Theory of Automata
Post Machine

Hafiz Tayyeb Javed

Week 14

Lecture 02

Contents

- Post Machine
- Turing Machine
- Examples
- Simulating PM on TM
- Simulating TM on PM
  - SHIFT-RIGHT CYCLICALLY.

Example

- The following language cannot be shown to be non-context-free by Theorem 34:

$$L = \{a^n b^m a^n b^m\}$$

  where n and m are integers $1, 2, 3, \ldots$, and n does not necessarily equal to m.

- However, we can use Theorem 35 to show that this language is non-context-free.
- Can you do it?

Check for following CFGs

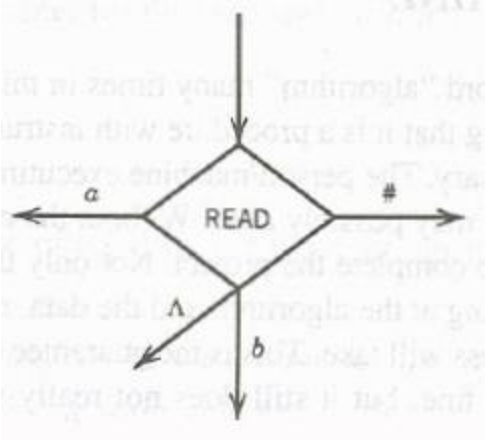
- CFL
  - $a^n b^n$
  - $a^n b^n c^m$
- Non-CFL
  - $a^n b^n a^n b^n$
  - $a^n b^{2n} a^n$
  - $a^n b^n c^n d^n$

POST Machine

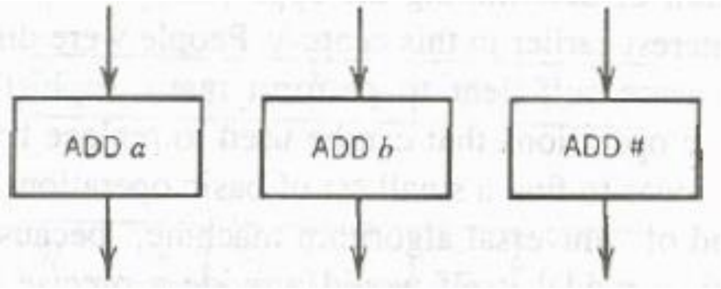
- **A Post machine** denoted by **PM** is a collection of five things
  1. The alphabet Σ plus the special symbol #. We generally use Σ = { a, b}.
  2. **A** linear storage location (a place where a string of symbols is kept called the **STORE or QUEUE,** which initially contains the input string. This location can be read by which we mean the leftmost character can be removed for inspection. The STORE can also be added to, which means a new character can be concatenated onto the right of whatever is there already. We allow for the possibility that characters not in Σ can be used in the **STORE,** characters from an alphabet Γ called the store alphabet.

POST Machine Contd.

3. READ states, for example,

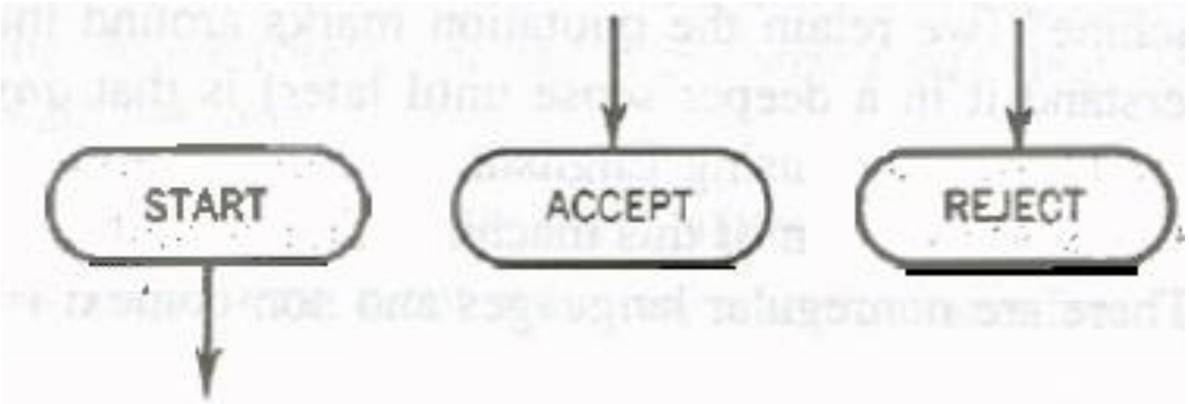


4. ADD states:



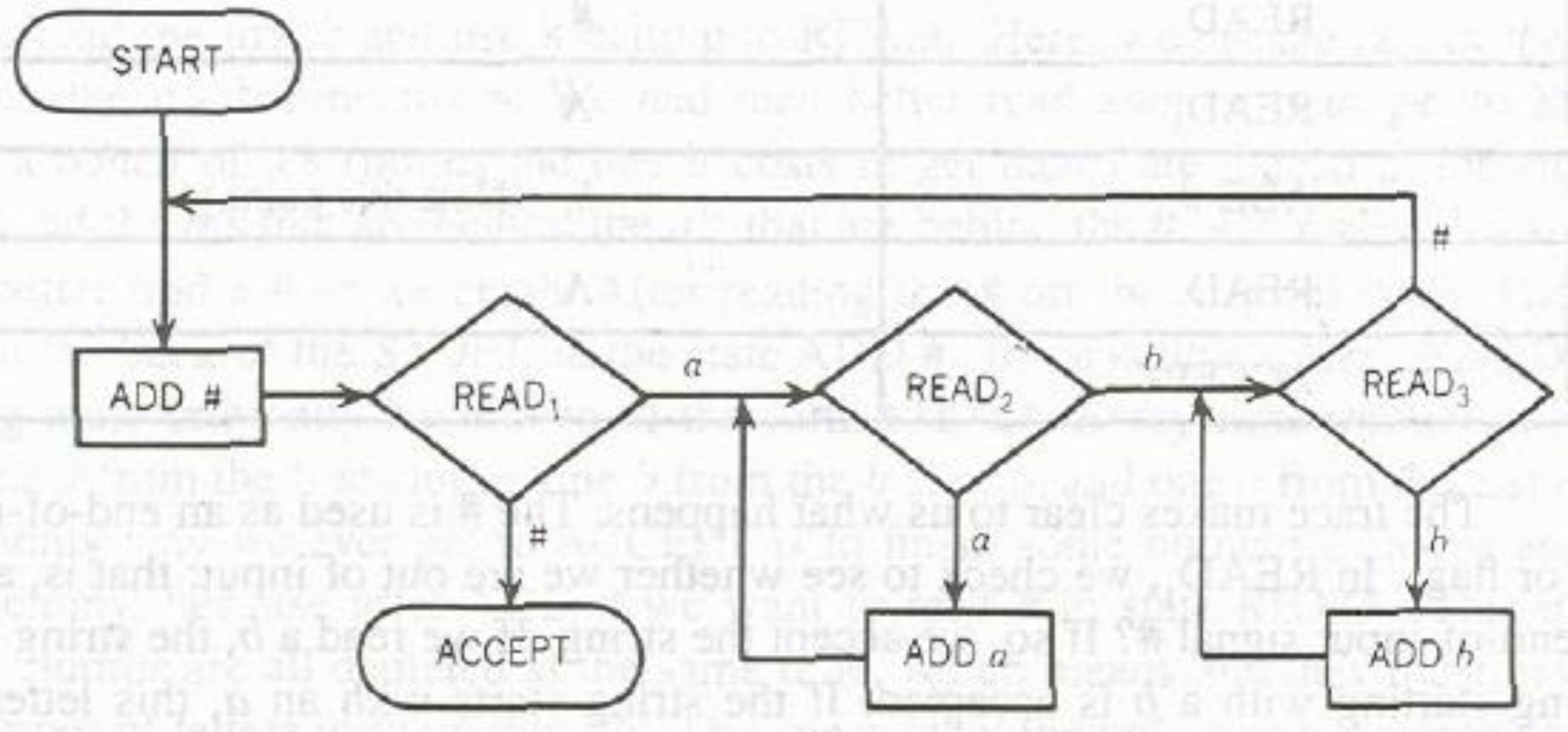
POST Machine Contd.

5. **A START** state (un-enterable) and some halt states called **ACCEPT and REJECT**

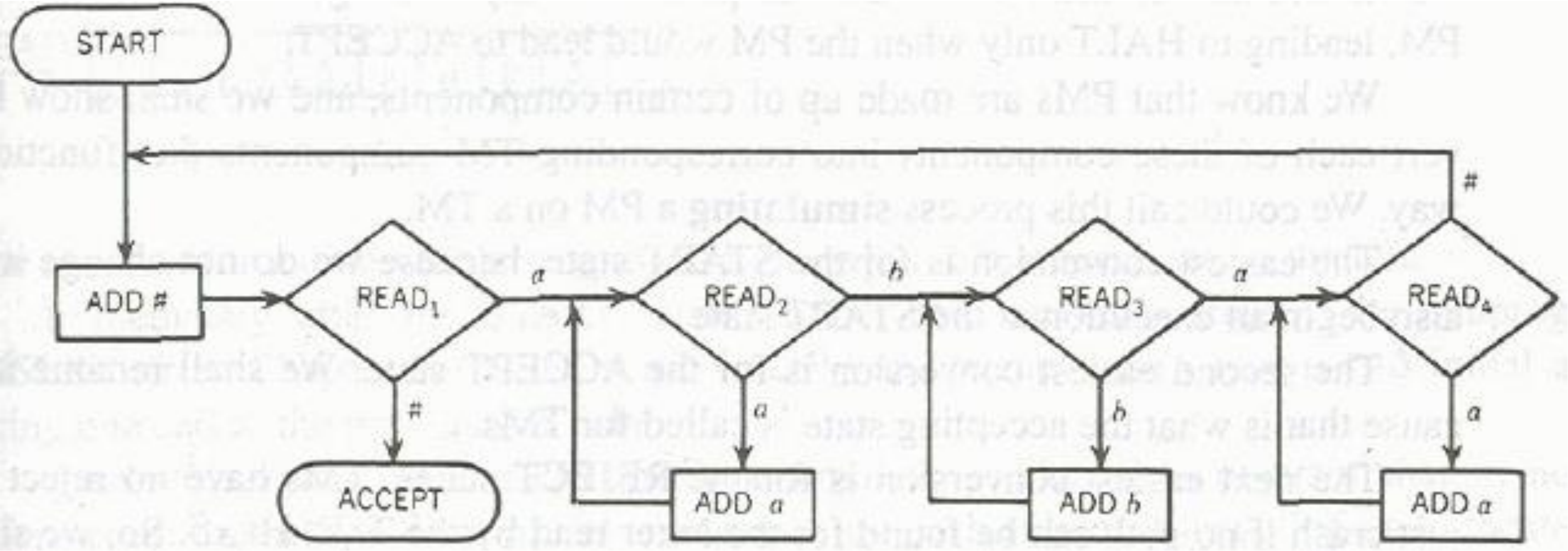


Example

- Consider the following PM and guess the language?



Another POST Machine



1. Add # in the start
2. Read back and add to front (last added to head)
3. Add $ to the end.
4. Read until you hit the #. Now what you read remember. Keep adding the rest until you hit $.
5. Now add what you saved.

Turing Machine

- Definition
- Example

Turing machine

The mathematical models (FAs, TGs, PDAs) that have been discussed so far can decide whether a string is accepted or not by them *i.e*. these models are language identifiers. However, there are still some languages which can’t be accepted by them *e.g.* there does not exist any FA or TG or PDA accepting any non-CFLs.

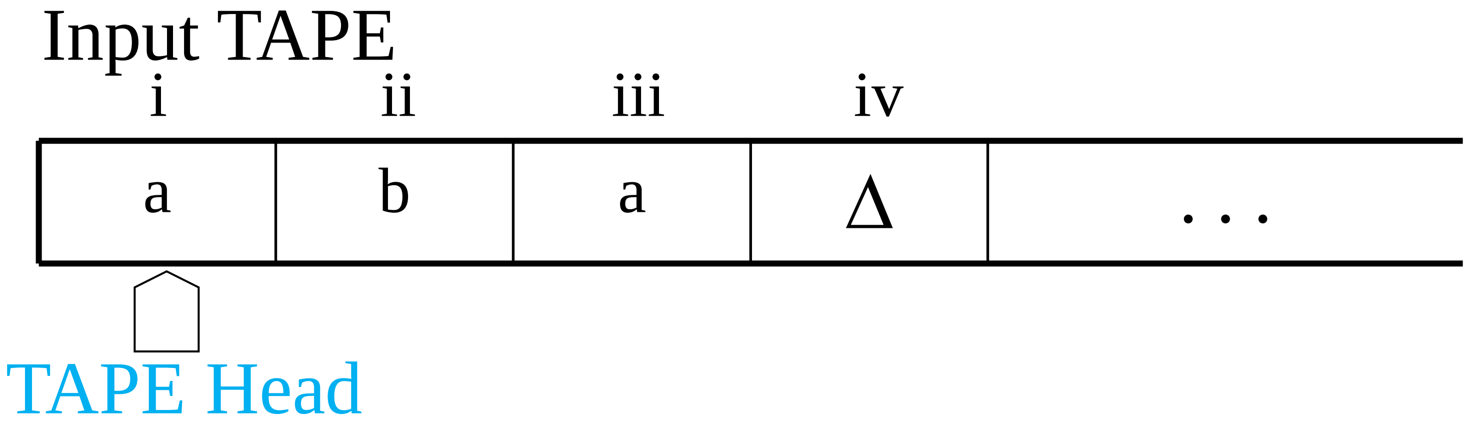
Alan Mathison Turing developed the machines called Turing machines, which accept some **non-CFLs** as well, in addition to **CFLs.**

Turing machine

Definition: A Turing machine (TM) consists of the following

1. An alphabet Σ of input letters.
2. An input TAPE partitioned into cells, having infinite many locations in one direction. The input string is placed on the TAPE starting its first letter on the cell i, the rest of the TAPE is initially filled with blanks (Δ’s).

Turing machine continued …



Input TAPE

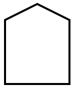
i	ii	iii	iv	
a	b	a	Δ	. . .

TAPE Head

3. A tape Head that can read the contents of cell on the TAPE in one step and it can replace the character at the cell and can reposition itself to the next cell to the right or to the left of that it has just read.

Turing machine continued …

Initially the TAPE Head is at the cell i. The TAPE Head can’t move to the left of cell i. the location of the TAPE Head is denoted by



.

4. An alphabet Γ of characters that can be printed on the TAPE by the TAPE Head. Γ may include the letters of Σ. Even the TAPE Head can print blank Δ, which means to erase some character from the TAPE.

Turing machine continued …

5. Finite set of states containing exactly one START state and some (may be none) HALT states that cause execution to terminate when the HALT states are entered.
6. A **program** which is the set of rules, which show that which state is to be entered when a letter is read form the TAPE and what character is to be printed. This program is shown by the states connected by directed edges labeled by triplet

   (letter, letter, direction)

Turing machine continued …

It may be noted that in the triplet on any edge *(a, b, R)* the first letter, *a* , is the character the TAPE Head reads from the cell to which it is pointing. The second letter, *b* , is what the TAPE Head prints the cell before it leaves. The third letter, *R* , signifies the direction the TAPE Head whether to move one cell to the right, R, or one cell to the left, L. Following is a note

Note

It may be noted that there may not be any outgoing edge at certain state for certain letter to be read from the TAPE, which creates non-determinism in Turing machines. It may also be noted that at certain state, there can’t be more than one out-going edges for certain letter to be read from the TAPE. The machine crashes if there is not a path for a letter to be read from the TAPE and the corresponding string is supposed to be rejected.

Note continued …

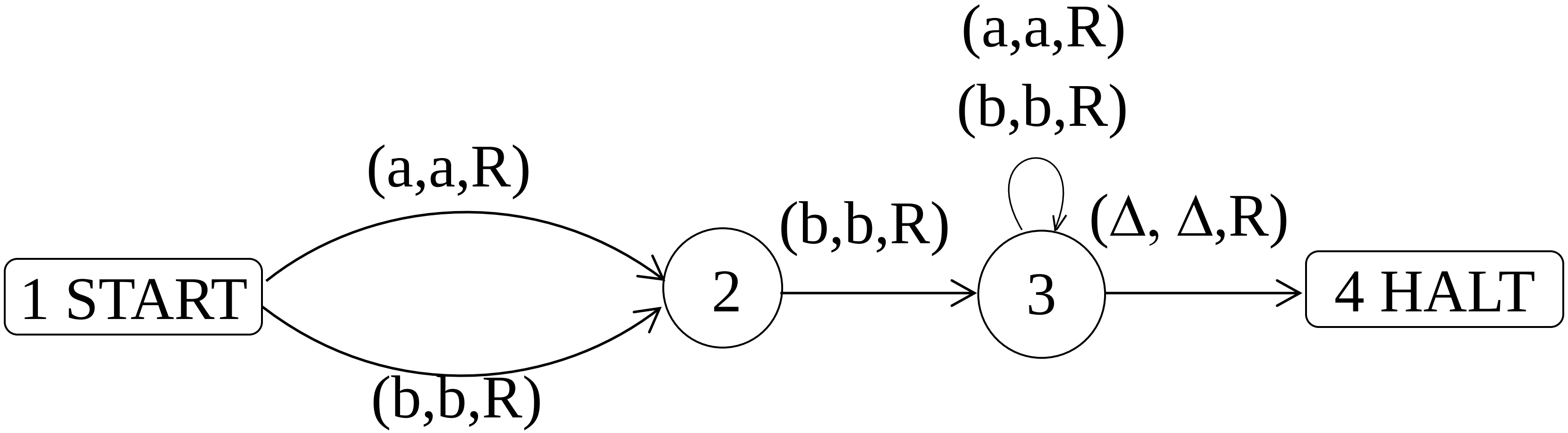
To terminate execution of certain input string successfully, a HALT state must be entered and the corresponding string is supposed to be accepted by the TM. The machine also crashes when the TAPE Head is instructed to move one cell to the left of cell i.

Following is an example of TM

Let the input string *aba, abb, bbb, bba* be run over this TM

Example

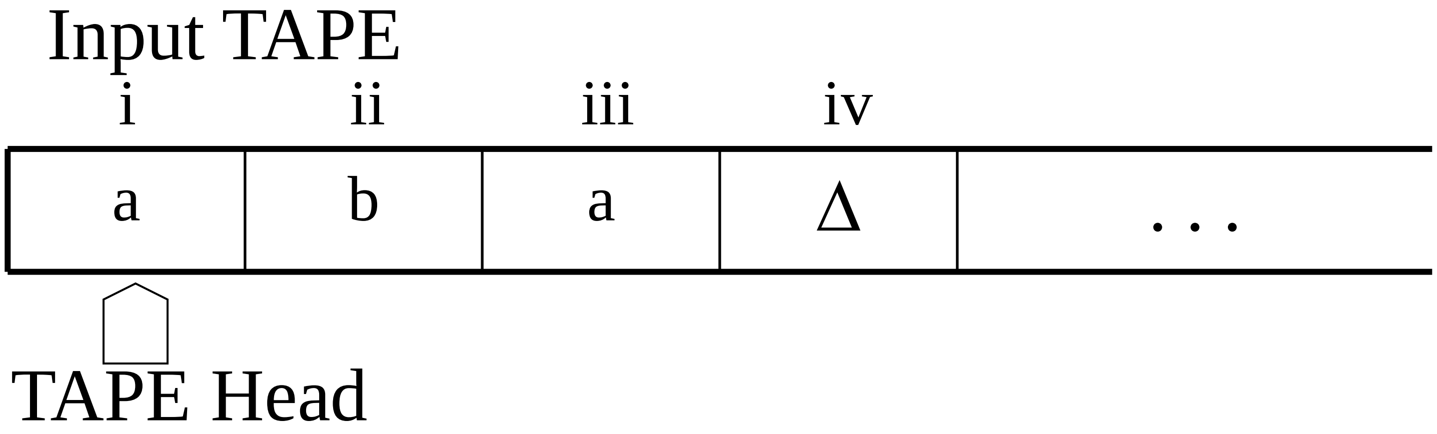
Consider the following Turing machine



Let the input string *aba* be run over this TM

(a+b)b(a+b)*.

Example continued …



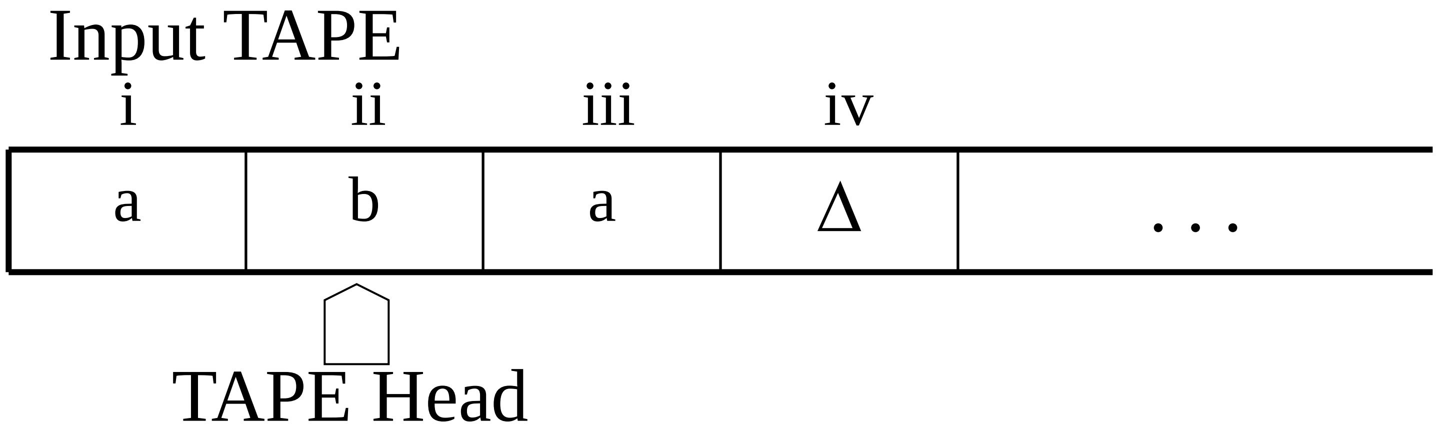
Input TAPE

i	ii	iii	iv	
a	b	a	Δ	. . .

TAPE Head

Starting from the START state, reading a form the TAPE and according to the TM program, a will be printed *i.e.* a will be replaced by a and the TAPE Head will be moved one cell to the right.

Which can be seen as



This process can be expressed as

$$\begin{array}{ccc} 1 & & 2 \\ \underline{a}ba & ⍰ & a\underline{b}a \end{array}$$

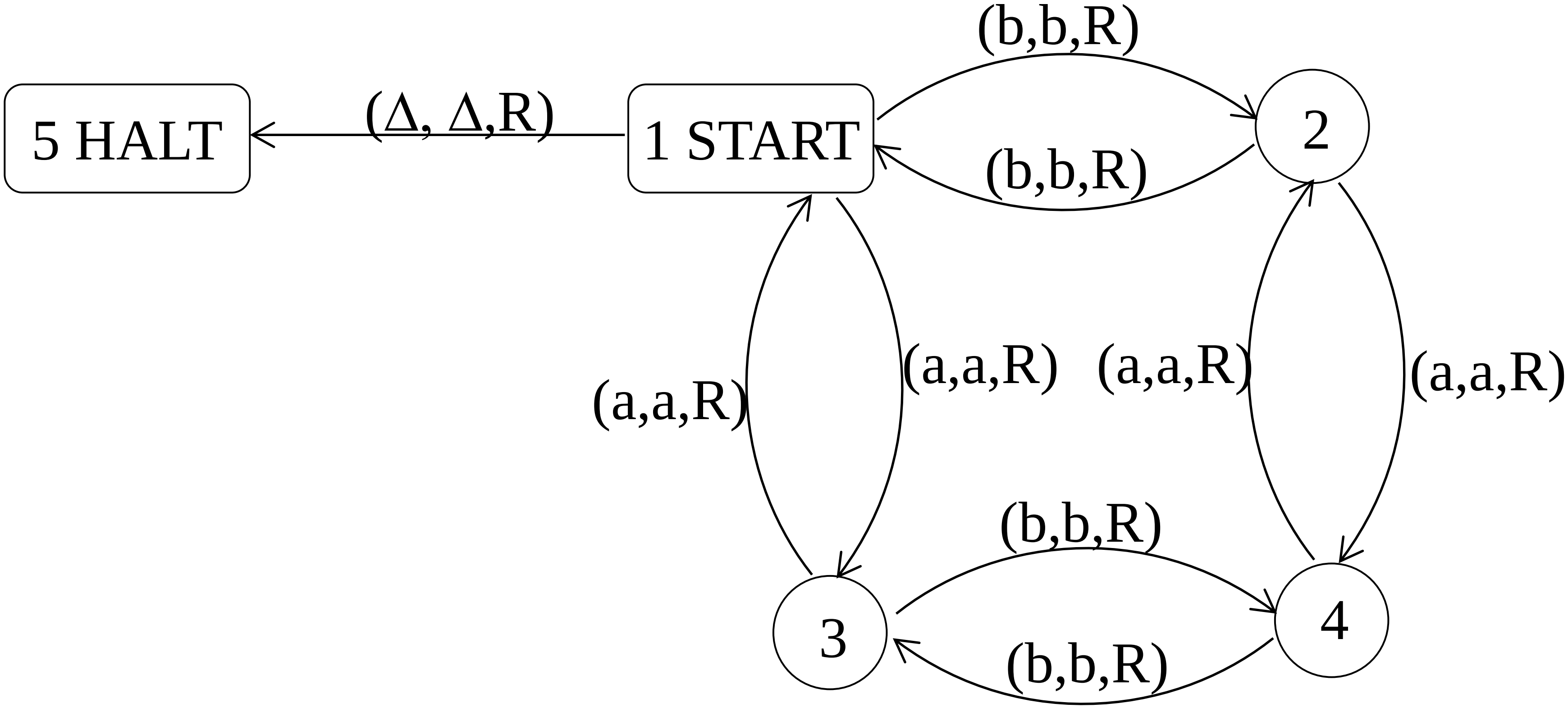
At state 2 reading b, state 3 is entered and the letter b is replaced by b, *i.e.*

$$\begin{array}{ccccc} 1 & & 2 & & 3 \\ \underline{a}ba & \boxed{?} & a\underline{b}a & \boxed{?} & ab\underline{a} \end{array}$$

At state 3 reading a, will keep the state of the TM unchanged. Lastly, the blank Δ is read and Δ is replaced by Δ and the HALT state is entered. Which can be expressed as

$$\overset{1}{\underline{a}ba} \rightarrow \overset{2}{a\underline{b}a} \rightarrow \overset{3}{ab\underline{a}} \rightarrow \overset{3}{aba\ \underline{\Delta}} \rightarrow \text{HALT}$$

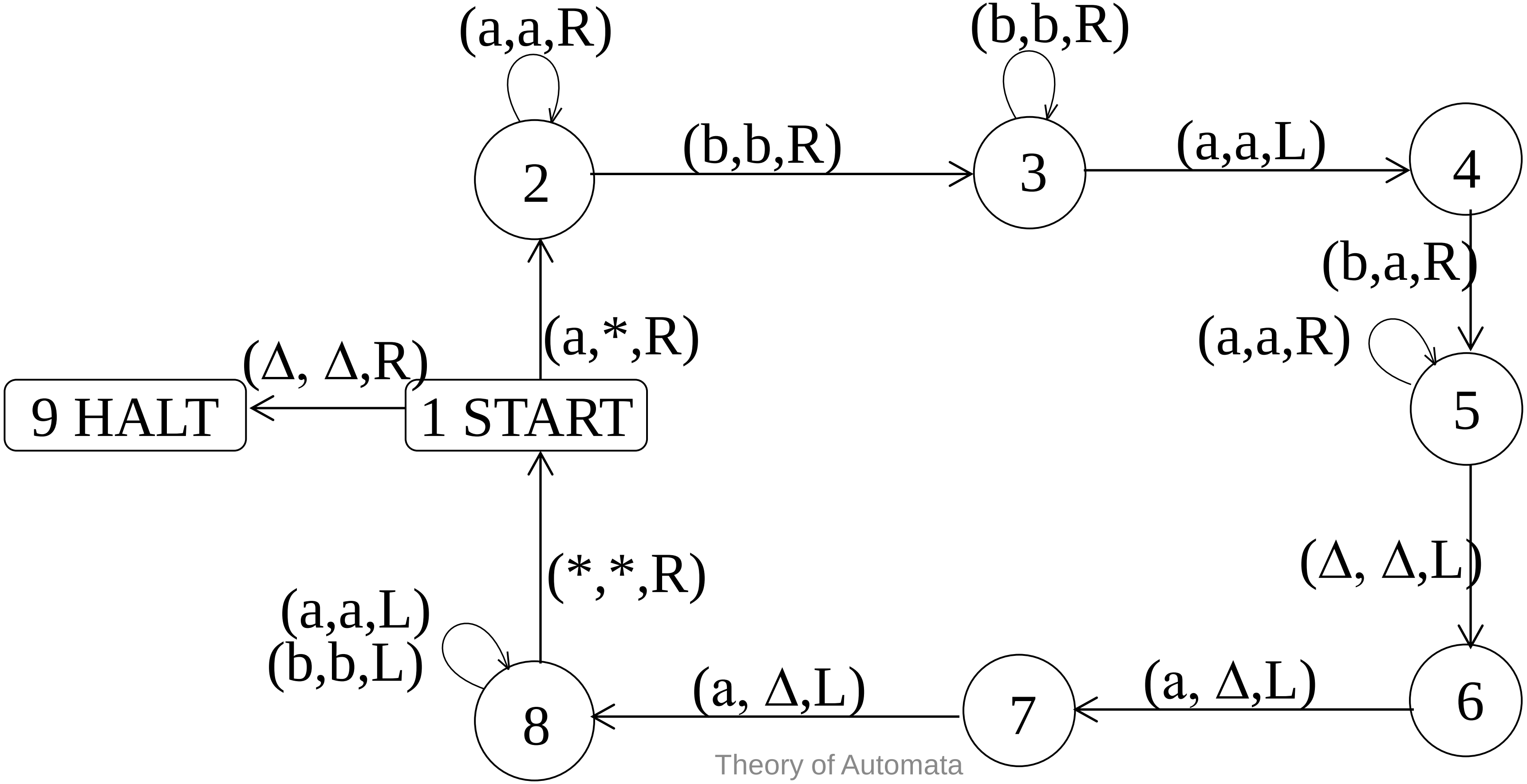
Which shows that the string aba is accepted by this machine. It can be observed, from the program of the TM, that the machine accepts the language expressed by (a+b)b(a+b)*.



It may be noted that the above diagram is similar to that of FA corresponding to EVEN-EVEN language. Following is another example

Example

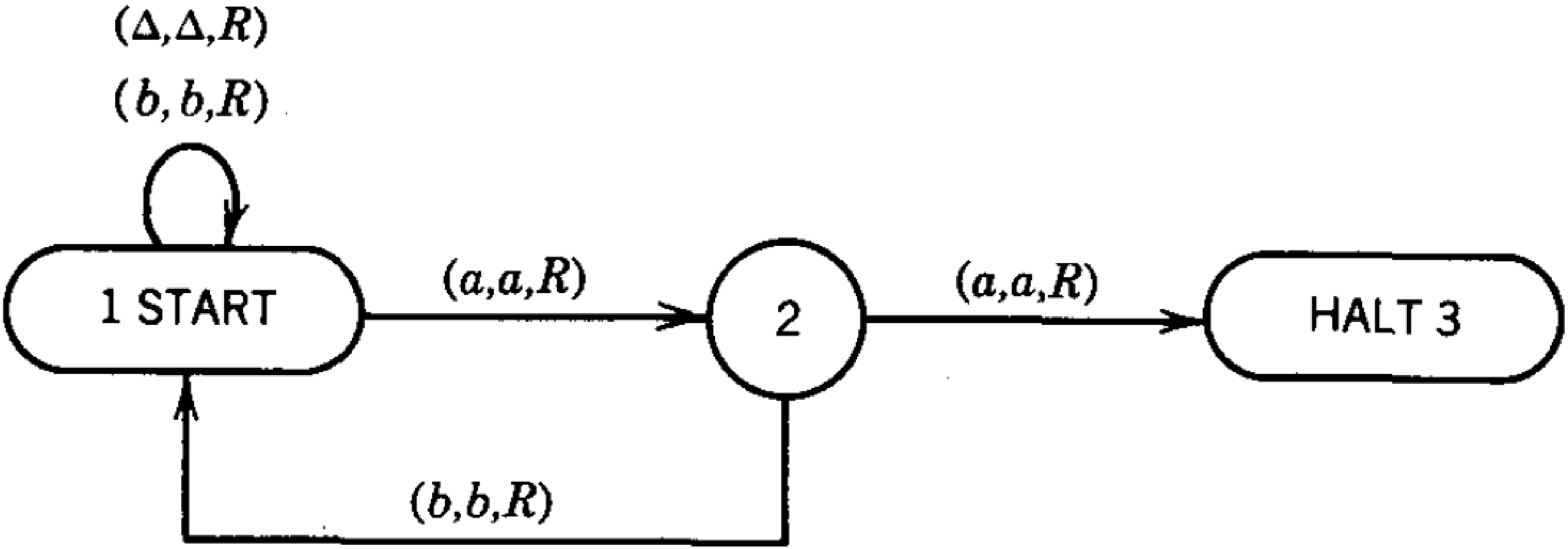
Consider the following TM



Example continued …

The string aaabbbaaa can be observed to be accepted by the above TM. It can also be observed that the above TM accepts the non-CFL $\{a^n b^n a^n\}$.

Must have double ‘aa’



1. Those with a double a. They are accepted by the TM.
2. Those without aa that end in a. They crash.
3. Those without aa that end in b. They loop forever.