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.

$H_f \not\in \text{TIME}(f([n/2]))$

- Suppose there is a TM $M_{H_f}$ deciding $H_f$ in time $f([n/2])$.
- Consider machine $D_f(M)$:
  
  \[
  \text{if } M_{H_f}(M; M) = \text{"yes" then } \text{"no" else } \text{"yes"}
  \]

- $D_f$ on input $M$ runs in the same time as $M_{H_f}$ on input $M; M$, i.e., in time $f(\lceil \frac{2n+1}{2} \rceil) = f(n)$.
- $D_f(D_f) = \text{"yes" } \Rightarrow \text{ } D_f \not\in H_f \Rightarrow D_f(D_f) = \text{"no,"}$
- Similarly, $D_f(D_f) = \text{"no" } \Rightarrow \text{ } D_f(D_f) = \text{"yes,"}.$

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

- For each input $M; x$, we simulate $M$ on $x$ with an alarm clock of length $f(|x|)$.
  - Use the single-string simulator (p. 57), the universal TM (p. 107), and the linear speedup theorem (p. 62).
- From p. 61, the total running time is $O(\ell k^2 f(n)^2)$, where $\ell$ is the length to encode each symbol or state of $M$ and $k$ is $M$'s number of strings.
- As $\ell = O(\log n)$, the running time is $O(f(n)^3)$, where the constant is independent of $M$.

The Time Hierarchy Theorem

**Theorem 16** 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 17** $P \subset EXP$.

- $P \subset \text{TIME}(2^n)$ because $\text{poly}(n) \leq 2^n$ for $n$ large enough.
- But by Theorem 16,
  \[ \text{TIME}(2^n) \subset \text{TIME}((2^{2n+1})^3) \subset \text{TIME}(2^{n^2}) \subset \text{EXP}. \]
The Space Hierarchy Theorem

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

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

**Corollary 19** $L \subset \text{PSPACE}$.

---

The Reachability Method

- A computation of a 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

Initial configuration

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. $\text{SPACE}(f(n)) \subset \text{NSPACE}(f(n))$,
   $\text{TIME}(f(n)) \subset \text{NTIME}(f(n))$.
2. $\text{NTIME}(f(n)) \subset \text{SPACE}(f(n))$.
3. $\text{NSPACE}(f(n)) \subset \text{TIME}(k^{\log n + f(n)})$.

- 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) (continued)

- We only care about
  \[
  (q_1, b \ w_2, w_2, \ldots, w_k, 1, u_k),
  \]
  where \( i \) is an integer between 0 and \( n \) for the position of the first cursor.
- The number of configurations is therefore at most
  \[
  |K| \times (n + 1) \times |\Sigma|^{(2k + 4)f(n)} = O(c_1^{\log n + f(n)})
  \]  
for some \( c_1 \), which depends on \( M \).
- Add edges to the configuration graph based on the transition function.

Proof of Theorem 20(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”, \( s \ldots \)) [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 \( \text{TIME}(c_1^{\log n + f(n)}) \) for some \( c \) because reachability is in \( \text{TIME}(n^k) \) for some \( k \) and
  \[
  \left( c_1^{\log n + f(n)} \right)^k = (c_1^k)^{\log n + f(n)}.
  \]
A Corollary of the Reachability Method

**Corollary 21** For any NTM $M$ in $\text{NSPACE}(f(n))$, where $f(n) = \Omega(\log n)$, there is a TM in $\text{SPACE}(f(n))$ that writes out the configuration graph of $M(x)$, given input $x$.

- From the proof of Theorem 20 (p. 169), especially Eq. (3) on p. 172, the number of configurations is $O(c^{f(n)})$ for some constant $c$.
- Use two counters each with space $O(f(n))$ to enumerate all possible pairs of configurations, $(C_1, C_2)$.
- Write $(C_1, C_2)$ to the output string if $C_1$ yields $C_2$.

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 $\text{NSPACE}$ vs. $\text{SPACE}$?
- Surprisingly, the relation is only quadratic, a polynomial, by Savitch’s theorem.

The Grand Chain of Inclusions

$L \subseteq NL \subseteq P \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP}$.

- It is known that $\text{PSPACE} \subseteq \text{EXP}$.
- By Corollary 19 (p. 166), we know $L \subseteq \text{PSPACE}$.
- The chain must break somewhere between $L$ and $\text{PSPACE}$.
- We suspect all four inclusions are proper.
- But there are no proofs yet.

Savitch’s Theorem

**Theorem 22** (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 $\text{PATH}(x, y, \lfloor \log n \rfloor)$ holds.
The Simple Idea for Computing \( \text{PATH}(x, y, i) \)

- For \( i > 0 \), \( \text{PATH}(x, y, i) \) if and only if there exists a \( z \) such that \( \text{PATH}(x, z, i - 1) \) and \( \text{PATH}(z, y, i - 1) \).
- For \( \text{PATH}(x, y, 0) \), check the input graph or if \( x = y \).
- We compute \( \text{PATH}(x, y, \lceil \log n \rceil) \) with a depth-first search on a tree with nodes \((x, y, i)\)s.
- 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, \( \lceil \log n \rceil \).

The Algorithm for \( \text{PATH}(x, y, i) \)

1. if \( i = 0 \) then
2. if \( x = y \) or \((x, y) \in G\) then
3. return true;
4. else
5. return false;
6. end if
7. else
8. for \( z = 1, 2, \ldots, n \) do
9. if \( \text{PATH}(x, z, i - 1) \) and \( \text{PATH}(z, y, i - 1) \) then
10. return true;
11. end if
12. end for
13. return false;
14. end if

The Relation between Nondeterministic Space and Deterministic Space Only Quadratic

**Corollary 23** 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. 172, the configuration graph has \( O(e^f(n)) \) nodes; hence each node takes space \( O(f(n)) \).
- But the graph is implicit we check for connectedness only when \( i = 0 \), by examining the input string.
Implications of Savitch's Theorem

- PSPACE = NSPACE.
- 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.

Functions and Nondeterministic TMs

- An NTM computes function $F$ if the following hold:
  - On input $x$, each computation path either outputs the correct answer $F(x)$ or ends up in state “no.”
  - At least one computation path ends up with $F(x)$.
- So all successful paths agree on their output.
- Existence of output indicates successful computation.
- As before, the machine observes a space bound $f(n)$ if at halting all strings (except for the input and output ones) are of length at most $f(|x|)$.

Nondeterministic Space Is Closed under Complement

- Closure under complement is trivially true for deterministic complexity classes (p. 160).
- On p. 186, we shall prove
  $$\text{coNSPACE}(f(n)) = \text{NSPACE}(f(n)).$$
- So
  $$\text{coNL} = \text{NL},$$
  $$\text{coPSPACE} = \text{NPSPACE},$$
- But there are still no hints of $\text{coNP} = \text{NP}$.

How an NTM Computes a Function

- $x$
The Immerman-Szelepscényi Theorem

Theorem 24 (Szelepscényi, 1987, Immerman, 1988)
Given a graph $G$ and a node $x$, the number of nodes reachable from $x$ in $G$ can be computed by an NTM within space $O(\log n)$.

- The algorithm has four nested loops.
- Let $n$ be the number of nodes.
- $S(k)$ denotes the set of nodes in $G$ that can be reached from $x$ by paths of length at most $k$.
- So $|S(n - 1)|$ is the desired answer.

The Algorithm: Top 2 Levels
1: $|S(0)| := 1$;
2: for $k = 1, 2, \ldots, n$ do
3:  \{ Compute $|S(k)|$ from $|S(k - 1)|$ saved in previous loop. \}
4:  $\ell := 0$
5:  for $u = 1, 2, \ldots, n$ do
6:    if $u \in S(k)$ then
7:      $\ell := \ell + 1$
8:    end if
9:  end for
10:  $|S(k)| := \ell$
11: end for
12: return $|S(n - 1)|$;
13: end

The Third Loop, for $u \in S(k)$
1: $m := 0$; \{ Count members of $S(k - 1)$ encountered. \}
2: reply := false;
3: for $v = 1, 2, \ldots, n$ do
4:  if $v \in S(k - 1)$ then
5:    $m := m + 1$
6:  if $G(v, u)$ then
7:    reply := true;
8:    end if
9:  end if
10: end for
11: if $m < |S(k - 1)|$ then
12:  “no”; \{ Cannot be sure of the validity of reply. \}
13: end if
14: return reply;

The Fourth Loop, for $v \in S(k - 1)$
1: $s := 2$
2: for $i = 1, 2, \ldots, k - 1$ do
3:  Guess a node $t \in \{1, 2, \ldots, n\}$; \{ Nondeterminism. \}
4:  if $(s, t) \notin G$ then
5:    “no”;
6:  end if
7:  $s := t$
8: end for
9: if $t = v$ then
10: return true;
11: else
12:  “no”;
13: end if
Wrap Up the Proof

- Space is needed for $k, |S(k - 1)|, \ell, u, m, v, s, i, t$.
- The nondeterministic algorithm needs space $O(\log n)$.

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 $B$.
- Problem $A$ is at least as hard as problem $B$ if $B$ reduces to $A$.

Closure under Complement of Nondeterministic Space

**Corollary 25** If $f \geq \log n$ is proper, then

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

- Run the above algorithm on the configuration graph of the NTM $M$ deciding $L \in \text{NSPACE}(f(n))$ on input $x$.
- We accept only if no accepting configurations have been encountered and if $|S(n - 1)|$ is computed.
  - The existence of $|S(n - 1)|$ means that every reachable configuration has been visited.

Reduction

Solving problem $B$ by calling the algorithm for problem once and without further processing its answer.
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 called a (Karp) reduction from $L_1$ to $L_2$.
- Note that by Theorem 20 (p. 169), $R$ runs in polynomial time.

Reduction of HAMILTONIAN PATH to SAT

- Given a graph $G$, we shall construct a CNF $R(G)$ such that $R(G)$ is satisfiable if and only if $G$ has a Hamiltonian path.
- Suppose $G$ has $n$ nodes: $1, 2, \ldots, n$.
- $R(G)$ has $n^2$ boolean variables $x_{ij}$, $1 \leq i, j \leq n$.
- $x_{ij}$ means “node $j$ is the $i$th node in the Hamiltonian path.”

A Paradox?

- Degree of difficulty is not defined in terms of absolute complexity.
- A language $A \in \text{TIME}(n^{99})$ may be “easier” than a language $B \in \text{TIME}(n^3)$.
- This happens when $A$ is reducible to $B$.
  - In this situation, it is necessary that $|R(x)| = \Omega(n^{33})$ or that $R$ runs in time $\Omega(n^{99})$ so that $A \notin \text{TIME}(n^k)$ for some $k < 99$.

The Clauses of $R(G)$

1. Each node $j$ must appear in the path.
   - $x_{ij} \lor x_{i+1,j} \lor \ldots \lor x_{n,j}$ for each $j$.
2. No node $j$ appears twice in the path.
   - $\neg x_{ij} \lor \neg x_{i,j}$ for all $i, j, k$ with $i \neq k$.
3. Every position $i$ on the path must be occupied.
   - $x_{i1} \lor x_{i2} \lor \ldots \lor x_{in}$ 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.
   - $\neg x_{i,j} \lor \neg x_{i+1,j}$ for all $(i, j) \notin G$ and $k = 1, 2, \ldots, n$.
The Proof

- $R(G)$ can be computed efficiently.
- Suppose $T \models R(G)$.
- Clauses of 1 and 2 imply that for each $j$, there is a unique $i$ such that $T \models x_{ij}$.
- Clauses of 3 and 4 imply that for each $i$, there is a unique $j$ such that $T \models x_{ij}$.
- So there is a permutation $\pi$ of the nodes such that $\pi(i) = j$ if and only if $T \models x_{ij}$.
- Clauses of 5 guarantees that $(\pi(1), \pi(2), \ldots, \pi(n))$ is a Hamiltonian path.

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 $G$.

The Proof (concluded)

- Conversely, suppose $G$ has a Hamiltonian path
  $$(\pi(1), \pi(2), \ldots, \pi(n)),$$
  where $\pi$ is a permutation.
- Clearly, the truth assignment
  $$T(x_{ij}) = \text{true} \text{ if and only if } \pi(i) = j$$
  satisfies all clauses of $R(G)$.

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 $g_{ij0} = \text{true}$ if and only if $i = j$ or $(i, j) \in E$. 
The Construction

- \( h_{ijk} \) is an AND gate with predecessors \( g_{i,k} \) and \( g_{k,j,k} \), where \( k = 1, 2, \ldots, n \).
- \( g_{ijk} \) is an OR gate with predecessors \( g_{i,j,k} \) and \( h_{i,j,k} \), where \( k = 1, 2, \ldots, n \).
- \( g_{nn} \) is the output gate.
- Interestingly, \( R(G) \) uses no \( \neg \) gates: It is a monotone circuit.
- The depth of \( R(G) \) is \( O(n) \), which is not optimal.

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 \( g \land h \).
  - Meaning: \( g \Leftrightarrow \neg h \).

Reduction of CIRCUIT SAT to SAT

- Given a circuit \( C \), we shall construct a boolean expression \( R(C) \) such that \( R(C) \) is satisfiable if and only if \( C \) is satisfiable.
  - \( R(C) \) will turn out to be a CNF.
- The variables of \( R(C) \) are those of \( C \) plus \( g \) for each gate \( g \) of \( C \).
- Each gate of \( C \) will be turned into equivalent clauses of \( R(C) \).
- Recall that clauses are \( \land \)ed together.

The Clauses of \( R(C) \) (concluded)

- \( g \) is a \( \lor \) gate with predecessor gates \( h \) and \( h' \): Add clauses \( \neg h \lor g \), \( \neg h' \lor g \), and \( h \lor h' \lor \neg g \).
  - Meaning: \( g \Leftrightarrow (h \lor h') \).
- \( g \) is a \( \land \) gate with predecessor gates \( h \) and \( h' \): Add clauses \( \neg g \lor h \), \( \neg g \lor h' \), and \( \neg h \lor \neg h' \lor g \).
  - Meaning: \( g \Leftrightarrow (h \land h') \).
- \( g \) is the output gate: Add clause \( g \).
  - Meaning: \( g \) must be true to make \( R(C) \) true.