Theory of Automata
Turing Machine

Week 12

Contents

- Definition
- Example
- Subprogram INSERT
- EQUAL
- Subprogram DELETE

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.

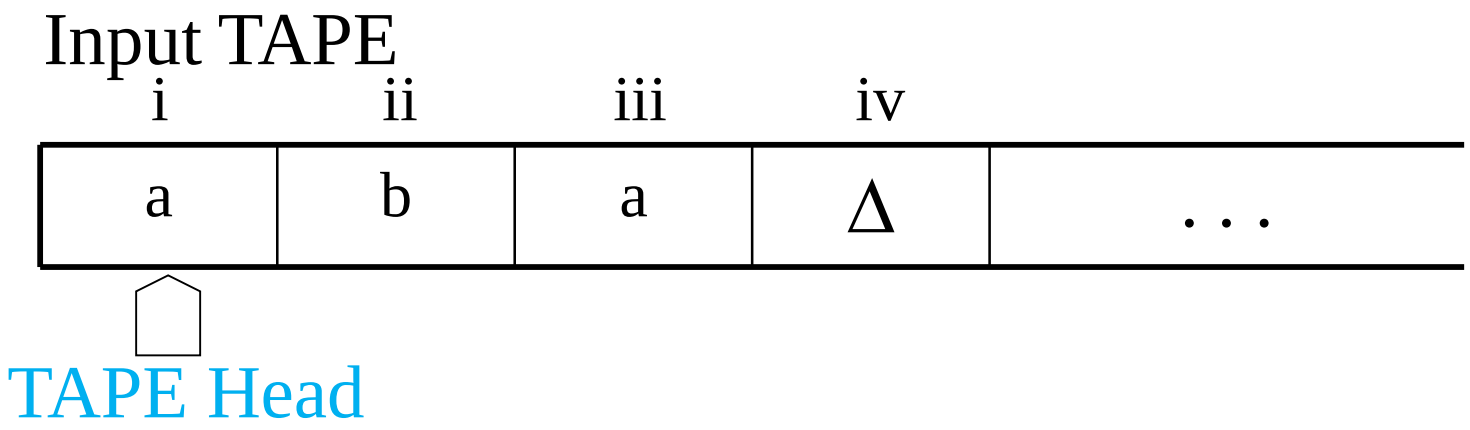
Alan Mathison Turing developed the machines called Turing machines, which accept some non-CFLs as well, in addition to CFLs.

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 …



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.

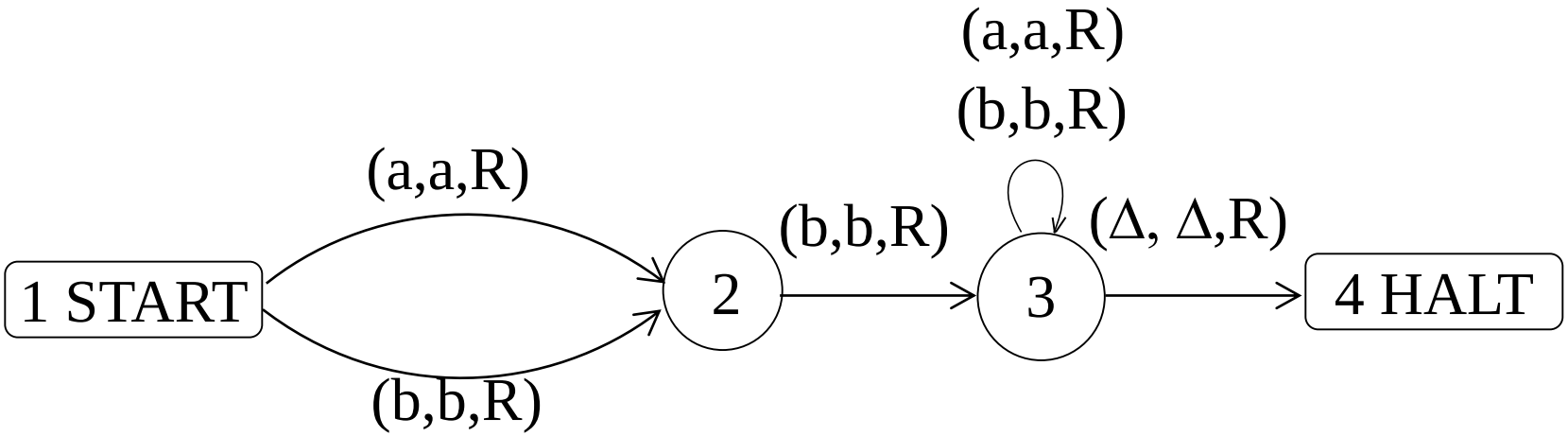
Note continued …

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

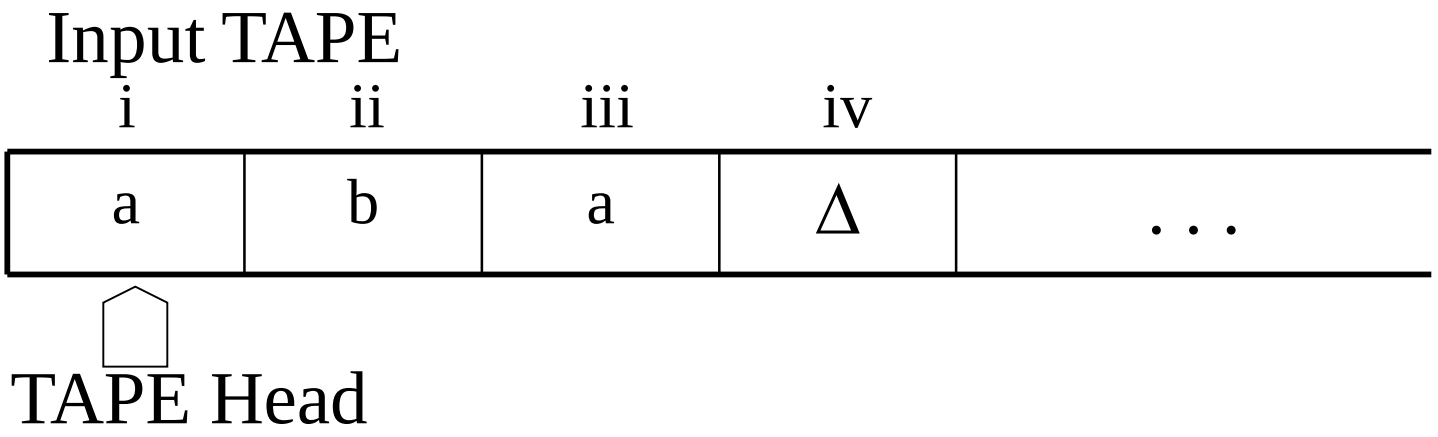
Example

Consider the following Turing machine



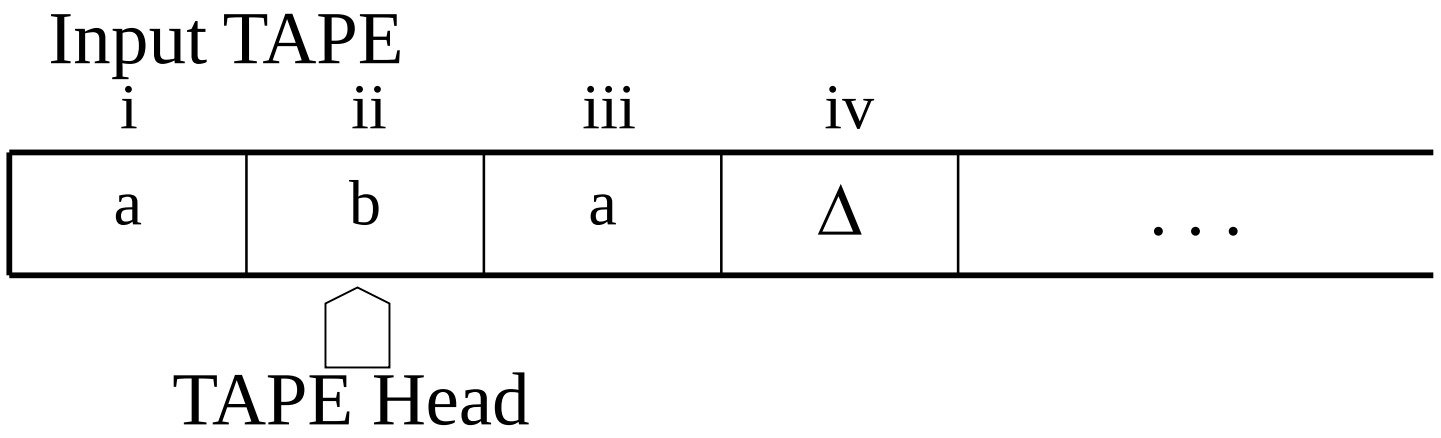
Let the input string *aba* be run over this TM

Example continued …

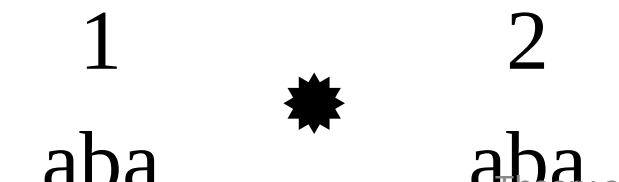


Starting from the START state, reading a form the TAPE and according to the TM program, a will be printed *i.e.* a will be replaced by a and the TAPE Head will be moved one cell to the right.

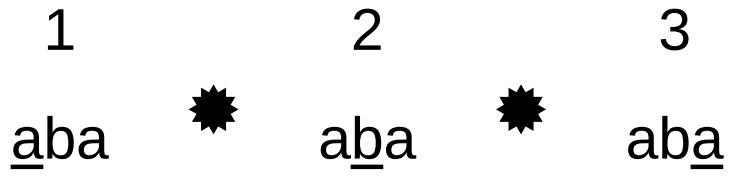
Which can be seen as



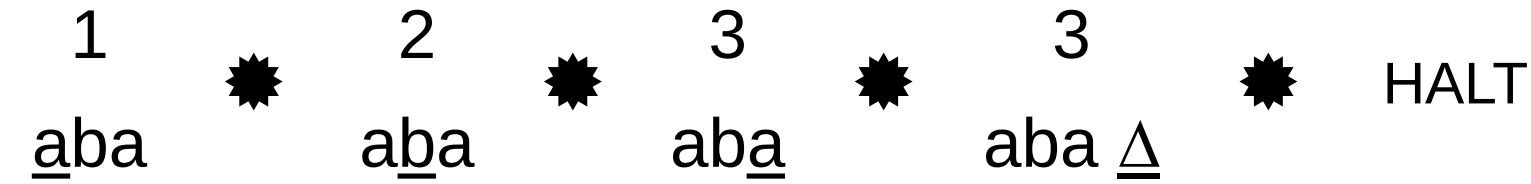
This process can be expressed as



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



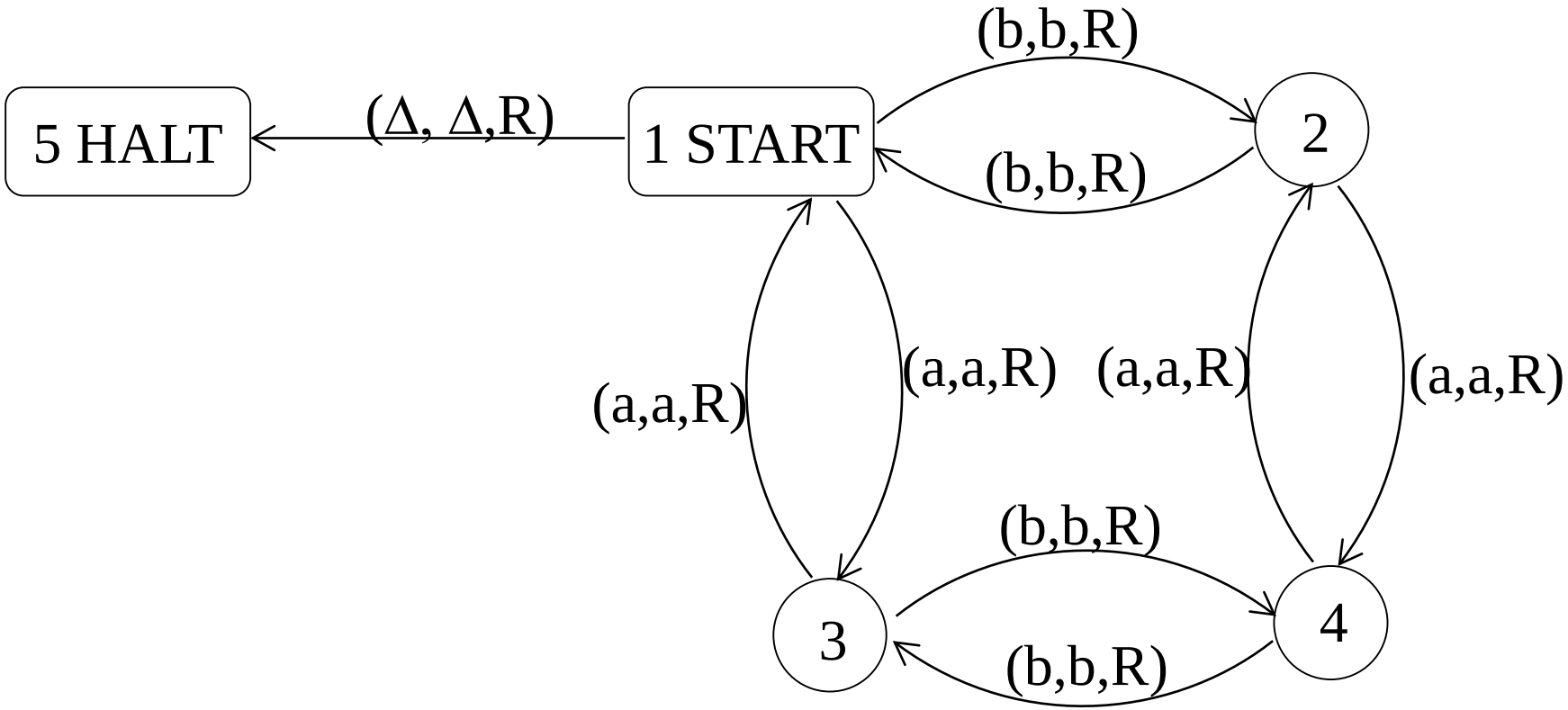
At state 3 reading a, will keep the state of the TM unchanged. Lastly, the blank Δ is read and Δ is replaced by Δ and the HALT state is entered. Which can be expressed as



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 $(a+b)b(a+b)^*$.

Theorem: Every regular language is accepted by some TM.

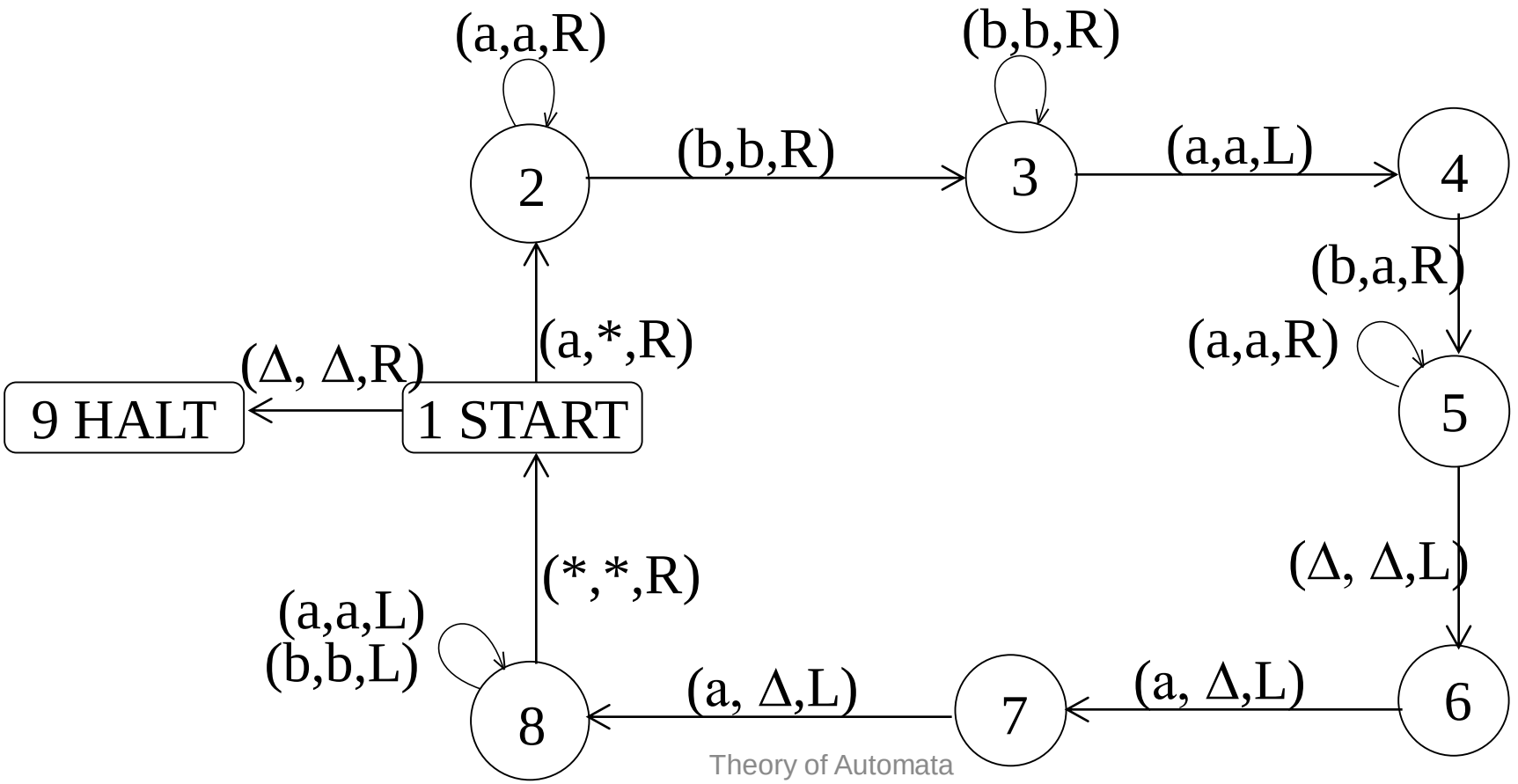
Example: Consider the EVEN-EVEN language. Following is a TM accepting the EVEN-EVEN language.



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

Example

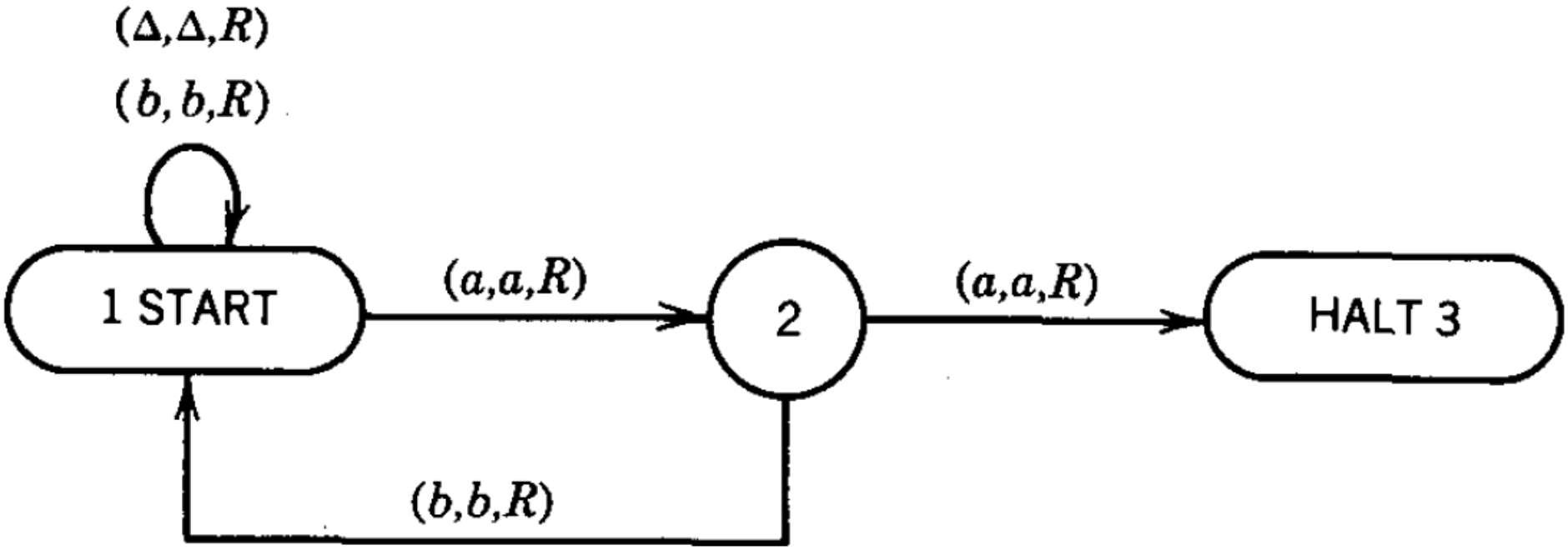
Consider the following TM



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.

DEFINITION

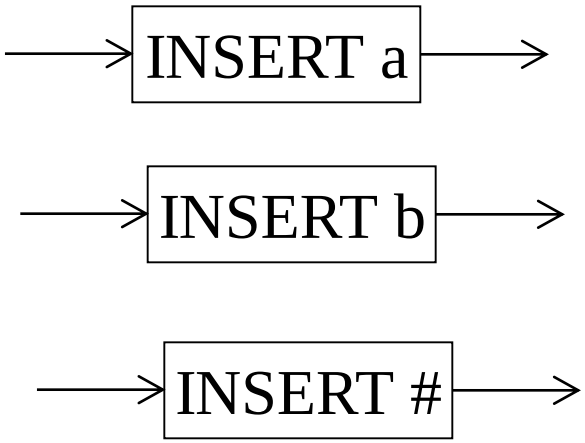
Every Turing machine T over the alphabet **divides the set of strings** into three classes:

1. ACCEPT(T) is the set of all strings leading to a HALT state. This is also called the *language accepted by T.*
2. REJECT(T) is the set of all strings that crash during execution by moving left from cell i or by being in a state that has no exit edge that wants to read the character the **TAPE HEAD is reading.**
3. LOOP(T) is the set of all other strings, that is, strings that loop forever while running on *T.*

INSERT subprogram

Sometimes, a character is required to be inserted on the TAPE exactly at the spot where the TAPE Head is pointing, so that the character occupies the required cell and the other characters on the TAPE are moved one cell right. The characters to the left of the pointed cell are also required to remain as such.

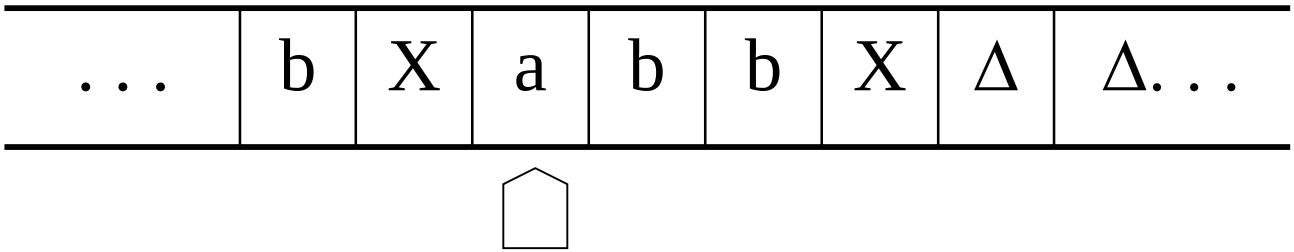
In the situation stated above, the part of TM program that executes the process of insertion does not affect the function that the TM is performing. The subprogram of insertion is independent and can be incorporated at any time with any TM program specifying what character to be inserted at what location. The subprogram of insertion can be expressed as



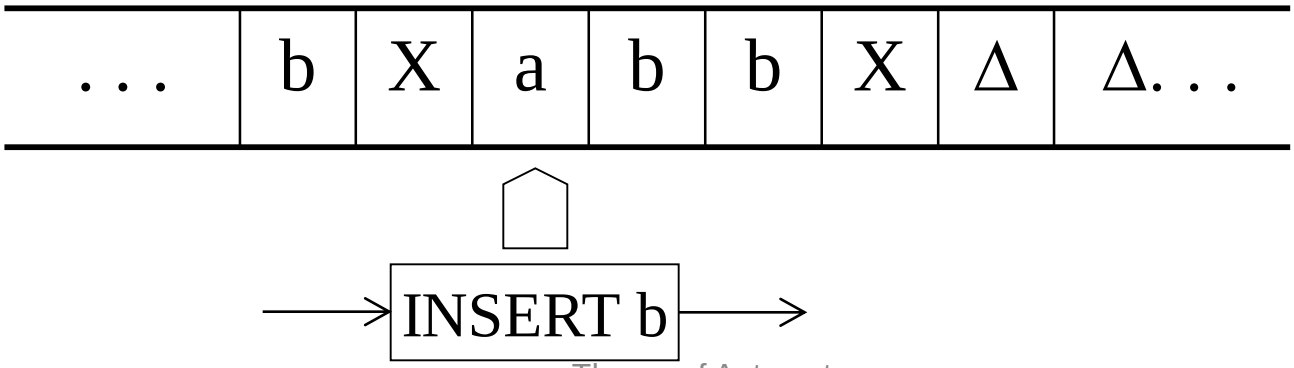
The above diagrams show that the characters a,b and # are to be inserted, respectively. Following is an example showing how does the subprogram INSERT perform its function

Example

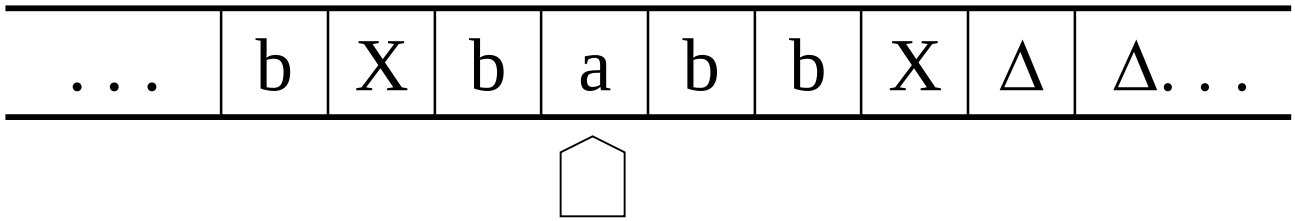
If the letter b is inserted at the cell where the TAPE Head is pointing as shown below



then, it is expressed as



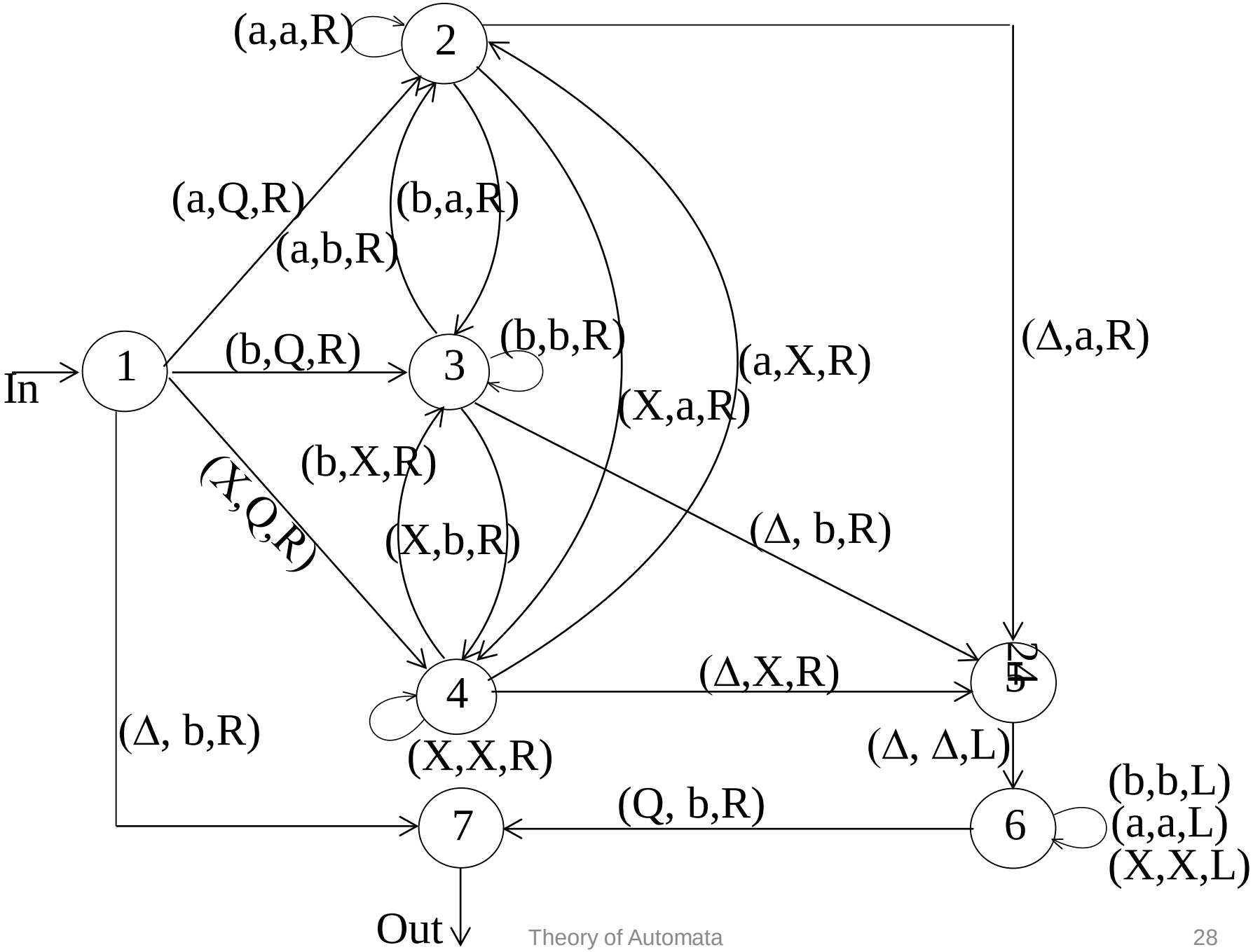
The function of subprogram INSERT b can be observed from the following diagram



Following is the INSERT subprogram

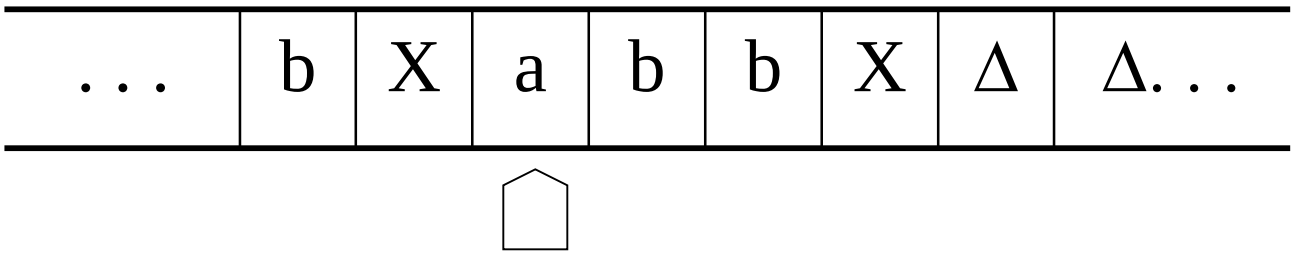
The subprogram INSERT

Keeping in view the same example of inserting b at specified location, to determine the required subprogram, first Q will be inserted as marker at the required location, so that the TAPE Head must be able to locate the proper cell to the right of the insertion cell. The whole subprogram INSERT is given as


(a,a,R)
2
(a,Q,R)
(b,a,R)
(a,b,R)
(b,b,R)
(b,Q,R)
1
3
In
(a,X,R)
(Δ,a,R)
(X,a,R)
(b,X,R)
(X,Q,R)
(Δ, b,R)
(X,b,R)
4
5
(Δ,X,R)
(Δ, b,R)
(X,X,R)
(Δ, Δ,L)
(b,b,L)
(Q, b,R)
7
6
(a,a,L)
(X,X,L)
Out

It is supposed that machine is at state 1, when b is to be inserted. All three possibilities of reading a, b or X are considered by introducing the states 2, 3 and 4 respectively. These states remember what letter displaced during the insertion of Q.

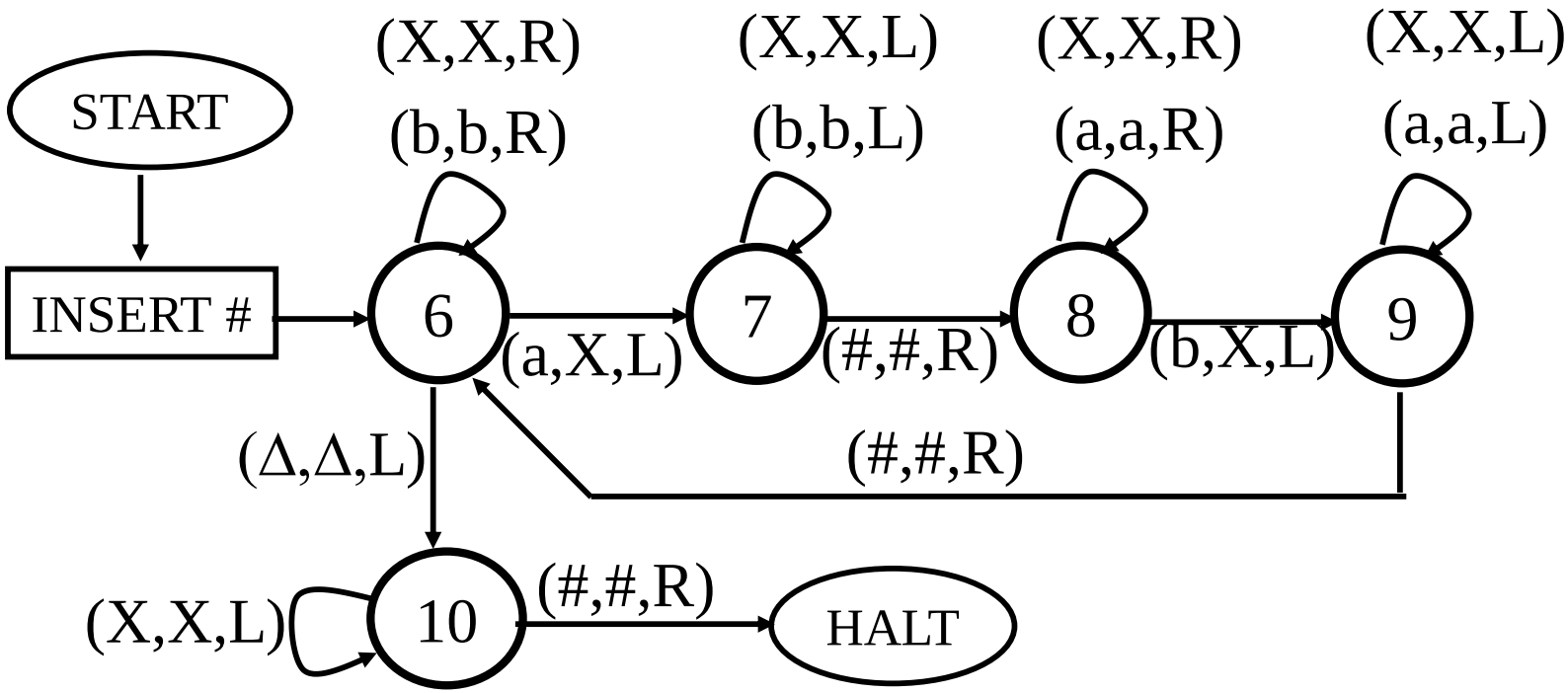
Consider the same location where b is to be inserted



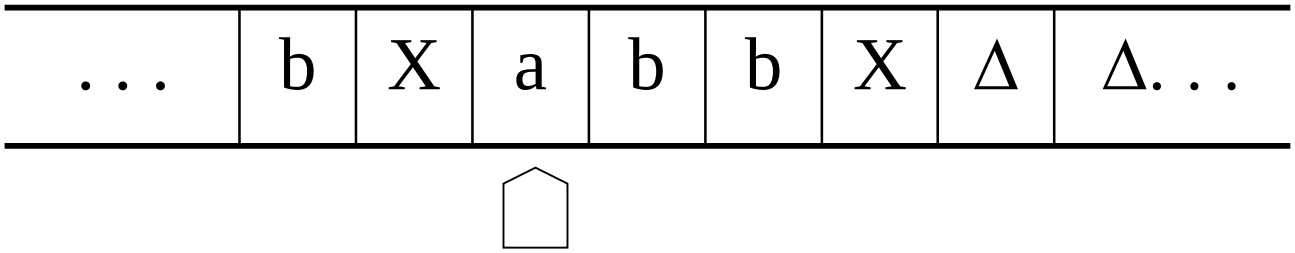
After reading a from the TAPE, the program replaces a by Q and the TAPE Head will be moved one step right. Here the state 2 is entered. Reading b at state 2, b will be replaced by a and state 3 will be entered. At state 3 b is read which is not replaced by any character and the state 3 will not be left.

At state 3, the next letter to be read is X, which will be replaced by b and the state 4 will be entered. At state 4, Δ will be read, which will be replaced by X and state 5 will be entered. At state 5 Δ will be read and without any change state 6 will be entered, while TAPE Head will be moved one step left. The state 6 makes no change whatever (except Q) is read at that state. However at each step, the TAPE Head is moved one step left. Finally, Q is read which is replaced by b and the TAPE Head is moved to one step right.

EQUAL


START
INSERT #
6
7
8
9
10
HALT
(X,X,R)
(b,b,R)
(X,X,L)
(b,b,L)
(X,X,R)
(a,a,R)
(X,X,L)
(a,a,L)
(a,X,L)
(#,#,R)
(b,X,L)
(Δ,Δ,L)
(#,#,R)
(X,X,L)
(#,#,R)

Hence, the required situation of the TAPE can be shown as



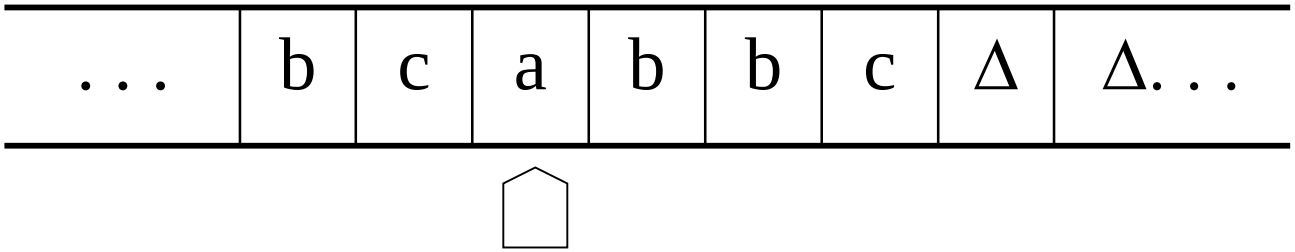
DELETE subprogram

Sometimes, a character is required to be DELETED on the TAPE exactly at the spot where the TAPE Head is pointing, so that the other characters on the right of the TAPE Head are moved one cell left. The characters to the left of the pointed cell are also required to remain as such.

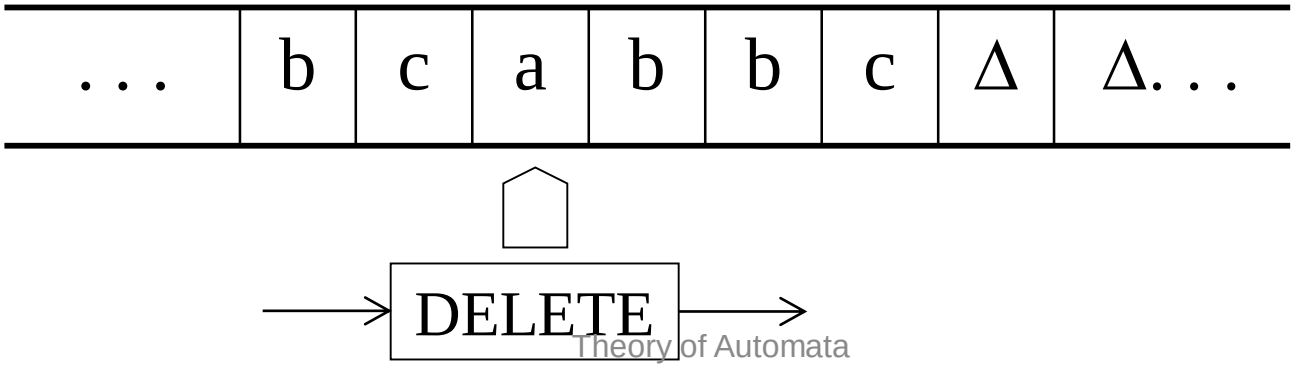
In the situation stated above, the part of TM program that executes the process of deletion does not affect the function that the TM is performing. The subprogram of deletion is independent and can be incorporated at any time with any TM program specifying what character to be deleted at what location. The subprogram of deletion can be expressed as

Example

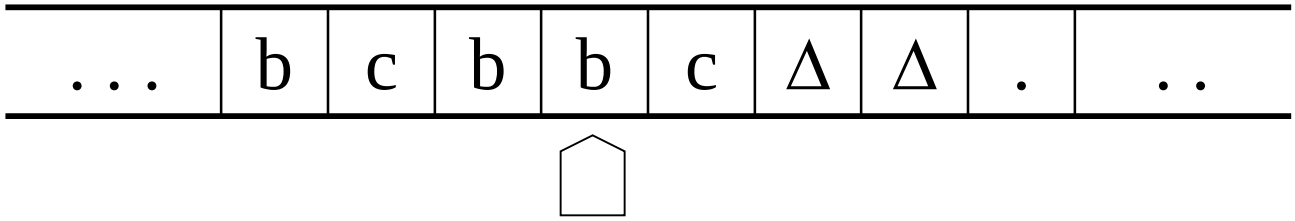
If the letter a is to be deleted from the string bcabbc, shown below



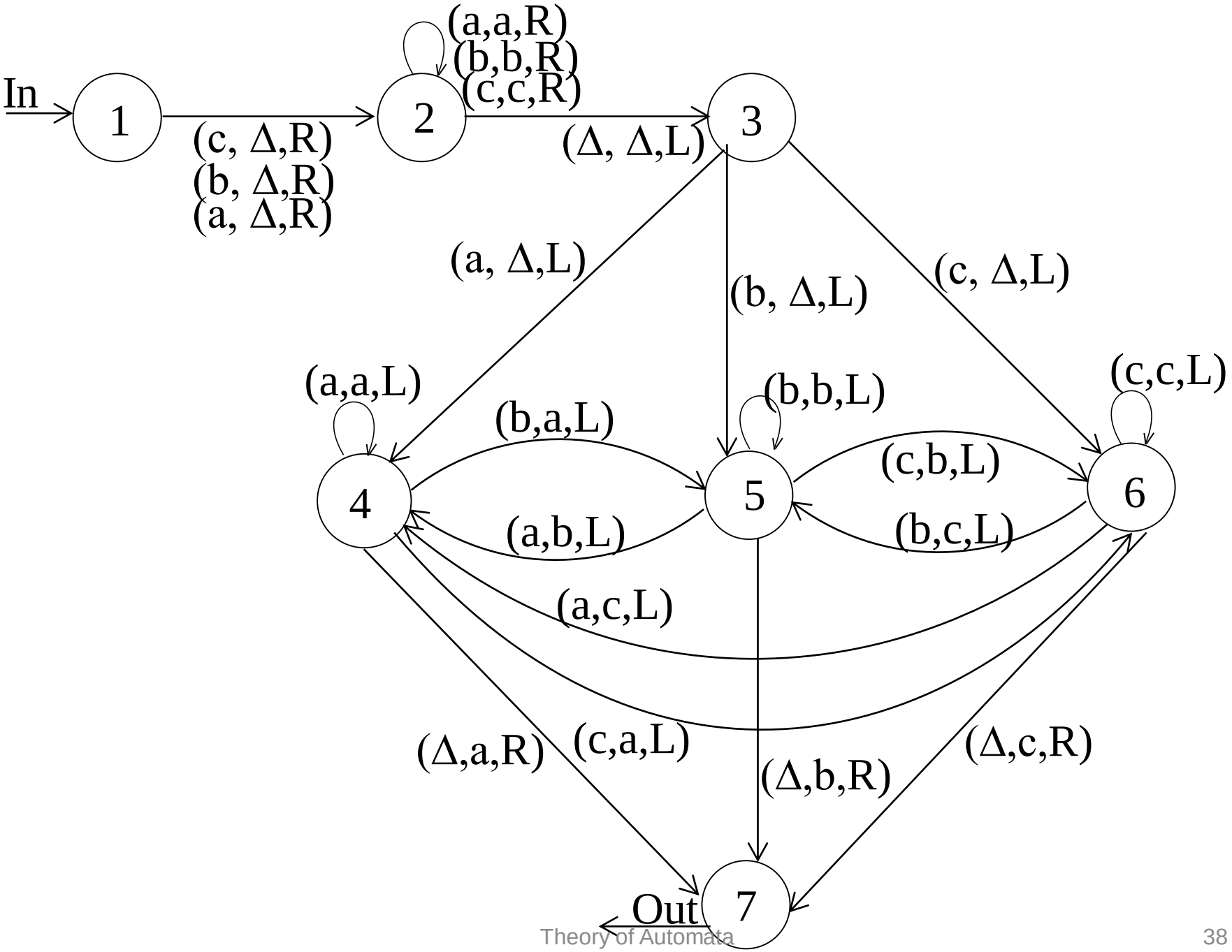
then, it is expressed as



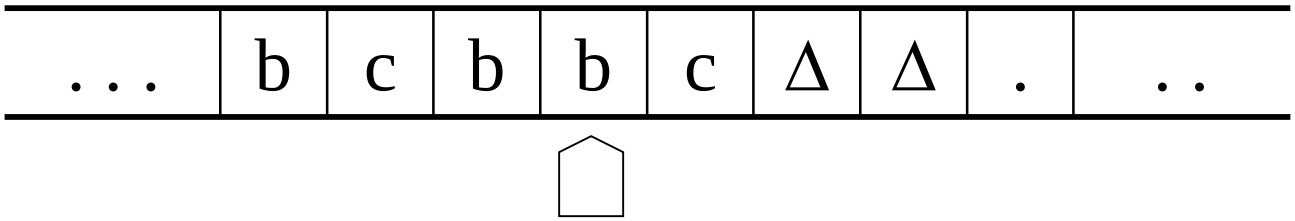
The function of subprogram DELETE can be observed from the following diagram



Following is the DELETE subprogram

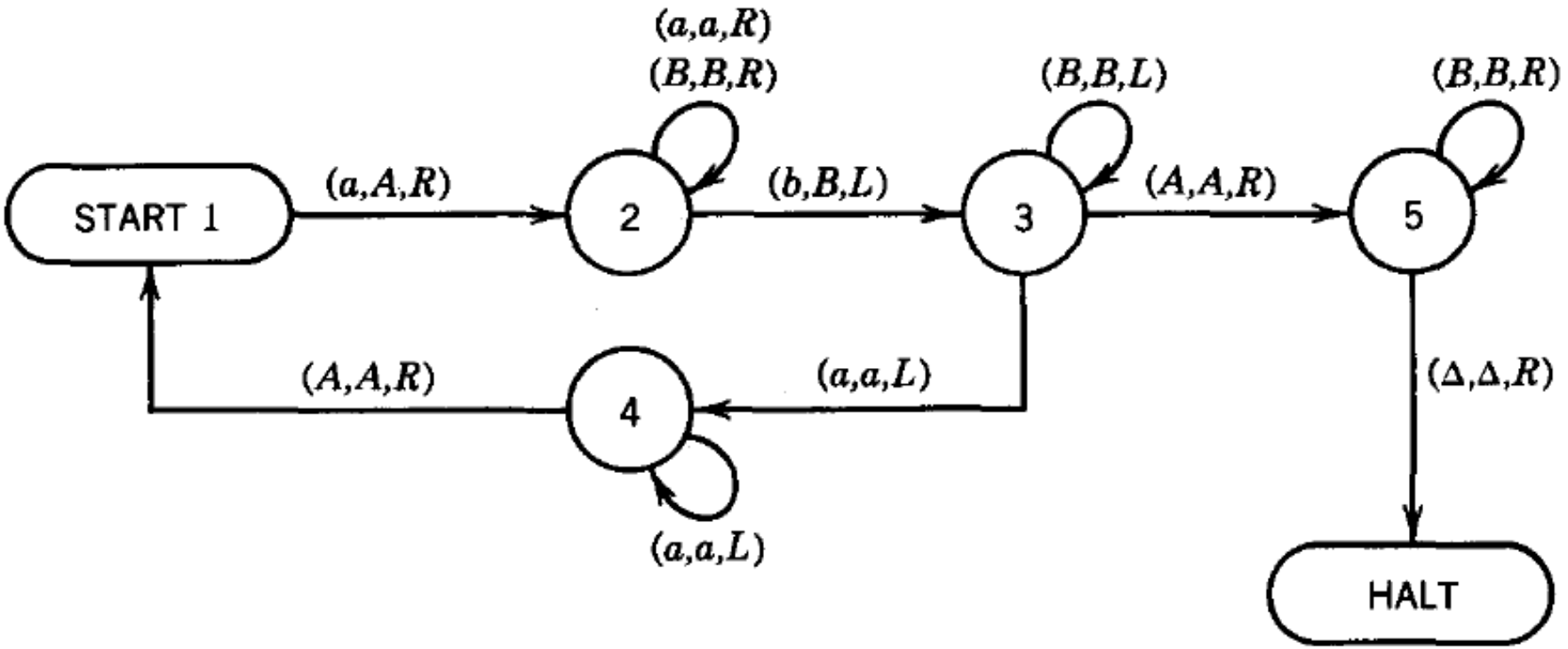

In
1
2
3
4
5
6
7
Out
(a,a,R)
(b,b,R)
(c,c,R)
(c, Δ,R)
(b, Δ,R)
(a, Δ,R)
(Δ, Δ,L)
(a, Δ,L)
(b, Δ,L)
(c, Δ,L)
(a,a,L)
(b,a,L)
(b,b,L)
(c,c,L)
(c,b,L)
(a,b,L)
(b,c,L)
(a,c,L)
(Δ,a,R)
(c,a,L)
(Δ,b,R)
(Δ,c,R)

The process of deletion of letter a from the string bcabbc can easily be checked, giving the TAPE situation as shown below

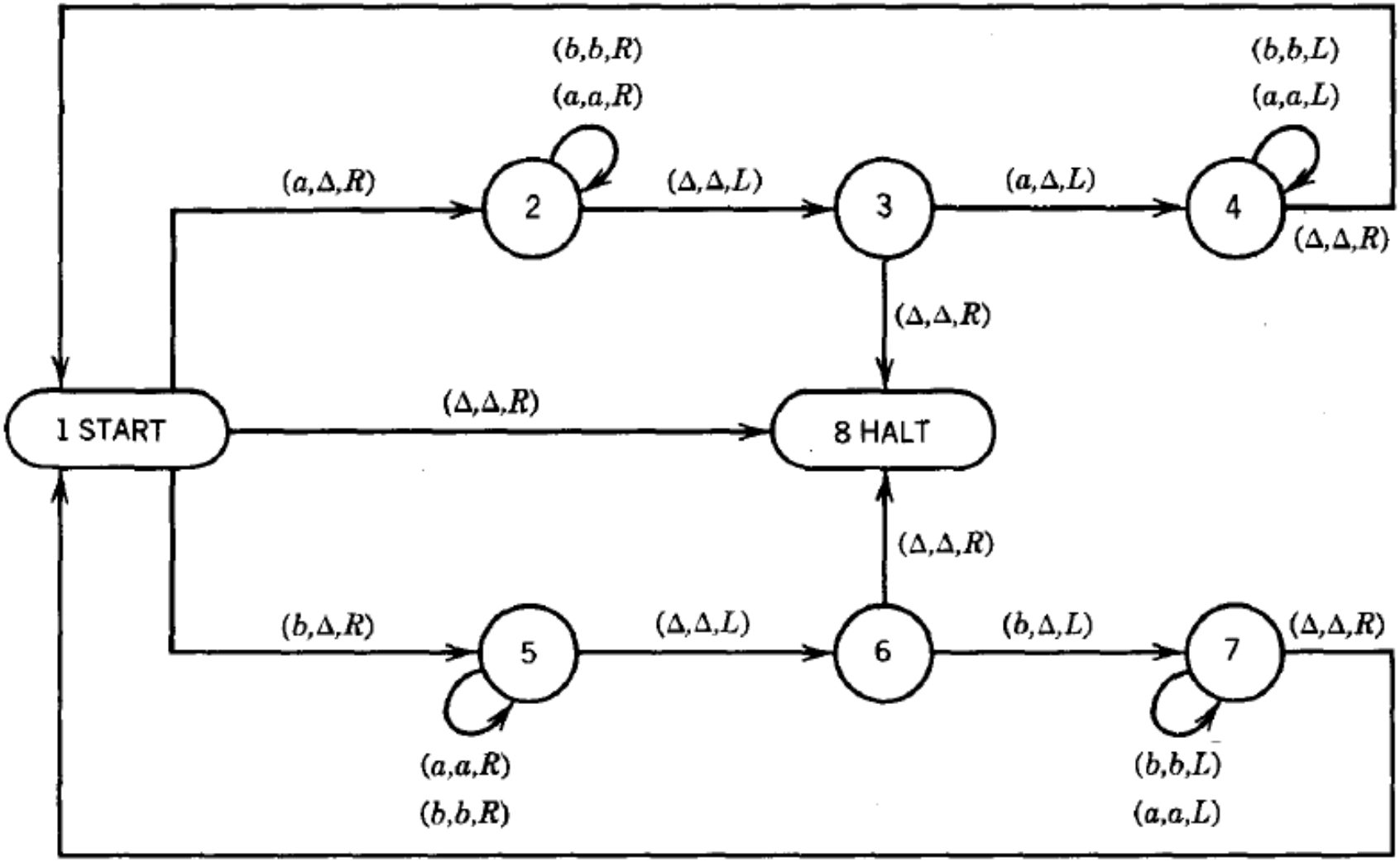


Language Defined by	Corresponding Acceptor	Nondeterminism = determinism?	Language Closed Under	What Can be Decided	Example of Application
Regular expression	Finite automaton Transition graph	Yes	Union, product, Kleene star, intersection, complement	Equivalence, emptiness, finiteness, membership	Text editors, sequential circuits
Context-free grammar	Pushdown automaton	No	Union, product, Kleene star	Emptiness finiteness membership	Programming language statements, compilers
Type 0 grammar	Turing machine, Post machine, 2PDA, *n*PDA	Yes	Union, product, Kleene star	Not much	Computers

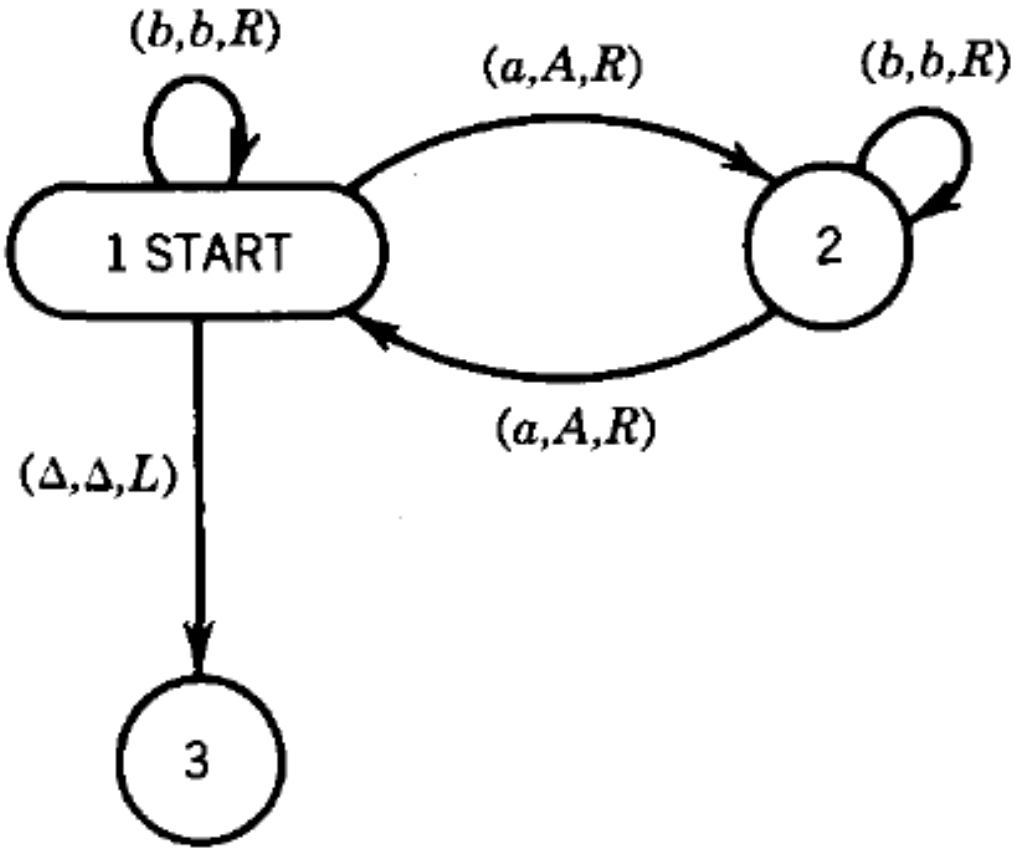
$a^n b^n$



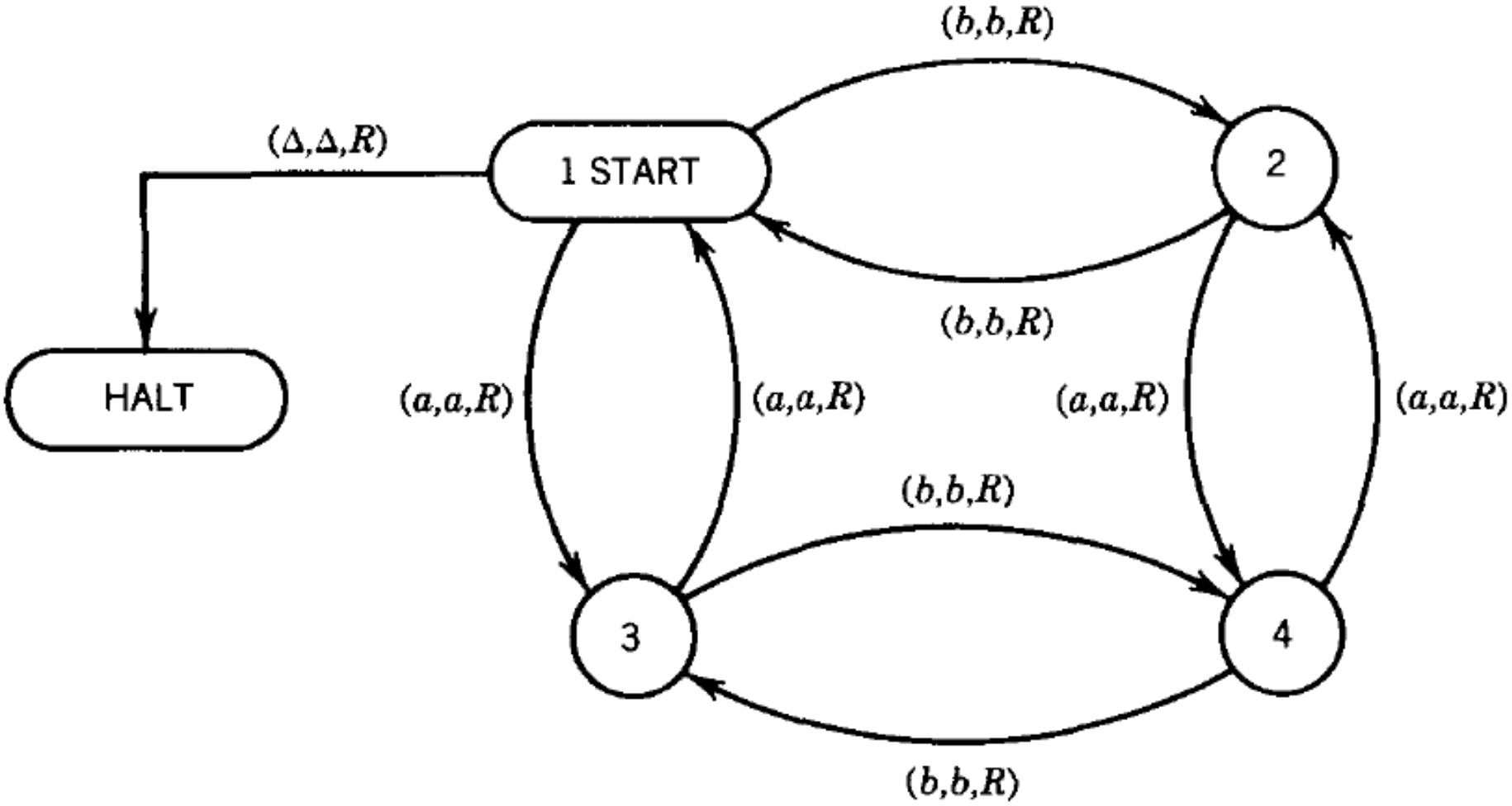
OddPalindrome / EvenPalindrome



Even a's



Even-Even



Double Word Problem

The language this TM accepts is DOUBLEWORD, the set of all words of the form ww where w is a nonnull string in *{a, b}*.*
DOUBLEWORD **= {aa *bb aaaa abab baba bbbb*** *aaaaaa aabaab . . }*

Sample Problems

16. Build a TM to accept the language $\{a^n b^n a^n\}$ based on the following algorithm:

    (i) Check that the input is in the form **a*b*a***.

    (ii) Use DELETE in an intelligent way.

11. Return to the example in the text of a TM to accept EVEN-EVEN based on the algorithm

    1. Move up the string changing *a*'s to *A*'s
    2. Move down the string changing *b*'s to *B*'s

    We can modify this algorithm in the following way: To avoid the problem of crashing on the way down the TAPE change the letter in the first cell to *X* if it is an *a* and to *Y* if it is a *b*. This way, while charging down the TAPE we can recognize when we are in cell i.

    Draw this TM.

13. Build a TM that accepts the language EVEN-EVEN based on the subroutine DELETE given in this chapter.

Equal-Equal

S → aSbS | bSaS | ε

OR

S → aSb | bSa | SS | ε