Coding Turing Machines

Turing Machines:
A turning machine has a finite set of states with one START state and some (may be none) HALT states. We always mark the START state with 1 and the HALT state with 2, (when there is only one HALT state). There are transitions between states, each of which is marked by a triplet:

(Read_SYMBOL, Write_SYMBOL, Move_DIRECTION)

(a, a, R)

Each Turing machine has an infinite Tape divided into a sequence of cells each containing a symbol or a blank. The input is presented to the machine one symbol per cell beginning the leftmost cell after a marker (#).

<table>
<thead>
<tr>
<th>#</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>Δ</th>
<th>Δ</th>
<th>Δ</th>
<th>Δ</th>
<th>Δ</th>
<th>Δ</th>
</tr>
</thead>
</table>

Figure 1: A Turing Machine for \{ aa*b \}
The input is processed by the transitions as the Turing machine goes from state to state starting from the start state. The Turing Machine given in Figure 1 accepts the language: \{ aa*b \}

Encoding of Turing Machines:
Any Turing Machine (TM) can be represented in a table and then the table can be encoded into a string of a’s and b’s. The above TM is represented in the table below:

<table>
<thead>
<tr>
<th>FROM</th>
<th>TO</th>
<th>READ</th>
<th>WRITE</th>
<th>MOVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>a</td>
<td>a</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>a</td>
<td>a</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>b</td>
<td>b</td>
<td>R</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Δ</td>
<td>Δ</td>
<td>R</td>
</tr>
</tbody>
</table>

Since the START state is 1 and the HALT state is 2 all the information for operating the TM is available in the table. Any row of the table can be coded into a string of a’s and b’s.
Consider the general row:

\[
\begin{array}{cccccc}
\text{FROM} & \text{TO} & \text{READ} & \text{WRITE} & \text{MOVE} \\
X_1 & X_2 & X_3 & X_4 & X_5
\end{array}
\]

Where \(X_1\) and \(X_2\) are numbers and \(X_3\) and \(X_4\) are characters from \{a, b, #, \Delta \} and \(X_5\) is a direction (either L or R). A separator, \(b\), is used between \(X_1\) and \(X_2\), and between \(X_2\) and \(X_3\). The concatenated sequence of \(X_1\) and \(X_2\) with the separator has the form:

\[a^{X_1}ba^{X_2}b\]

which means a string of \(a\)'s of length \(X_1\) concatenated to a \(b\) concatenated to a string of \(a\)'s \(X_2\) long. Concatenated to a \(b\). \(X_3\) and \(X_4\) are encoded by the following table:

\[
\begin{array}{c|c}
X_3/ X_4 & \text{CODE} \\
\hline
a & aa \\
b & ab \\
\Delta & ba \\
# & bb \\
\end{array}
\]

Next, \(X_5\) is encoded as follows:

\[
\begin{array}{c|c}
X_5 & \text{CODE} \\
\hline
L & a \\
R & b \\
\end{array}
\]

For the Turing Machine of \{ aa*b \} the code for each row is given in the following table.

\[
\begin{array}{cccccccc}
\text{From} & \text{Sep.} & \text{To} & \text{Sep.} & \text{Read} & \text{Write} & \text{Move} & \text{Code for each Row} \\
1 & 3 & a & a & R & ababaaba
\end{array}
\]

\[
\begin{array}{cccccccc}
3 & 3 & a & a & a & R & aaabaabaa
\end{array}
\]

\[
\begin{array}{cccccccc}
3 & 4 & b & b & R & aaabaaaababa
\end{array}
\]

\[
\begin{array}{cccccccc}
4 & 2 & \Delta & \Delta & R & aaaaabaababab
\end{array}
\]
The code for the Turing Machine in Code Word Language (CWL) is the concatenation of the four encoded rows:

\[
\text{abaabaaaabaaabaaabaaaabaaabaaaabababaaaabababababababab}
\]

The Code Word Language (CWL) is characterized with following pattern:

\[
\text{CWL} = \text{the language defined by } (aa^*baa^*b(a+b)^5)^* \\
\]

Informally, \( aa^* = a^+ \) and, therefore, \( \text{CWL} = \{ (a^*ba^*b(a+b)^5)^* \} \)

Some TMs accept their own code; others do not accept their own code. The TM of Figure 1 does not accept its own code. The TM of Figure 2 accepts its own code.

![Diagram of a Turing Machine](image)

Figure 2: A Turing Machine for \((a+b)^*\)

Please note that there are strings in CWL, that do not represent any TM. Thus, the string, \(\text{aabaabaaaab}\) is a valid string in CWL but it does not represent any TM. Now, we define the language ALAN as follows.

\[
\text{ALAN} = \{ \text{all the words in } \text{CWL} \text{ that are not accepted by the TMs they represent or that do not represent any TM} \}
\]

It can be proven that ALAN is not computable. The language ALAN is a strange one, so strange that it is not even recursively enumerable. We will prove this by contradiction. Assume that ALAN is r.e. Then there is a Turing machine that accepts ALAN. Call this Turing machine T. Now T can be encoded in CWL just as any Turing machine can be encoded. Call its encoding \(\text{code}(T)\). Consider the question, "Is \(\text{code}(T)\) in ALAN?" We shall consider the two possible answers separately.

**Case 1:** \(\text{code}(T)\) is in ALAN

Since T accepts words that are in ALAN, T accepts \(\text{code}(T)\). But ALAN contains no code word that is accepted by the machine it encodes, so this is a contradiction.

**Case 2:** \(\text{code}(T)\) is not in ALAN

Then T does not accept \(\text{code}(T)\). But then \(\text{code}(T)\) is in ALAN because any encoding that is not accepted by the machine it encodes is in ALAN. Again, this is a contradiction.
Since both cases lead to contradictions, ALAN is not a recursively enumerable language. This example proves the following theorem.

**CASE 1: code(T) is in ALAN**

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. T accepts ALAN</td>
<td>1. Definition of T</td>
</tr>
<tr>
<td>3. Code(T) is in ALAN</td>
<td>3. Hypothesis</td>
</tr>
<tr>
<td>4. T accepts code(T)</td>
<td>4. From 1 and 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>2. ALAN contains no code word that is accepted by the machine it represents.</td>
<td>2. Definition of ALAN</td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4. T accepts code(T)</td>
<td>4. From 1 and 3</td>
</tr>
<tr>
<td>5. Code(T) is not in ALAN</td>
<td>5. From 2 and 4</td>
</tr>
<tr>
<td>6. Contradiction</td>
<td>6. From 3 and 5</td>
</tr>
</tbody>
</table>

**CASE 2: code(T) is NOT in ALAN**

<table>
<thead>
<tr>
<th>CLAIM</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. T accepts ALAN</td>
<td>1. Definition of T</td>
</tr>
<tr>
<td>2. If a word is not accepted by the machine it represents, it is in ALAN.</td>
<td>2. Definition of ALAN</td>
</tr>
<tr>
<td>3. Code(T) is NOT in ALAN</td>
<td>3. Hypothesis</td>
</tr>
<tr>
<td>4. code(T) is not accepted by T</td>
<td>4. From 1 and 3</td>
</tr>
<tr>
<td>5. Code(T) is in ALAN</td>
<td>5. From 2 and 4</td>
</tr>
<tr>
<td>6. Contradiction</td>
<td>6. From 3 and 5</td>
</tr>
</tbody>
</table>

The contradiction proves that ALAN is not computable.