Counter Machines

A counter machine is a machine consisting of a read-only input tape, a finite-state control, and a fixed number of counters, each capable of storing an arbitrary natural number.

The following operations can be performed on the counters:

- increment by 1;
- decrement by 1 if nonzero; and
- comparison with 0.

**Theorem 8.14:** Every RE language is accepted by a 3-counter machine.

**Proof sketch:** Let $L$ be an RE language, and let $M$ be a 2-stack machine such that $L(M) = L$.

- Suppose the stack alphabet $\Gamma$ of $M$ contains $r - 1$ symbols.
- We can define $f : \Gamma \xrightarrow{1} \{1, \ldots, r - 1\}$. 


• We define \( g : \Gamma^* \rightarrow \mathbb{N} \) such that
\[
g(\varepsilon) = 0 \\
g(X\alpha) = f(X) + rg(\alpha) \quad \text{for } X \in \Gamma
\]

• It is easily seen by induction on \( \alpha \in \Gamma^* \) that if \( g(\alpha) = g(\beta) \), then \( \alpha = \beta \).

• We can store the contents \( \alpha \) of a stack in a counter as \( g(\alpha) \).

• The third counter is used as temporary storage.

To examine the top symbol of a stack, we compute
\[
g(X\alpha) \mod r = f(X)
\]
as follows:

• Use finite control to implement a mod \( r \) counter \( c \).

• While decrementing the counter storing \( g(X\alpha) \), increment the temporary counter and \( c \).
To remove the top symbol of a stack, we compute

\[[g(X\alpha)/r] = g(\alpha).\]

To push a symbol \(X\), we compute

\[rg(\alpha) + f(X) = g(X\alpha)\]

---

**Theorem 8.15:** Every RE language is accepted by a 2-counter machine.

**Proof sketch:** Let \(L\) be an RE language, and let \(M\) be a 3-counter machine such that \(L(M) = L\).

- We define \(f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}\) such that \(f(i, j, k) = 2^i3^j5^k\).
- Because 2, 3, and 5 are prime, and because every positive integer has a unique prime factorization, \(f\) is one-to-one.
- We encode the three counter values \(i, j, \text{ and } k\) as \(f(i, j, k)\).
• We can increment/decrement these values by multiplying/dividing by 2, 3, or 5, using the other counter as temporary storage.

• We can compare any of these values with 0 by computing \( f(i, j, k) \mod 2, 3, \) or 5.

**Defn:** Given a string \( w \in \{0, 1\}^* \), let \( M(w) \) denote the TM with input alphabet \( \{0, 1\} \) that \( w \) encodes in some fixed TM encoding scheme. We can define a function \( f : \{0, 1\}^* \to 2^{\{0,1\}^*} \) such that

\[
f(w) = L(M(w)).
\]

Note that the range of \( f \) is the set of RE languages over \( \{0, 1\} \).
**Cantor’s Theorem:** Let $A$ be a set, and let $f : A \to 2^A$. Then

$$B = \{ x \in A \mid x \not\in f(x) \} \not\in \text{ran}(f)$$

**Proof:** By contradiction.

- Assume that for some $a \in A$, $f(a) = B$.
- Then $a \in B$ iff $a \not\in B$ — a contradiction.

**Theorem 9.2:** Let

$L_d = \{ w \in \{0,1\}^* \mid w \not\in L(M(w)) \}$. Then $L_d$ is not RE.