THE LANDSCAPE OF THE CHOMSKY HIERARCHY
A reduction is a way of converting one problem to another problem, so that the solution to the second problem can be used to solve the first problem.

- Finding the area of a rectangle, reduces to measuring its width and height
- Solving a set of linear equations, reduces to inverting a matrix.

Reducibility involves two problems $A$ and $B$.

- If $A$ reduces to $B$, you can use a solution to $B$ to solve $A$
- When $A$ is reducible to $B$ solving $A$ can not be “harder” than solving $B$.
- If $A$ is reducible to $B$ and $B$ is decidable, then $A$ is also decidable.
- If $A$ is undecidable and reducible to $B$, then $B$ is undecidable.
**Theorem 5.1**

\[ \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \} \text{ is undecidable.} \]

**Proof**

- Use the idea that “If A is undecidable and reducible to B, then B is undecidable.”
- Suppose R decides \( \text{HALT}_{TM} \). We construct S to decide \( A_{TM} \).
- \( S = \) “On input \( \langle M, w \rangle \)
  1. Run R on input \( \langle M, w \rangle \).
  2. If R rejects reject.
  3. If R accepts, simulate M on w until it halts.
  4. If M has accepted, accept; If M has rejected, reject.”
- Since \( A_{TM} \) is reduced to \( \text{HALT}_{TM} \), \( \text{HALT}_{TM} \) is undecidable.
Theorem 5.2

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \Phi \} \text{ is undecidable.} \]

- Suppose \( R \) decides \( E_{TM} \). We try to construct \( S \) to decide \( A_{TM} \) using \( R \).
  - Note that \( S \) takes \( \langle M, w \rangle \) as input.
- One idea is to run \( R \) on \( \langle M \rangle \) to check if \( M \) accepts some string or not – but that that does not tell us if \( M \) accepts \( w \).
- Instead we modify \( M \) to \( M_1 \). \( M_1 \) rejects all strings other than \( w \) but on \( w \), it does what \( M \) does.
- Now we can check if \( L(M_1) = \Phi \).
**Theorem 5.2**

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Phi\}$ is undecidable.

**Proof**

- For any $w$ define $M_1$ as
  
  $M_1 = "\text{On input } x:\n  1. \text{If } x \neq w, \text{ reject.}\n  2. \text{If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does."}\n
- Note that $M_1$ either accepts $w$ only or nothing!
Assume \( R \) decides \( E_{TM} \)

\( S \) defines below uses \( R \) to decide on \( A_{TM} \)
\( S = \text{“On input } \langle M, w \rangle \) 

1. Use \( \langle M, w \rangle \) to construct \( M_1 \) above.
2. Run \( R \) on input \( \langle M_1 \rangle \)
3. If \( R \) accepts, reject, if \( R \) rejects, accept.

So, if \( R \) decides \( M_1 \) is empty,
- then \( M \) does NOT accept \( w \),
- else \( M \) accepts \( w \).

If \( R \) decides \( E_{TM} \) then \( S \) decides \( A_{TM} \) – Contradiction.
Can we find out if a language accepted by a Turing machine $M$ is accepted by a simpler computational model?

- Is the language of a TM actually a regular language? ($\text{REGULAR}_{TM}$)
- Is the language of a TM actually a CFL? ($\text{CFL}_{TM}$)
- Does that language of a TM have an “interesting” property?
  - Rice’s Theorem.
**Testing for Regularity**

\[ \text{REGULAR}_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language} \} \] is undecidable.

**Proof Idea**

- We assume \( \text{REGULAR}_{TM} \) is decidable by a TM \( R \) and use this assumption to construct a TM \( S \) that decides \( A_{TM} \).
- The basic idea is for \( S \) to take as input \( \langle M \rangle \) and modify \( M \) into \( M_2 \) so that the resulting TM recognizes a regular language if and only if \( M \) accepts \( w \).
- \( M_2 \)
  - accepts \( \{ 0^n1^n \mid n \geq 0 \} \) if \( M \) does not accept \( w \),
  - but recognizes \( \Sigma^* \) if \( M \) accepts \( w \).
PROOF IDEA – CONTINUED

- $M_2$ accepts $\{0^n1^n \mid n \geq 0\}$ if $M$ does not accept $w$, but recognizes $\Sigma^*$ if $M$ accepts $w$.
- What does $M_2$ look like?
- $M_2 =$ “On input $x$
  1. If $x$ has the form $0^n1^n$, accept.
  2. If $x$ does not have this form, run $M$ on input $w$ and accept if $M$ accepts $w$.”
- All strings $x$ (that is $\Sigma^*$) are accepted if $M$ accepts $w$. 
Testing for Regularity

\begin{align*}
S & \rightarrow \text{Build } M_2 \\
< M_2 > & \rightarrow \text{Is } L(M_2) \text{ Regular?} \\
R & \rightarrow M \text{ accepts } w \\
& \rightarrow M \text{ rejects } w
\end{align*}

\begin{align*}
\text{Is } x = a^n b^n \text{? } M_2 & \\
\text{Yes} & \rightarrow \text{Run } M \text{ on } w \\
& \rightarrow \text{Accept} \\
\text{No} & \rightarrow \text{Reject}
\end{align*}

So \( L(M_2) = \Sigma^* \) if \( M \) accepts \( w \)

\( L(M_2) = \{ a^n b^n \} \) otherwise
TESTING FOR REGULARITY

PROOF

- \( S = \text{“On input } \langle M, w \rangle \text{, where } M \text{ is a TM and } w \text{ is a string:} \)
- 1. Construct the following TM \( M_2 \).
- 2. \( M_2 = \text{“On input } x \text{”} \)
  1. If \( x \) has the form \( 0^n1^n \), accept.
  2. If \( x \) does not have this form, run \( M \) on input \( w \) and accept if \( M \) accepts \( w \).”
- 3. Run \( R \) on \( \langle M_2 \rangle \)
- 4. If \( R \) accepts, accept, if \( R \) rejects, reject.

- So, \( R \) will say \( M_2 \) is a regular language, if \( M \) accepts \( w \).
- \( S \) says “\( M \) accepts \( w \)” if \( R \) decides \( M_2 \) is regular – Contradiction!
Testing for Language Equality

**Theorem 5.4**

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \] is undecidable.

**Proof Idea**

- We reduce \( E_{TM} \) (the emptiness problem) to this problem.
- If one of the languages is empty, determining equality is the same as determining if the second language is empty!
- In fact, the \( E_{TM} \) is a special case of the \( EQ_{TM} \) problem!!
**Theorem 5.4**

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \text{ is undecidable.} \]

**Proof**

- Assume \( R \) decides \( EQ_{TM} \)
- \( S = \) “On input \( \langle M \rangle \) where \( M \) is a TM:
  1. Run \( R \) on input \( \langle M, M_1 \rangle \) where \( M_1 \) is a TM that rejects all inputs.
  2. If \( R \) accepts, accept; if \( R \) rejects reject”

- Thus, if \( R \) decides \( EQ_{TM} \), then \( S \) decides \( E_{TM} \)
- But \( E_{TM} \) is undecidable, so \( EQ_{TM} \), must be undecidable.
An accepting computation history for a TM is a sequence of configurations

\[ C_1, C_2, \ldots, C_l \]

such that
- \( C_1 \) is the start configuration for input \( w \)
- \( C_l \) is an accepting configuration, and
- each \( C_i \) follows legally from the preceding configuration.

A rejecting computation history is defined similarly.

Computation histories are finite sequences – if \( M \) does not halt on \( w \), there is no computation history.

Deterministic v.s nondeterministic computation histories.
Suppose we cripple a TM so that the head never moves outside the boundaries of the input string. Such a TM is called a linear bounded automaton (LBA). Despite their memory limitation, LBAs are quite powerful.

**Lemma**

Let $M$ be a LBA with $q$ states, $g$ symbols in the tape alphabet. There are exactly $qng^n$ distinct configurations for a tape of length $n$.

**Proof.**

- The machine can be in one of $q$ states.
- The head can be on one of the $n$ cells.
- At most $g^n$ distinct strings can occur on the tape.
**Decidability of LBA Problems**

**Theorem 5.9**

\[ A_{LBA} = \{ \langle M, w \rangle \mid M \text{ is an LBA that accepts string } w \} \text{ is decidable.} \]

**Proof Idea**

- We simulate LBA \( M \) on \( w \) with a TM \( L \) (which is NOT an LBA!)
- If during simulation \( M \) accepts or rejects, we accept or reject accordingly.
- What happens if the LBA \( M \) loops?
  - Can we detect if it loops?
- \( M \) has a finite number of configurations.
  - If it repeats any configuration during simulation, it is in a loop.
  - If \( M \) is in a loop, we will know this after a finite number of steps.
  - So if the LBA \( M \) has not halted by then, it is looping.
**Theorem 5.9**

\[ A_{LBA} = \{ \langle M, w \rangle \mid M \text{ is an LBA that accepts string } w \} \] is decidable.

**Proof**

- The following TM decides \( A_{LBA} \).
  - \( L = "\text{On input } \langle M, w \rangle" \)
    - 1. Simulate \( M \) on for \( qng^n \) steps or until it halts.
    - 2. If \( M \) has halted, *accept* if it has accepted, and *reject* if it has rejected. If it has NOT halted, *reject*.

LBAs and TMs differ in one important way. \( A_{LBA} \) is decidable.
Now for a really wild and crazy idea!
Consider an accepting computation history of a TM $M$, $C_1, C_2, \ldots, C_l$
Note that each $C_i$ is a string.
Consider the string

$$\# C_1 \# C_2 \# C_3 \# \cdots \# C_l \#$$

The set of all valid accepting histories is also a language!!
This string has length $m$ and an LBA $B$ can check if this is a valid computation history for a TM $M$ accepting $w$.
Check if $C_1 = q_0 w_1 w_2 \cdots w_n$
Check if $C_i = \cdots q_{\text{accept}} \cdots$
Check if each $C_{i+1}$ follows from $C_i$ legally.
Note that $B$ is not constructed for the purpose of running it on any input!
If $L(B) \neq \emptyset$ then $M$ accepts $w$
Decidability of LBA Problems

**Theorem 5.10**

\[ E_{LBA} = \{ \langle M \rangle \mid M \text{ is an LBA and } L(M) = \Phi \} \text{ is undecidable.} \]

**Proof.**

1. Suppose TM \( R \) decides \( E_{LBA} \), we can construct a TM \( S \) which decides \( A_{TM} \).
2. \( S = “ \text{On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string} \)
   1. Construct LBA \( B \) from \( M \) and \( w \) as described earlier.
   2. Run \( R \) on \( \langle B \rangle \).
   3. If \( R \) rejects, accept; if \( R \) accepts, reject.”
3. So if \( R \) says \( L(B) = \Phi \), the \( M \) does NOT accept \( w \).
4. If \( R \) says \( L(B) \neq \Phi \), the \( M \) accepts \( w \).
5. But, \( A_{TM} \) is undecidable – contradiction.