Time Hierarchy Theorem

**Theorem (Hartmanis-Stearns’65).** For time constructible functions, if

\[ f(n) \log f(n) = o(g(n)) \]

then exist problems in \( \text{DTIME}(g(n)) \) but not in \( \text{DTIME}(f(n)) \).

**Theorem (Cook’72).** For time constructible functions \( f \) and \( g \), if

\[ f(n + 1) = o(g(n)) \]

then exist problems in \( \text{NTIME}(g(n)) \) but not in \( \text{NTIME}(f(n)) \).

It is still open if \( \text{NP} \not\subset \text{DEXP} \).
A decision problem $\Pi$ is undecidable if there is no algorithm that can solve $\Pi$ in finite time.

**Halting**

**Instance:** A computer program $A$ and an input $x$ for $A$.

**Question:** Does $A(x)$ halt in finite time?

**Key.** A computer program $\simeq$ a string $x$.

The answer is clearly "yes" or "no."

Can we always tell (decide) in finite time?
The Diagonalization Technique

The monster program:
\[ \forall A, \ Halt(M, A) = \neg Halt(A, A). \]

<table>
<thead>
<tr>
<th>program</th>
<th>input</th>
<th>( A_1 )</th>
<th>( A_2 )</th>
<th>( A_3 )</th>
<th>( \ldots )</th>
<th>( M )</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
<td>( \neg A_1(A_1) )</td>
<td>( \neg A_2(A_2) )</td>
<td>( \neg A_3(A_3) )</td>
<td>( \ldots )</td>
<td>( \neg M(M) )</td>
<td>( \ldots )</td>
<td></td>
</tr>
<tr>
<td>( A_2 )</td>
<td>( \neg A_1(A_1) )</td>
<td>( \neg A_2(A_2) )</td>
<td>( \ldots )</td>
<td>( \neg M(M) )</td>
<td>( \ldots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A_3 )</td>
<td>( \neg A_1(A_1) )</td>
<td>( \neg A_2(A_2) )</td>
<td>( \ldots )</td>
<td>( \neg M(M) )</td>
<td>( \ldots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M )</td>
<td>( \neg A_1(A_1) )</td>
<td>( \neg A_2(A_2) )</td>
<td>( \ldots )</td>
<td>( \neg M(M) )</td>
<td>( \ldots )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is \( Halt(M, M) \)?

Existential but hardly “practical” proof.
**Reductions**

**Halt-All**

**Instance:** A computer program $A$.

**Question:** Does $A$ halt in finite time on all inputs?

**Theorem.** Halting $\leq$ Halt-All.

**Proof.** Take an instance $(A, x)$ of Halting. Define an instance of Halt-All as the program $B$ which simulates $A(x)$. 