Computation Histories

- The computation history for a Turing machine on an input is simply the sequence of configurations that the machine goes through as it processes the input.
- An accepting computation history for $M$ on $w$ is a sequence of configurations $C_1, C_2, ..., C_l$, where
  - $C_1$ is the start configuration $q_0w$,
  - $C_l$ an accepting configuration of $M$, and
  - each $C_i$ legally follows from $C_{i-1}$ according to the rules of $M$

- Similarly one can define a rejecting computation history.
- Computation histories are finite sequences — if $M$ doesn’t halt on $w$, no accepting or rejecting computation history exists for $M$ on $w$.

Linear Bounded Automata

- A linear bounded automaton (LBA) is a Turing machine that cannot use extra working space.
- It can only use the space taken up by the input.
- Because the tape alphabet can, in any case, be larger than the input alphabet, it allows the available memory to be increased up to a constant factor.
- Deciders for problems concerning context-free languages.
- If a LBA has
  - $q$ states
  - $g$ symbols in its tape alphabet, and
  - an input of length $n$,
  then the number of its possible configurations is $q \cdot n \cdot g^n$. 
Theorem 5.9
The acceptance problem for linear bounded automata
\[ A_{LBA} = \{ \langle M, w \rangle \mid M \text{ is an LBA that accepts string } w \} \]
is decidable.

Proof. As \( M \) computes on \( w \), it goes from configuration to configuration. If it ever repeats a configuration, it will go on to repeat this configuration over and over again and thus be in a loop.

Because an LBA has only \( q \cdot n \cdot g \) distinct configurations, if the computation of \( M \) has not halted in so many steps, it must be in a loop. Thus, to decide \( A_{LBA} \) it is enough to simulate \( M \) on \( w \) for \( q \cdot n \cdot g \) steps or until it halts. \( \square \)

Theorem 5.10
The emptiness problem for linear bounded automata
\[ E_{LBA} = \{ \langle M \rangle \mid M \text{ is an LBA and } L(M) = \emptyset \} \]
is undecidable.

Proof. Reduction from the universal language (acceptance problem for general TMs).

Counter-assumption: \( E_{LBA} \) is decidable; i.e., there exists a decider \( M_{LBA} \) for \( E_{LBA} \).

Let \( M \) be an arbitrary Turing machine, whose operation on input \( w \) is under scrutiny. Let us compose an LBA \( B \) that recognizes all accepting computation histories for \( M \) on \( w \).
Now we can reduce the acceptance problem for general Turing machines to the emptiness testing for LBAs:

\[
L(B) \neq \emptyset \quad \text{if } w \in L(M) \\
L(B) = \emptyset \quad \text{if } w \not\in L(M)
\]

The LBA $B$ must accept input string $x$ if it is an accepting computation history for $M$ on $w$.

Let the input be presented as $x = C_1 \# C_2 \# \cdots \# C_l$.

$B$ checks that $x$ satisfies the conditions of an accepting computation history:

- $C_1 = q_0 \cdot w$,
- $C_i$ is an accepting configuration for $M$; i.e. accept is the state in $C_i$, and
- $C_{i-1} \Rightarrow_M C_i$:
  - configurations $C_{i-1}$ and $C_i$ are identical except for the position under and adjacent to the tape head in $C_{i-1}$, and
  - the changes correspond to the transition function of $M$.

Given $M$ and $w$ it is possible to construct LBA $B$ mechanically.
By combining machines $B$ and $M_t^T$ as shown in the following figure, we obtain a decider for the acceptance problem of general Turing machines (universal language).

\[
\begin{align*}
\langle M, w \rangle &\in L(M_t^T) \\
\iff &\langle B \rangle \notin L(M_t^T) \\
\iff &L(B) \neq \emptyset \\
\iff &w \in L(M) \\
\implies &\langle M, w \rangle \in U
\end{align*}
\]

This is a contradiction, and the language $E_{LBA}$ cannot be decidable.
5.3 Mapping Reducibility

- Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \text{accept}, \text{reject})$ be an arbitrary standard Turing machine.
- Let us define the partial function $f_M: \Sigma^* \rightarrow \Gamma^*$ computed by the TM as follows:

$$f_M(w) = \begin{cases} u, & \text{if } q_0 \xrightarrow{w} Q \text{ and } q \in \{\text{accept, reject}\} \\ \text{undefined}, & \text{otherwise} \end{cases}$$

- Thus, the TM $M$ maps a string $w \in \Sigma^*$ to the string $u$, which is the contents of the tape, if the computation halts on $w$.
- If it does not halt, the value of the function is not defined in $w$.

Definition 5.20

- Partial function $f$ is computable, if it can be computed with a total Turing machine. I.e. if its value $f(w)$ is defined for every $w$.

- Let us formulate the idea that problem $A$ is "at most as difficult as" problem $B$ as follows:

- Let $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$ be two formal languages.
- $A$ is mapping reducible to $B$, written $A \leq_m B$, if there is a computable function $f: \Sigma^* \rightarrow \Gamma^*$ s.t. $w \in A \Leftrightarrow f(w) \in B$ for all $w \in \Sigma^*$.
- The function $f$ is called the reduction of $A$ to $B$.
Mapping an instance $w$ of $A$ computably into an instance $f(w)$ of $B$ and

"does $w$ have property $A$?" $\iff$ "does $f(w)$ have property $B$?"

Lemma J  For all languages $A$, $B$, $C$ the following hold

1. $A_s m A$, (reflexive)
2. if $A_s m B$ and $B_s m C$, then $A_s m C$, (transitive)
3. if $A_s m B$ and $B$ is Turing-recognizable, then so is $A$, and
4. if $A_s m B$ and $B$ is decidable, then so is $A$

Proof.

i. Let us choose $f(x) = x$ as the reduction.

ii. Let $f$ be reduction of $A$ to $B$ and $g$ a reduction of $B$ to $C$.

In other words, $f: A_s m B$, $g: B_s m C$.

We show that the composite function $h$, $h(x) = g(f(x))$ is a reduction $h: A_s m C$. 
1. \( h \) is computable: Let \( M_f \) and \( M_g \) be the total Turing machines computing \( f \) and \( g \). \( M_{REW} \) replaces all symbols to the right of the tape head with \( \square \) and moves the tape head to the beginning of the tape. The total machine depicted in the following figure computes function \( h \).

2. \( h \) is a reduction:

\[
x \in A \Leftrightarrow f(x) \in B \\
\Leftrightarrow g(f(x)) = h(x) \in C,
\]

hence, \( h : A \leq_m C \).
iii. (and iv.) Let $f : A \leq_m B$, $M_B$ the recognizer of $B$ and $M_f$ the TM computing $f$. The TM depicted below recognizes language $A$ and it is total whenever $M_B$ is.

![Diagram of TM recognizing A]

We have already used the following consequence of Lemma J to prove undecidability.

**Corollary 5.23** If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

Let us call language $B \subseteq \{0, 1\}^*$ **TR-complete**, if

1. $B$ is Turing-recognizable (TR), and
2. $A \leq_m B$ for all Turing-recognizable languages $A$
**Theorem K** The universal language $U$ is TR-complete.

**Proof.** We know that $U$ is Turing-recognizable. Let $B$ be any Turing-recognizable language. Furthermore, let $B = L(M_B)$.

Now, $B$ can be reduced to $U$ with the function $f(x) = \langle M_B, x \rangle$, which is clearly computable, and for which it holds

$$x \in B \iff L(M_B) \land f(x) = \langle M_B, x \rangle \in U.$$  

\[\square\]

**Theorem L** Let $A$ be a TR-complete language, $B$ TR, and $A \leq_m B$. Then also $B$ is a TR-complete language.

- All "natural" languages belonging to the difference of TR and decidable languages are TR-complete, but it contains also other languages

- The class of TR languages is not closed under complementation, thus it has the dual class

  $$\text{co-TR} = \{ \overline{A} \mid A \in \text{TR} \}$$

- $\text{TR} \cap \text{co-TR} = \text{decidable languages}$ (by Theorem 4.22)

- $B \subseteq \{0, 1\}^*$ is co-TR-complete, if

  1. $B \in \text{co-TR}$ and
  2. $A \leq_m B$ for all $A \in \text{co-TR}$

- A language $A$ is co-TR-complete, if and only if the language $\overline{A}$ is TR-complete

- Language $\text{TOT} = \{ \langle M \rangle \mid M \text{ halts on all inputs} \}$ does not belong to either TR or co-TR
TR-complete

\(-U\)

\(-NE\)

\(-TOT\)

TR

decidable

co-TR

co-TR-complete