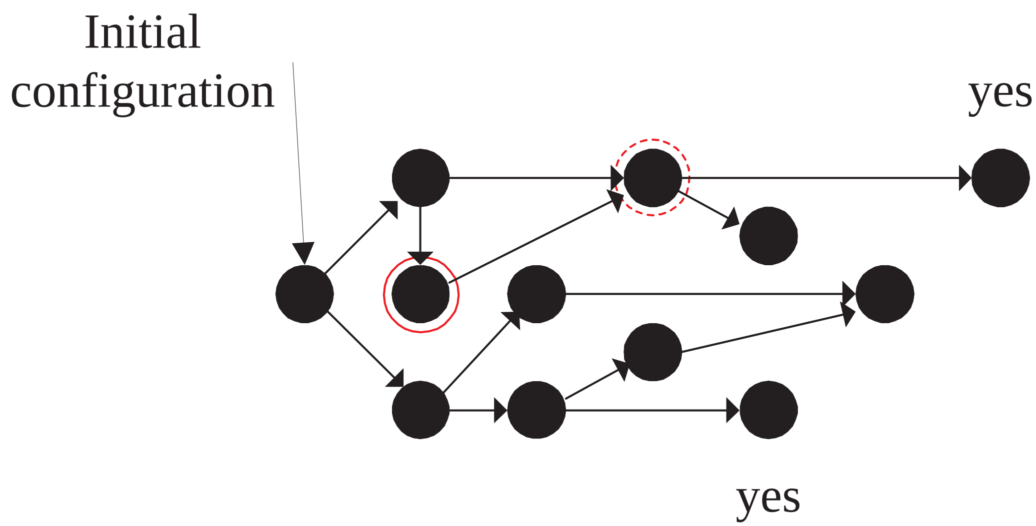
# The Reachability Method

- 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.
- 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



## Relations between Complexity Classes

**Theorem 22** *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."
  - Specifically, generate a $f(n)$-bit sequence denoting the nondeterministic choices over $f(n)$ steps.

# Proof of Theorem 22(2)

- (continued)
  - 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 $O(f(n))$ because space is recycled.

## Proof of Theorem 22(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 22(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

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

for some $c_1$, which depends on $M$.

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

# Proof of Theorem 22(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].
- This is REACHABILITY on a graph with $O(c_1^{\log n + f(n)})$ nodes.
- It is in $\text{TIME}(c^{\log n + f(n)})$ for some $c$ because REACHABILITY $\in \text{TIME}(n^j)$ for some $j$ and

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

## Space-Bounded Computation and Proper Functions

- In the definition of *space-bounded* computations earlier, 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. 198).
  - So we can prevent infinite loops by counting steps against $c^{n+f(n)}$.

# The Grand Chain of Inclusions

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

- By Corollary 19 (p. 191), we know $L \subsetneq PSPACE$.
- The chain must break somewhere between L and PSPACE.[a]
- It is suspected that all four inclusions are proper.
- But there are no proofs yet.[b]

[a]Bill Gates (1996), "I keep bumping into that silly quotation attributed to me that says 640K of memory is enough."

[b]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 4 (p. 87),

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

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.

# Savitch's Theorem

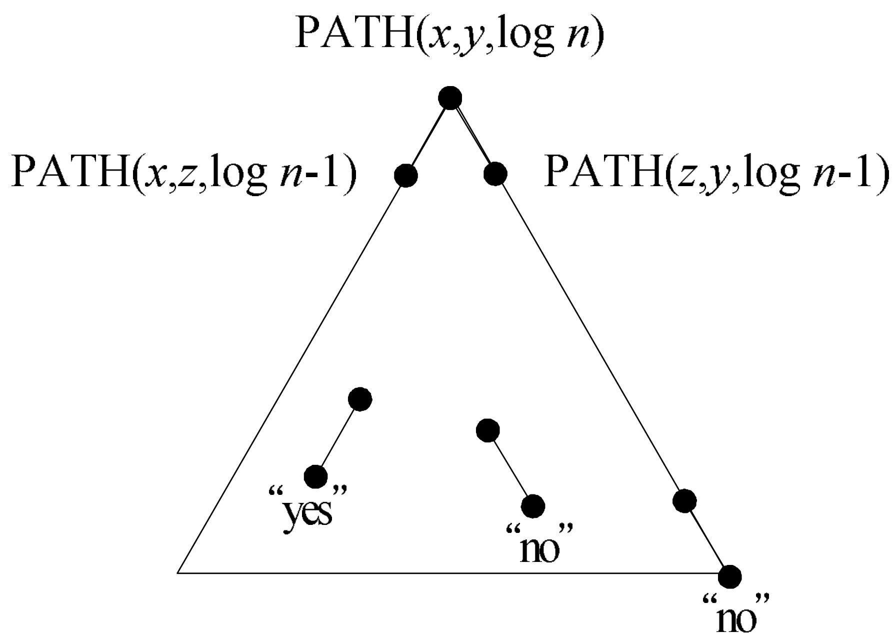
**Theorem 23 (Savitch (1970))**

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

- Let $G(V, E)$ 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) ∈ E 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 24** *Let $f(n) \geq \log n$ be proper. Then*

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

- Apply Savitch's theorem to the configuration graph of the NTM on the input.
- From p. 198, the configuration graph has $O(c^{f(n)})$ nodes; hence each node takes space $O(f(n))$.
- But if we construct explicitly 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 $i = 0$ on p. 206, by examining the input string $G$.
- 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 on p. 206 simply runs through all possible valid configurations.
  - Let $z = 0, 1, \ldots, O(c^{f(n)})$.
  - Make sure $z$ is a valid configuration before using it in the recursive calls.[a]
- Each $z$ has length $O(f(n))$ by Eq. (1) on p. 198.

[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.
- Nondeterminism 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. 184).
- It is known that[a]

$$\text{coNSPACE}(f(n)) = \text{NSPACE}(f(n)). \tag{2}$$

- 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).

*Reductions and Completeness*

## Degrees of Difficulty

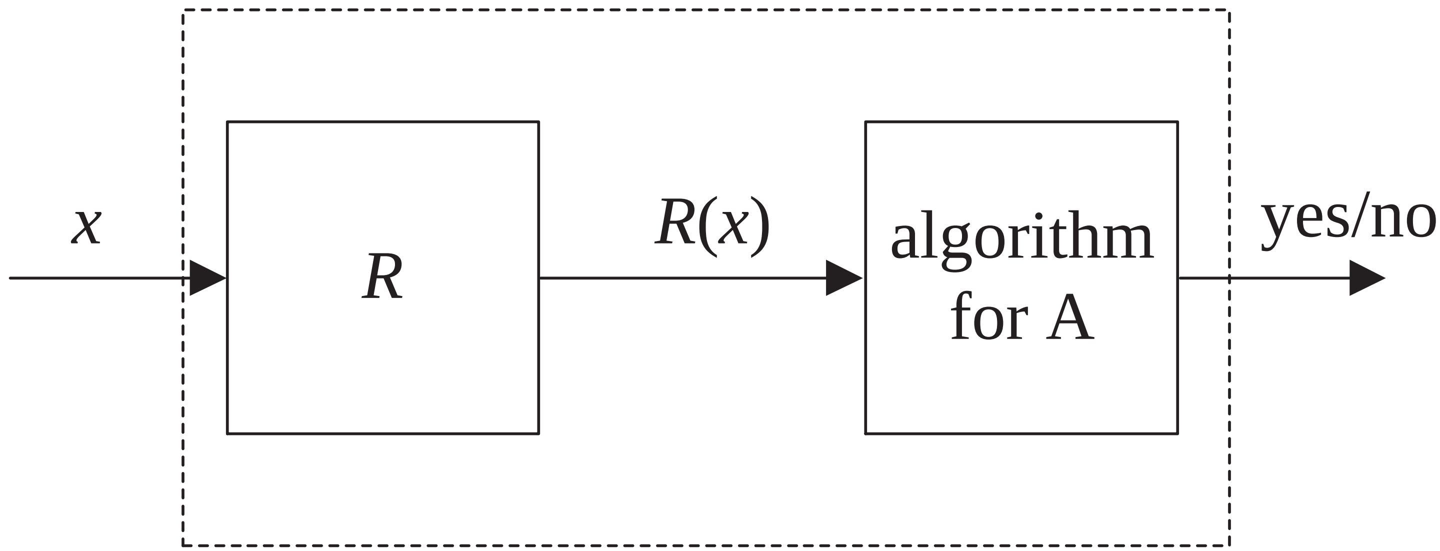
- When is a problem more difficult than another?
- B **reduces to** A if there is a transformation $R$ which for every input $x$ of B yields an equivalent input $R(x)$ of A.
  - The answer to $x$ for B is the same as the answer to $R(x)$ for A.
  - There must be restrictions on the complexity of computing $R$.
  - Otherwise, $R(x)$ might as well solve B.
    - E.g., $R(x) =$ "yes" if and only if $x \in$ B!

## Degrees of Difficulty (concluded)

- We say problem A is at least as hard as problem B if B reduces to A.
- This makes intuitive sense: If A is able to solve your problem B after only a little bit of work ($R$), then A must be at least as hard.
  - If A were easy, it combined with $R$ (which is also easy) would make B easy, too.[a]

[a]Thanks to a lively class discussion on October 13, 2009.

## Reduction




Solving problem B by calling the algorithm for problem *once* and *without* further processing its answer.

# Comments[a]

- Suppose B reduces to A via a transformation $R$.
- The input $x$ is an instance of B.
- The output $R(x)$ is an instance of A.
- $R(x)$ may not span all possible instances of A.[b]
- So some instances of A may never appear in the range of the reduction $R$.

---

[a] Contributed by Mr. Ming-Feng Tsai (`D92922003`) on October 29, 2003.

[b] $R(x)$ may not be onto; Mr. Alexandr Simak (`D98922040`) on October 13, 2009.

## Reduction between Languages

- Language $L_1$ is **reducible to** $L_2$ if there is a function $R$ computable by a deterministic TM in space $O(\log n)$.
- Furthermore, for all inputs $x$, $x \in L_1$ if and only if $R(x) \in L_2$.
- $R$ is said to be a (**Karp**) **reduction** from $L_1$ to $L_2$.

## Reduction between Languages (concluded)

- Note that by Theorem 22 (p. 195), $R$ runs in polynomial time.
  - In most cases, you do not need to distinguish between L and P in proofs involving reductions.
- Suppose $R$ is a reduction from $L_1$ to $L_2$.
- Then solving "$R(x) \in L_2$?" is an algorithm for solving "$x \in L_1$?"[a]

[a] But it may not be an optimal one.

# A Paradox?

- Degree of difficulty is not defined in terms of *absolute* complexity.
- So a language B $\in$ TIME($n^{99}$) may be "easier" than a language A $\in$ TIME($n^3$).
  - This happens when B is reducible to A.
- But isn't this a contradiction if the best algorithm for B requires $n^{99}$ steps?
- That is, how can a problem *requiring* $n^{99}$ steps be reducible to a problem solvable in $n^3$ steps?

## Paradox Resolved

- The so-called contradiction does not hold.
- When we solve the problem "$x \in$ B?" via "$R(x) \in$ A?", we must consider the time spent by $R(x)$ and its length $|\, R(x) \,|$.
- If $|\, R(x) \,| = \Omega(n^{33})$, then answering "$R(x) \in$ A?" takes $\Omega((n^{33})^3) = \Omega(n^{99})$ steps, which is fine.
- Suppose, on the other hand, that $|\, R(x) \,| = o(n^{33})$.
- Then $R(x)$ must run in time $\Omega(n^{99})$ to make the overall time for answering "$R(x) \in$ A?" take $\Omega(n^{99})$ steps.
- In either case, the contradiction disappears.

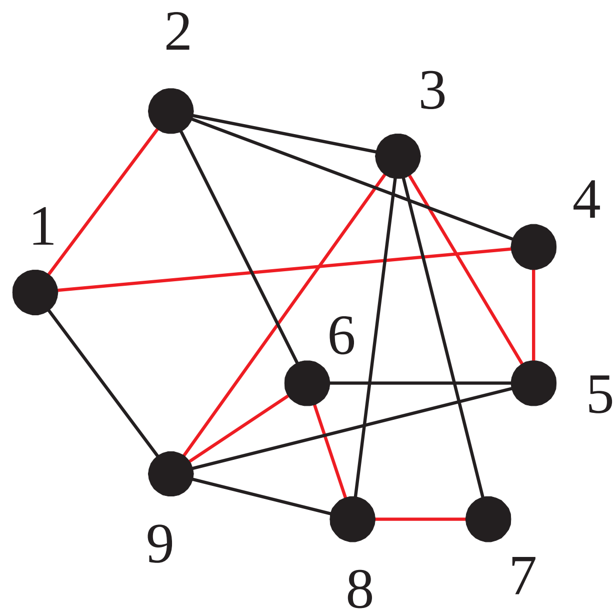
## HAMILTONIAN PATH

- A **Hamiltonian path** of a graph is a path that visits every node of the graph exactly once.
- Suppose graph $G$ has $n$ nodes: $1, 2, \ldots, n$.
- A Hamiltonian path can be expressed as a permutation $\pi$ of $\{ 1, 2, \ldots, n \}$ such that
  - $\pi(i) = j$ means the $i$th position is occupied by node $j$.
  - $(\pi(i), \pi(i+1)) \in G$ for $i = 1, 2, \ldots, n-1$.
- HAMILTONIAN PATH asks if a graph has a Hamiltonian path.

## Reduction of HAMILTONIAN PATH to SAT

- Given a graph $G$, we shall construct a CNF $R(G)$ such that $R(G)$ is satisfiable iff $G$ has a Hamiltonian path.
- $R(G)$ has $n^2$ boolean variables $x_{ij}$, $1 \leq i, j \leq n$.
- $x_{ij}$ means

  the $i$th position in the Hamiltonian path is occupied by node $j$.




$x_{12} = x_{21} = x_{34} = x_{45} = x_{53} = x_{69} = x_{76} = x_{88} = x_{97} = 1$; $\pi(1) = 2, \pi(2) = 1, \pi(3) = 4, \pi(4) = 5, \pi(5) = 3, \pi(6) = 9, \pi(7) = 6, \pi(8) = 8, \pi(9) = 7$.

# The Clauses of $R(G)$ and Their Intended Meanings

1. Each node $j$ must appear in the path.
   - $x_{1j} \lor x_{2j} \lor \cdots \lor x_{nj}$ for each $j$.
2. No node $j$ appears twice in the path.
   - $\neg x_{ij} \lor \neg x_{kj}$ for all $i, j, k$ with $i \neq k$.
3. Every position $i$ on the path must be occupied.
   - $x_{i1} \lor x_{i2} \lor \cdots \lor x_{in}$ for each $i$.
4. No two nodes $j$ and $k$ occupy the same position in the path.
   - $\neg x_{ij} \lor \neg x_{ik}$ for all $i, j, k$ with $j \neq k$.
5. Nonadjacent nodes $i$ and $j$ cannot be adjacent in the path.
   - $\neg x_{ki} \lor \neg x_{k+1,j}$ for all $(i, j) \not\in G$ and $k = 1, 2, \ldots, n-1$.

# The Proof

- $R(G)$ contains $O(n^3)$ clauses.
- $R(G)$ can be computed efficiently (simple exercise).
- Suppose $T \models R(G)$.
- From clauses of 1 and 2, for each node $j$ there is a unique position $i$ such that $T \models x_{ij}$.
- From clauses of 3 and 4, for each position $i$ there is a unique node $j$ such that $T \models x_{ij}$.
- So there is a permutation $\pi$ of the nodes such that $\pi(i) = j$ if and only if $T \models x_{ij}$.

## The Proof (concluded)

- Clauses of 5 furthermore guarantee that $(\pi(1), \pi(2), \ldots, \pi(n))$ is a Hamiltonian path.
- Conversely, suppose $G$ has a Hamiltonian path

$$(\pi(1), \pi(2), \ldots, \pi(n)),$$

  where $\pi$ is a permutation.
- Clearly, the truth assignment

$$T(x_{ij}) = \texttt{true} \text{ if and only if } \pi(i) = j$$

  satisfies all clauses of $R(G)$.

# A Comment[a]

- An answer to "Is $R(G)$ satisfiable?" does answer "Is $G$ Hamiltonian?"
- But a positive answer does not give a Hamiltonian path for $G$.
  - *Providing* witness is not a requirement of reduction.
- A positive answer to "Is $R(G)$ satisfiable?" plus a satisfying truth assignment does provide us with a Hamiltonian path for $G$.

---
[a] Contributed by Ms. Amy Liu (`J94922016`) on May 29, 2006.

## Reduction of REACHABILITY to CIRCUIT VALUE

- Note that both problems are in P.
- Given a graph $G = (V, E)$, we shall construct a *variable-free* circuit $R(G)$.
- The output of $R(G)$ is true if and only if there is a path from node 1 to node $n$ in $G$.
- Idea: the Floyd-Warshall algorithm.

# The Gates

- The gates are
  - $g_{ijk}$ with $1 \leq i, j \leq n$ and $0 \leq k \leq n$.
  - $h_{ijk}$ with $1 \leq i, j, k \leq n$.
- $g_{ijk}$: There is a path from node $i$ to node $j$ without passing through a node bigger than $k$.
- $h_{ijk}$: There is a path from node $i$ to node $j$ passing through $k$ but not any node bigger than $k$.
- Input gate $g_{ij0} = \texttt{true}$ if and only if $i = j$ or $(i, j) \in E$.

## The Construction

- $h_{ijk}$ is an AND gate with predecessors $g_{i,k,k-1}$ and $g_{k,j,k-1}$, where $k = 1, 2, \ldots, n$.
- $g_{ijk}$ is an OR gate with predecessors $g_{i,j,k-1}$ and $h_{i,j,k}$, where $k = 1, 2, \ldots, n$.
- $g_{1nn}$ is the output gate.
- Interestingly, $R(G)$ uses no $\neg$ gates: It is a **monotone circuit**.

## Reduction of CIRCUIT SAT to SAT

- Given a circuit $C$, we will construct a boolean expression $R(C)$ such that $R(C)$ is satisfiable iff $C$ is.
  - $R(C)$ will turn out to be a CNF.
  - $R(C)$ is a depth-2 circuit; furthermore, each gate has out-degree 1.
- The variables of $R(C)$ are those of $C$ plus $g$ for each gate $g$ of $C$.
  - The $g$'s propagate the truth values for the CNF.
- Each gate of $C$ will be turned into equivalent clauses.
- Recall that clauses are $\land$-ed together by definition.

## The Clauses of $R(C)$

$g$ **is a variable gate** $x$**:** Add clauses $(\neg g \lor x)$ and $(g \lor \neg x)$.

- Meaning: $g \Leftrightarrow x$.

$g$ **is a `true` gate:** Add clause $(g)$.

- Meaning: $g$ must be true to make $R(C)$ true.

$g$ **is a `false` gate:** Add clause $(\neg g)$.

- Meaning: $g$ must be false to make $R(C)$ true.

$g$ **is a** $\neg$ **gate with predecessor gate** $h$**:** Add clauses $(\neg g \lor \neg h)$ and $(g \lor h)$.

- Meaning: $g \Leftrightarrow \neg h$.

## The Clauses of $R(C)$ (concluded)

**$g$ is a $\vee$ gate with predecessor gates $h$ and $h'$:** Add clauses $(\neg h \vee g)$, $(\neg h' \vee g)$, and $(h \vee h' \vee \neg g)$.

- Meaning: $g \Leftrightarrow (h \vee h')$.

**$g$ is a $\wedge$ gate with predecessor gates $h$ and $h'$:** Add clauses $(\neg g \vee h)$, $(\neg g \vee h')$, and $(\neg h \vee \neg h' \vee g)$.

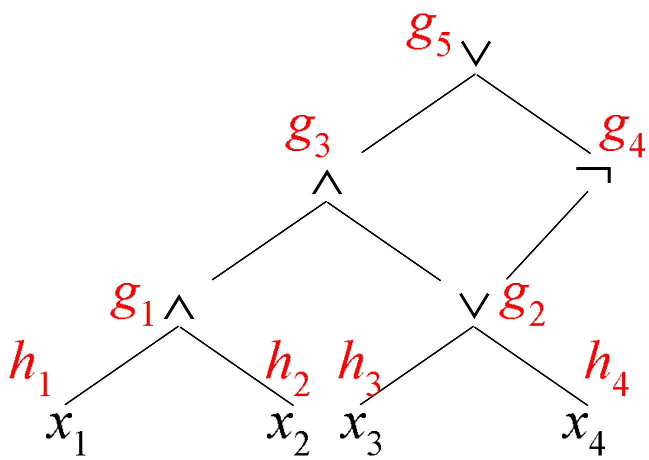
- Meaning: $g \Leftrightarrow (h \wedge h')$.

**$g$ is the output gate:** Add clause $(g)$.

- Meaning: $g$ must be true to make $R(C)$ true.

Note: If gate $g$ feeds gates $h_1, h_2, \ldots$, then variable $g$ appears in the clauses for $h_1, h_2, \ldots$ in $R(C)$.

# An Example




$$
\begin{array}{ll}
 & (h_1 \Leftrightarrow x_1) \land (h_2 \Leftrightarrow x_2) \land (h_3 \Leftrightarrow x_3) \land (h_4 \Leftrightarrow x_4) \\
\land & [\, g_1 \Leftrightarrow (h_1 \land h_2)\,] \land [\, g_2 \Leftrightarrow (h_3 \lor h_4)\,] \\
\land & [\, g_3 \Leftrightarrow (g_1 \land g_2)\,] \land (g_4 \Leftrightarrow \neg g_2) \\
\land & [\, g_5 \Leftrightarrow (g_3 \lor g_4)\,] \land g_5.
\end{array}
$$

## An Example (concluded)

- In general, the result is a CNF.
- The CNF has size proportional to the circuit's number of gates.
- The CNF adds new variables to the circuit's original input variables.

## Composition of Reductions

**Proposition 25** *If* $R_{12}$ *is a reduction from* $L_1$ *to* $L_2$ *and* $R_{23}$ *is a reduction from* $L_2$ *to* $L_3$*, then the composition* $R_{12} \circ R_{23}$ *is a reduction from* $L_1$ *to* $L_3$*.*

- So reducibility is transitive.

## Completeness[a]

- As reducibility is transitive, problems can be ordered with respect to their difficulty.
- Is there a *maximal* element?
- It is not altogether obvious that there should be a maximal element.
  - Many infinite structures (such as integers and real numbers) do not have maximal elements.
- Hence it may surprise you that most of the complexity classes that we have seen so far have maximal elements.

[a]Cook (1971) and Levin (1971).

## Completeness (concluded)

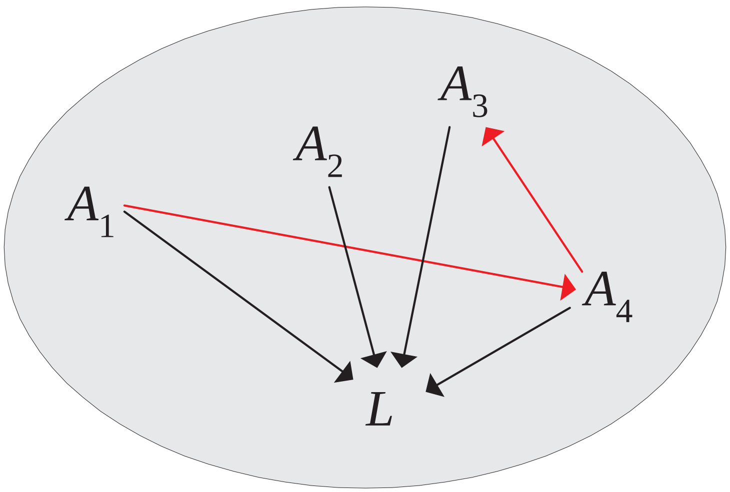
- Let $\mathcal{C}$ be a complexity class and $L \in \mathcal{C}$.
- $L$ is **$\mathcal{C}$-complete** if every $L' \in \mathcal{C}$ can be reduced to $L$.
  - Most complexity classes we have seen so far have complete problems!
- Complete problems capture the difficulty of a class because they are the hardest problems in the class.

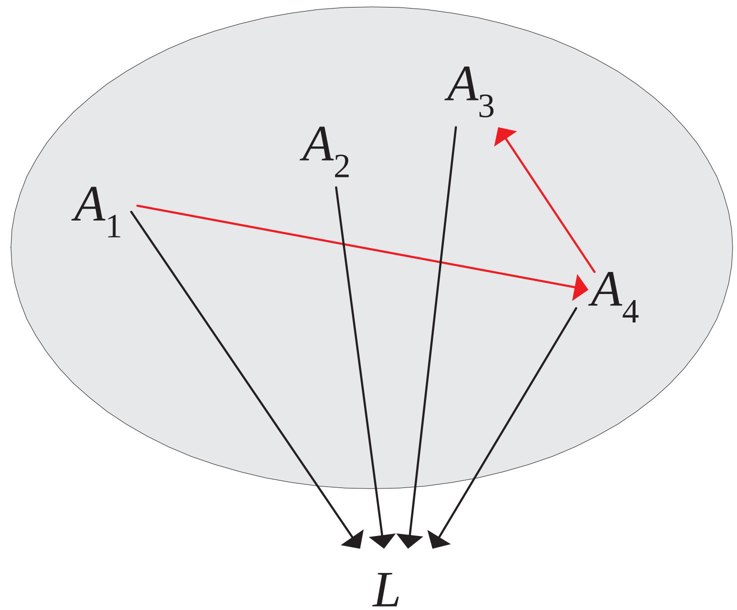
# Hardness

- Let $\mathcal{C}$ be a complexity class.
- $L$ is $\mathcal{C}$**-hard** if every $L' \in \mathcal{C}$ can be reduced to $L$.
- It is not required that $L \in \mathcal{C}$.
- If $L$ is $\mathcal{C}$-hard, then by definition, every $\mathcal{C}$-complete problem can be reduced to $L$.[a]

[a] Contributed by Mr. Ming-Feng Tsai (`D92922003`) on October 15, 2003.

## Illustration of Completeness and Hardness



## Closedness under Reductions

- A class $\mathcal{C}$ is **closed under reductions** if whenever $L$ is reducible to $L'$ and $L' \in \mathcal{C}$, then $L \in \mathcal{C}$.
- P, NP, coNP, L, NL, PSPACE, and EXP are all closed under reductions.