2^n)$ because $\mathrm{poly}(n) \leq 2^n$ for n large enough.
- But by Theorem 16,

$$\mathrm{TIME}(2^n) \subsetneq \mathrm{TIME}((2^{2n+1})^3) \subseteq \mathrm{TIME}(2^{n^2}) \subseteq \mathrm{EXP}.$$

The Space Hierarchy Theorem

Theorem 18 (Hennie and Stearns (1966)) *If $f(n)$ is proper, then*

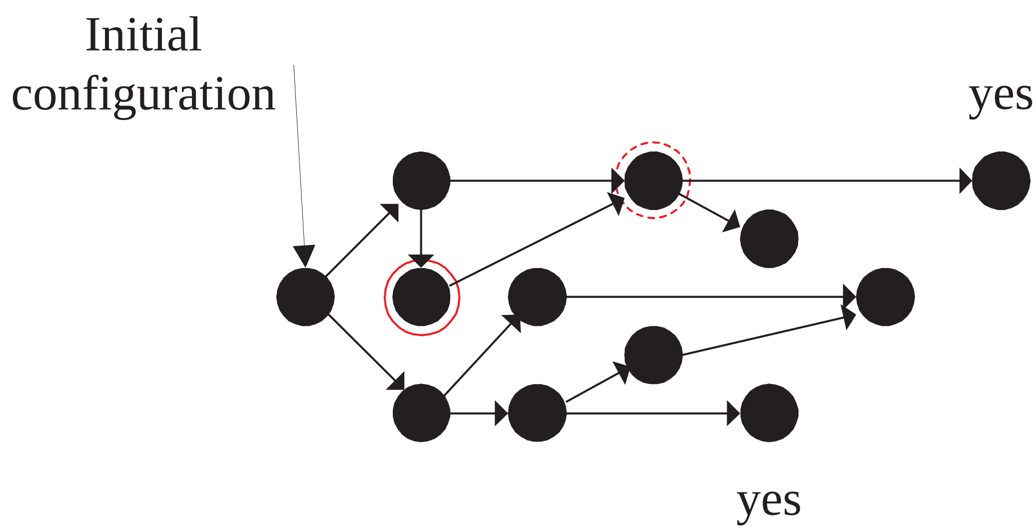
$$\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n)\log f(n)).$$

Corollary 19 $\text{L} \subsetneq \text{PSPACE}$.

The Reachability Method

- The computation of a time-bounded TM can be represented by directional transitions between configurations.
- The reachability method constructs a directed graph with all the TM configurations as its nodes and edges connecting two nodes if one yields the other.
- The start node representing the initial configuration has zero in degree.
- When the TM is nondeterministic, a node may have an out degree greater than one.

Illustration of the Reachability Method



The reachability method may give the edges on the fly without explicitly storing the whole configuration graph.

Relations between Complexity Classes

Theorem 20 *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. $\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{\log n + f(n)})$.

- Proof of 2:
  - Explore the computation *tree* of the NTM for "yes."
  - Use the *depth-first* search as f is proper.

Proof of Theorem 20(2)

- (continued)
  - Specifically, generate a $f(n)$-bit sequence denoting the nondeterministic choices over $f(n)$ steps.
  - Simulate the NTM based on the choices.
  - Recycle the space and then repeat the above steps until a "yes" is encountered or 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 $O(f(n))$ as space is recycled.

Proof of Theorem 20(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 20(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

$$|K| \times (n+1) \times |\Sigma|^{(2k-4)f(n)} = O(c_1^{\log n + f(n)}) \qquad (2)$$

for some c_1, which depends on M.

- Add edges to the configuration graph based on M's transition function.

Proof of Theorem 20(3) (concluded)

- $x \in L \Leftrightarrow$ there is a path in the configuration graph from the initial configuration to a configuration of the form ("yes", $i, \ldots$) [there may be many of them].
- The problem is therefore that of REACHABILITY on a graph with $O(c_1^{\log n + f(n)})$ nodes.
- It is in TIME($c^{\log n + f(n)}$) for some c because REACHABILITY is in TIME(n^k) for some k and

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

The Grand Chain of Inclusions

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP.$$

- By Corollary 19 (p. 173), we know $L \subsetneq PSPACE$.
- The chain must break somewhere between L and PSPACE.
- It is suspected that all four inclusions are proper.
- But there are no proofs yet.[a]

[a] Carl Friedrich Gauss (1777–1855), "I could easily lay down a multitude of such propositions, which one could neither prove nor dispose of."

Nondeterministic Space and Deterministic Space

- By Theorem 5 (p. 88),

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

  an exponential gap.

- There is no proof that the exponential gap is inherent, however.
- How about NSPACE vs. SPACE?
- Surprisingly, the relation is only quadratic, a polynomial, by Savitch's theorem.

Savitch's Theorem

Theorem 21 (Savitch (1970))

$$\text{REACHABILITY} \in \text{SPACE}(\log^2 n).$$

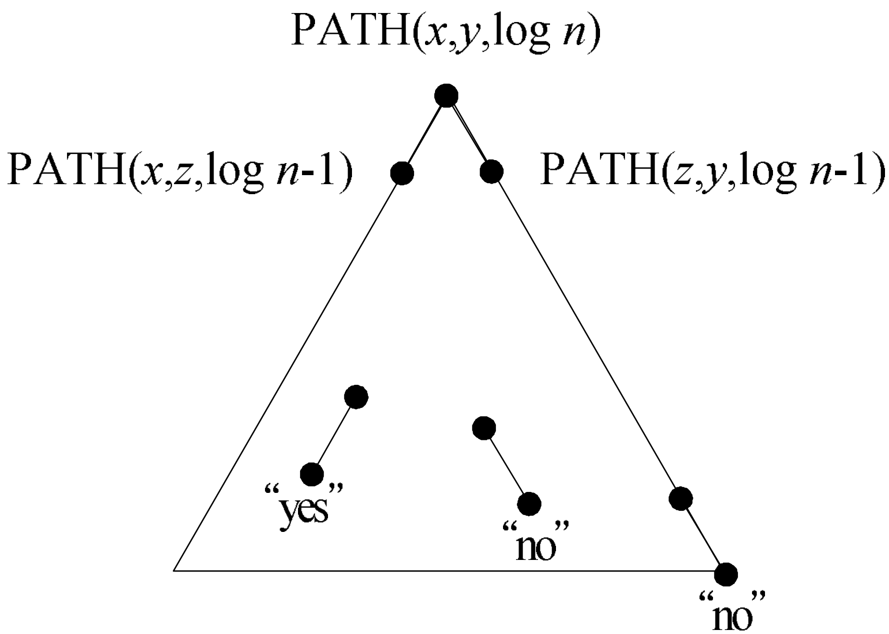
- Let G be a graph with n nodes.
- For $i \geq 0$, let

  $$\text{PATH}(x, y, i)$$

  mean there is a path from node x to node y of length at most 2^i.
- There is a path from x to y if and only if $\text{PATH}(x, y, \lceil \log n \rceil)$ holds.

The Proof (continued)

- For $i > 0$, PATH(x, y, i) if and only if there exists a z such that PATH$(x, z, i-1)$ and PATH$(z, y, i-1)$.
- For PATH$(x, y, 0)$, check the input graph or if $x = y$.
- Compute PATH$(x, y, \lceil \log n \rceil)$ with a depth-first search on a graph with nodes (x, y, i)s (see next page).
- Like stacks in recursive calls, we keep only the current path of (x, y, i)s.
- The space requirement is proportional to the depth of the tree, $\lceil \log n \rceil$.



- Depth is $\lceil \log n \rceil$, and each node (x, y, i) needs space $O(\log n)$.
- The total space is $O(\log^2 n)$.

The Proof (concluded): Algorithm for PATH(x, y, i)

```
if i = 0 then
    if x = y or (x, y) ∈ G then
        return true;
    else
        return false;
    end if
else
    for z = 1, 2, ..., n do
        if PATH(x, z, i − 1) and PATH(z, y, i − 1) then
            return true;
        end if
    end for
    return false;
end if
```

The Relation between Nondeterministic Space and Deterministic Space Only Quadratic

Corollary 22 *Let $f(n) \geq \log n$ be proper. Then*

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n)).$$

- Apply Savitch's theorem to the configuration graph of the NTM on the input.
- From p. 179, the configuration graph has $O(c^{f(n)})$ nodes; hence each node takes space $O(f(n))$.
- But if we supply the whole graph before applying Savitch's theorem, we get $O(c^{f(n)})$ space!

The Proof (continued)

- The way out is *not* to generate the graph at all.
- Instead, keep the graph implicit.
- We check for connectedness only when $i = 0$, by examining the input string.
- There, given configurations x and y, we go over the Turing machine's program to determine if there is an instruction that can turn x into y in one step.[a]

[a] Thanks to a lively class discussion on October 15, 2003.

The Proof (concluded)

- The z variable in the algorithm simply runs through all possible valid configurations.
- Each z has length $O(f(n))$ by Eq. (2) on p. 179.
- An alternative is to let $z = 0, 1, \ldots, O(c^{f(n)})$ and makes sure it is a valid configuration before using it in the recursive calls.[a]

---

[a]Thanks to a lively class discussion on October 13, 2004.

Implications of Savitch's Theorem

- PSPACE = NPSPACE.
- Nondeterminism is less powerful with respect to space.
- It may be very powerful with respect to time as it is not known if P = NP.

Nondeterministic Space Is Closed under Complement

- Closure under complement is trivially true for deterministic complexity classes (p. 166).
- It is known that[a]

$$\text{coNSPACE}(f(n)) = \text{NSPACE}(f(n)). \tag{3}$$

- So

$$\begin{aligned} \text{coNL} &= \text{NL}, \\ \text{coNPSPACE} &= \text{NPSPACE}. \end{aligned}$$

- But there are still no hints of coNP $=$ NP.

[a]Szelepscényi (1987) and Immerman (1988).