Theory of Automata
2PDA

Week 13

Contents

- Two Stack PDA
- Just another TM “2PDA = TM”
- Simulate a TM over PM
- nPDA = TM

2PDA or Two Stack PDA

- Let us define a two-stack pushdown automaton

as follows

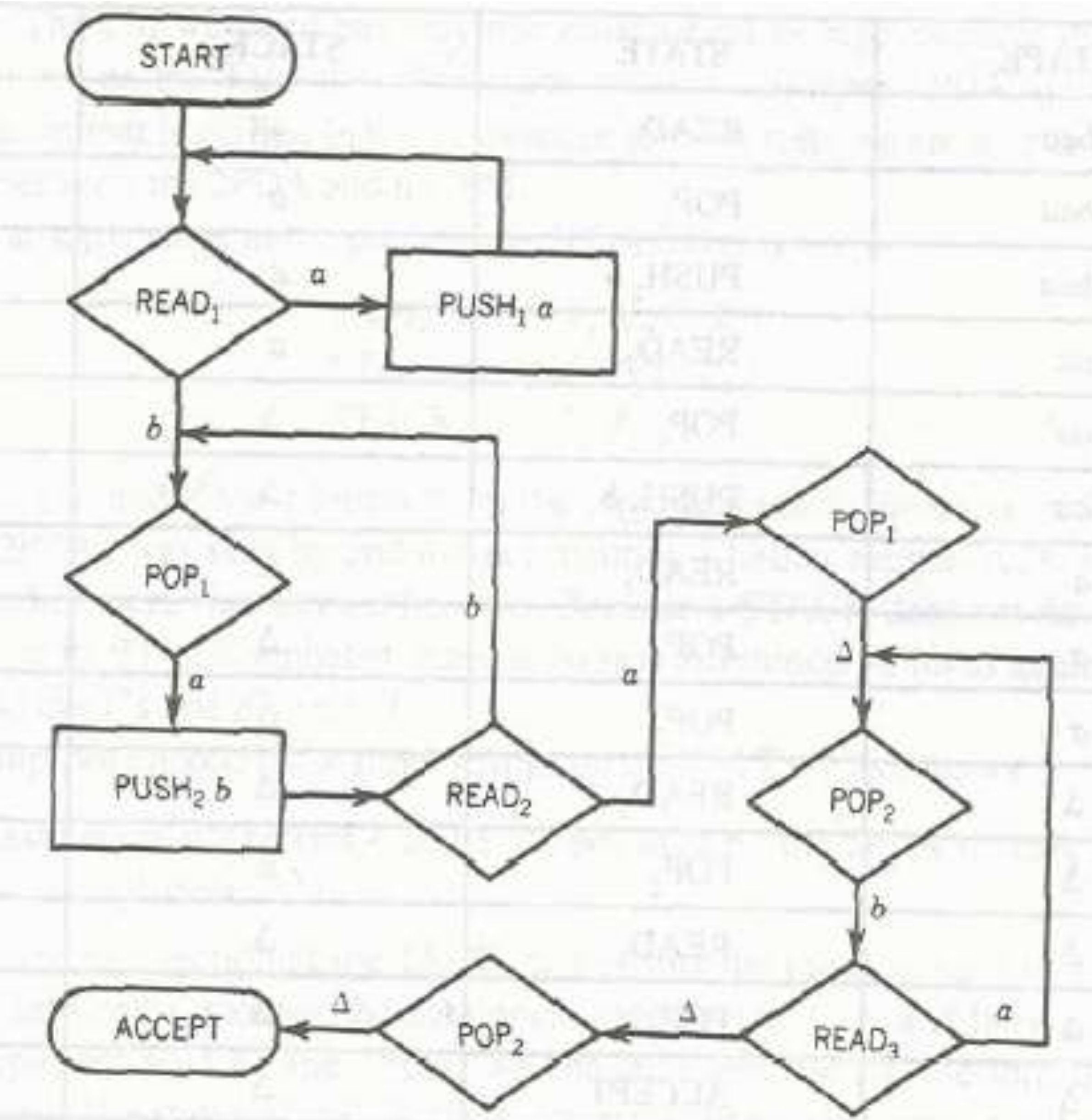
M = (K, Σ, Γ, Δ, s, F)

- K is a finite set of states,
- Σ is an alphabet (the input symbols),
- Γ is an alphabet (the stack symbols),
- $s \in K$ is the start state,
- $F \subseteq K$ is the set of final states, and
- Finite set of transition from one state to another state

2PDA

- As we have two stacks for storage purpose, so change the name of POP and PUSH stack accordingly.

- POP_1 will pop from the first stack
- POP_2 will pop from the second stack
- $PUSH_1$ will push into first stack
- $PUSH_2$ will push into second stack



- Guess the Language
- $a^n b^n a^n$

TAPE	STATE	$STACK_1$	$STACK_2$
aabbaa	START	Δ	Δ
abbaa	$READ_1$	Δ	Δ
abbaa	$PUSH_1$ *a*	*a*	Δ
bbaa	$READ_1$	*a*	Δ
bbaa	$PUSH_1$ *a*	*aa*	Δ
baa	$READ_1$	*aa*	Δ
baa	POP_1	*a*	Δ
baa	$PUSH_2$ *b*	*a*	*b*
aa	$READ_2$	*a*	*b*
aa	POP_1	Δ	*b*
aa	$PUSH_2$ *b*	Δ	*bb*
a	$READ_2$	Δ	*bb*
a	POP_1	Δ	*bb*
a	POP_2	Δ	*b*
Δ	$READ_3$	Δ	*b*
Δ	POP_2	Δ	Δ
Δ	$READ_3$	Δ	Δ
Δ	POP_2	Δ	Δ
Δ	ACCEPT	Δ	Δ

- Run *aabbaa*

Just another TM “2PDA = TM”

Minsky’s Theorem

Theorem:

Any language accepted by a 2PDA can be accepted by some TM and any language accepted by a TM can also accepted by 2PDA.

Proof :

2PDA has 3 locations for storage

1. Stack1
2. Stack2
3. Input Tape

2PDA = TM

Step-1 assume # and $ are not used by 2PDA
Step-2 Divide TM into three parts
Insert Δ in cell i.

First Part

contains input string

- We change each letter into Δ as the letters are read by the 2PDA.
- Put # at the end of input strings as end marker

Second Part

Contains the Stack1 of the 2PDA
$ is placed as the end of stack1 contents over the input tape

Third Part

Contains the Stack2 of the 2PDA
will indicating the end of stack2 contents

Read simulation

TAPE $x_1, x_2\ x_3\ x_4$

$STACK_1$ $Y_1\ Y_2\ Y_3\ Y_4\ Y_5$

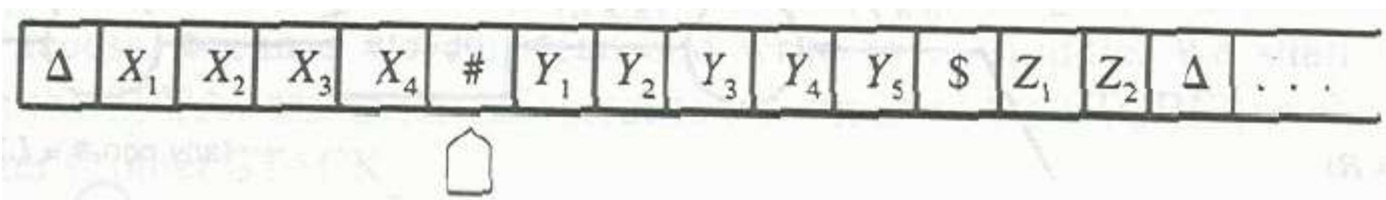
$STACK_2$ Z_1, Z_2

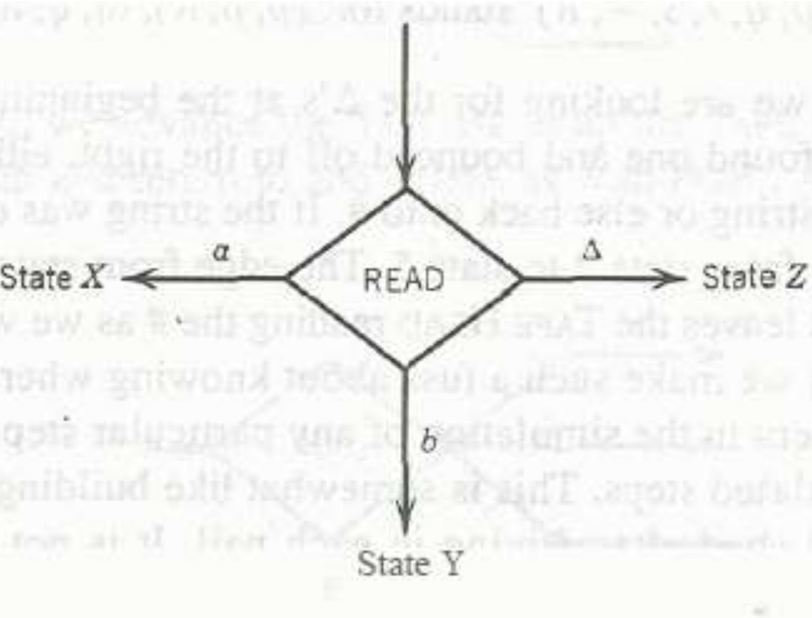
The input tape will be

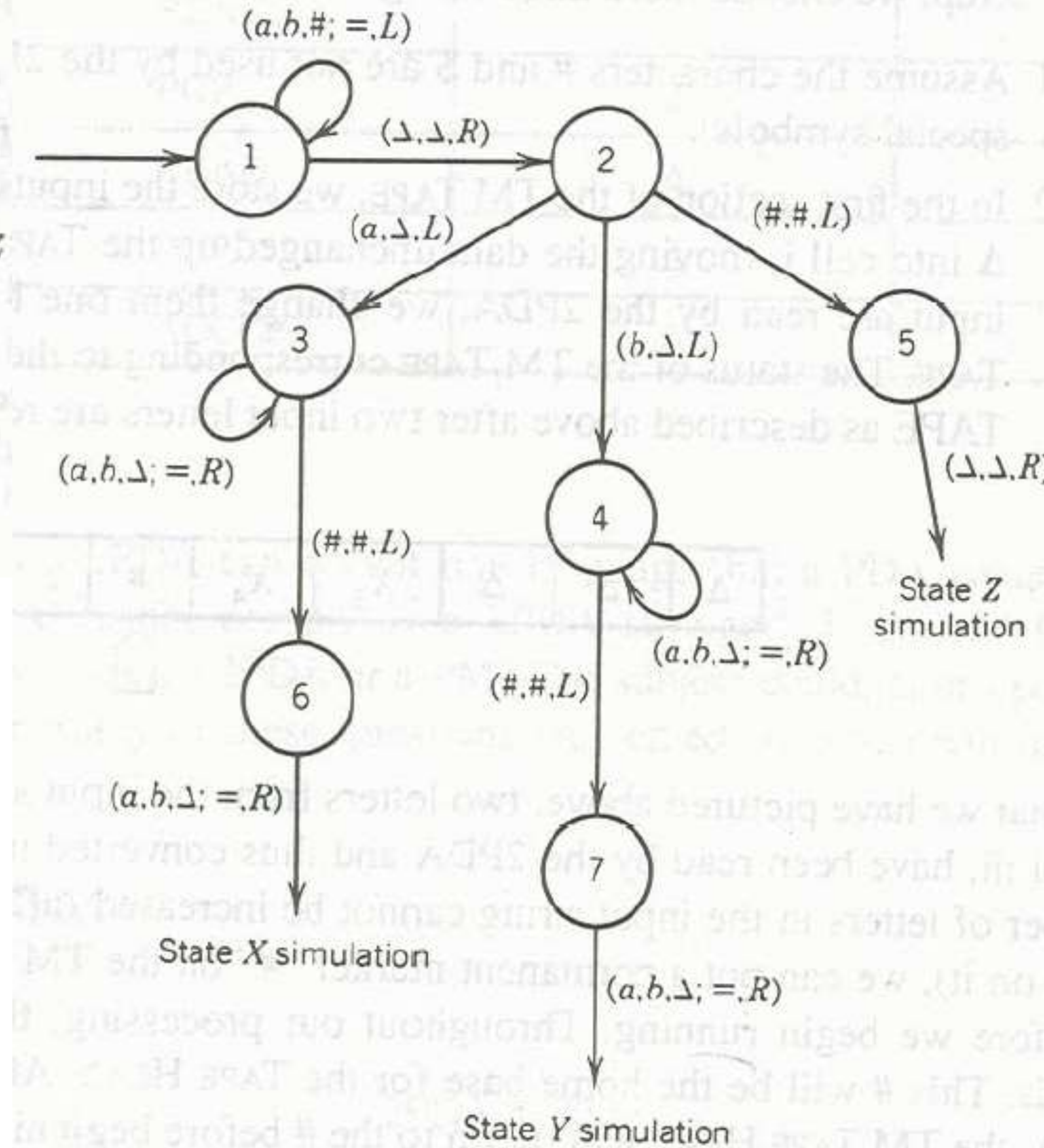
Δ	Δ	Δ	X_3	X_4	#	…

After simulating any action of 2PDA TAPE HEAD will return to # before beginning its next operation

- After putting input string and two stack over the input TAPE of Turing machine, the Input TAPE will be as under







(a,b,#;=,L)

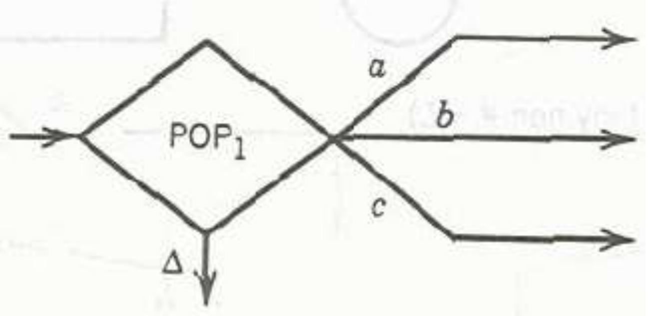
Above statement equals

(a,a,L)

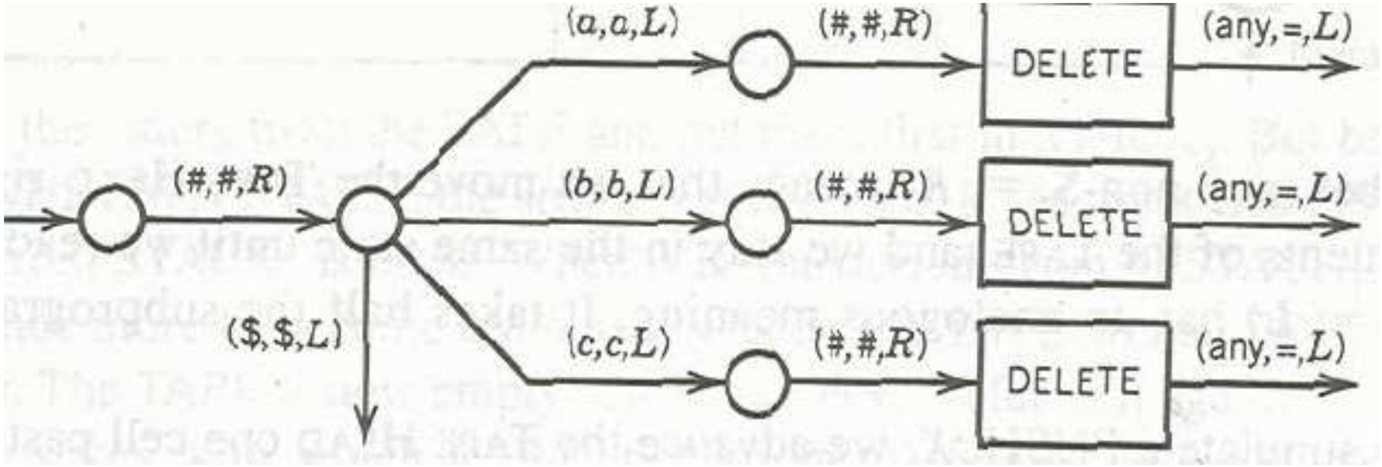
(b,b,L)

(#,#,L)

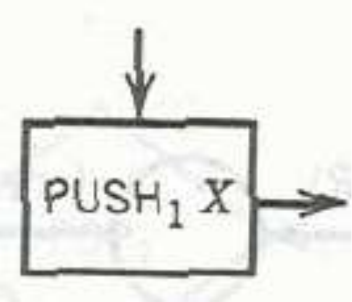
Simulate POP_1 instruction



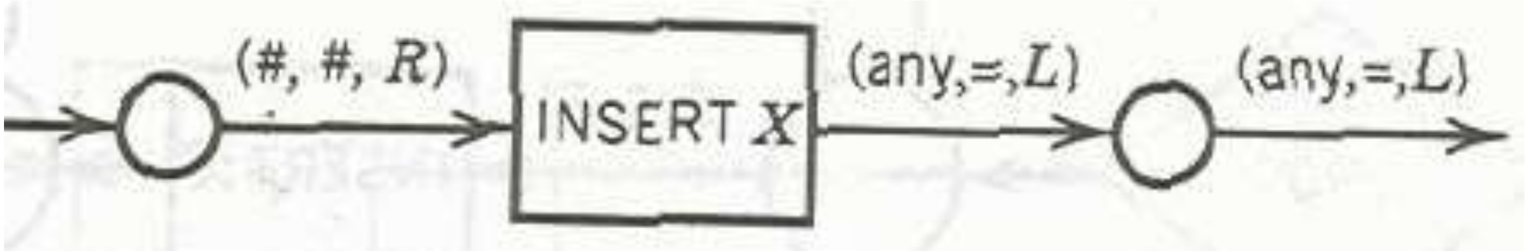
- Becomes



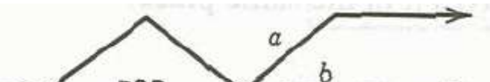
Simulating $Push_1$

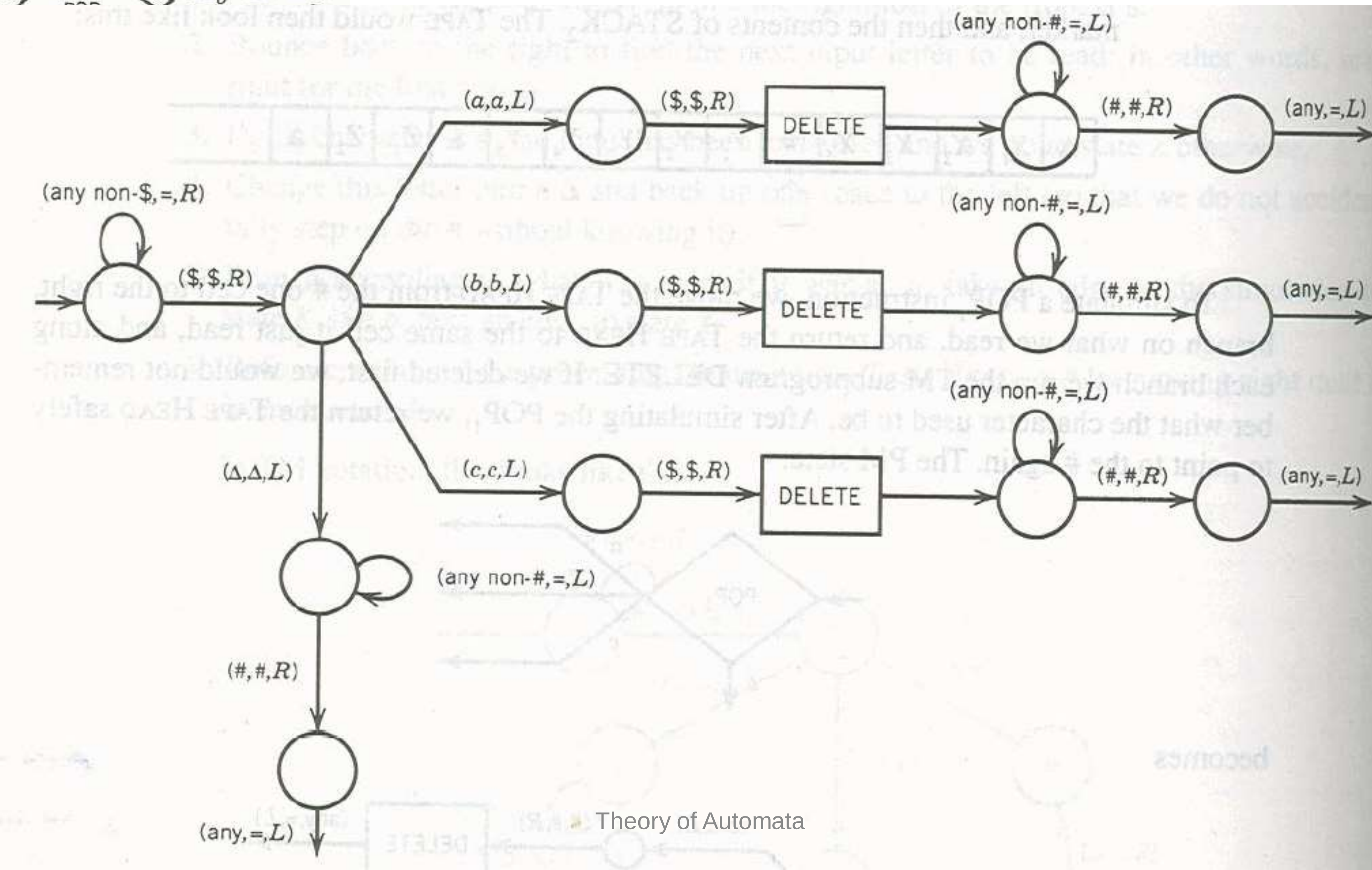


Becomes

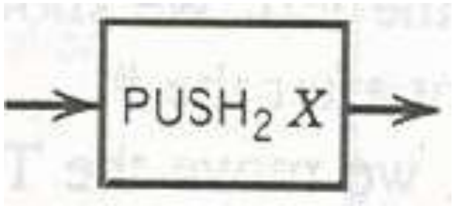


Simulating POP$_2$

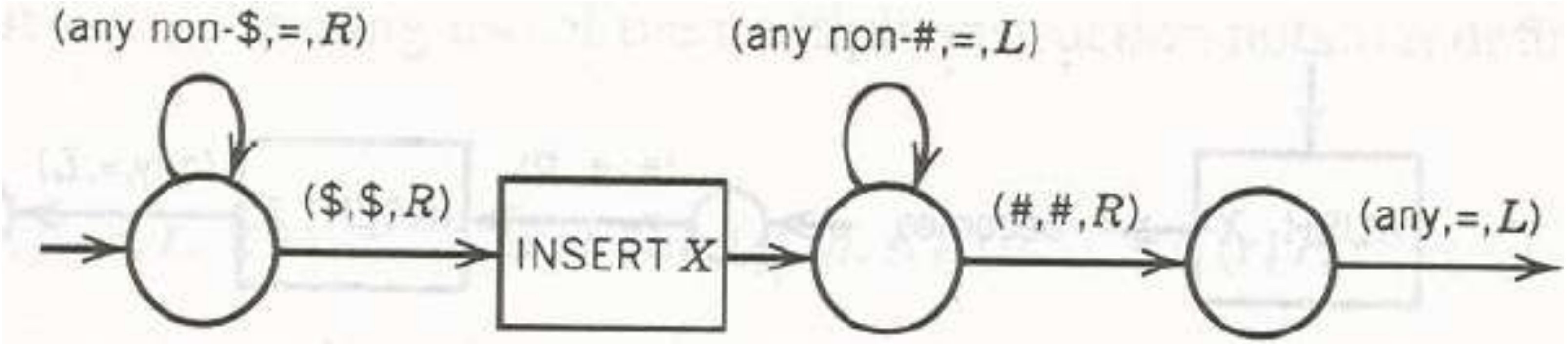




Simulating $Push_2$



- Becomes



Simulate a TM over PM

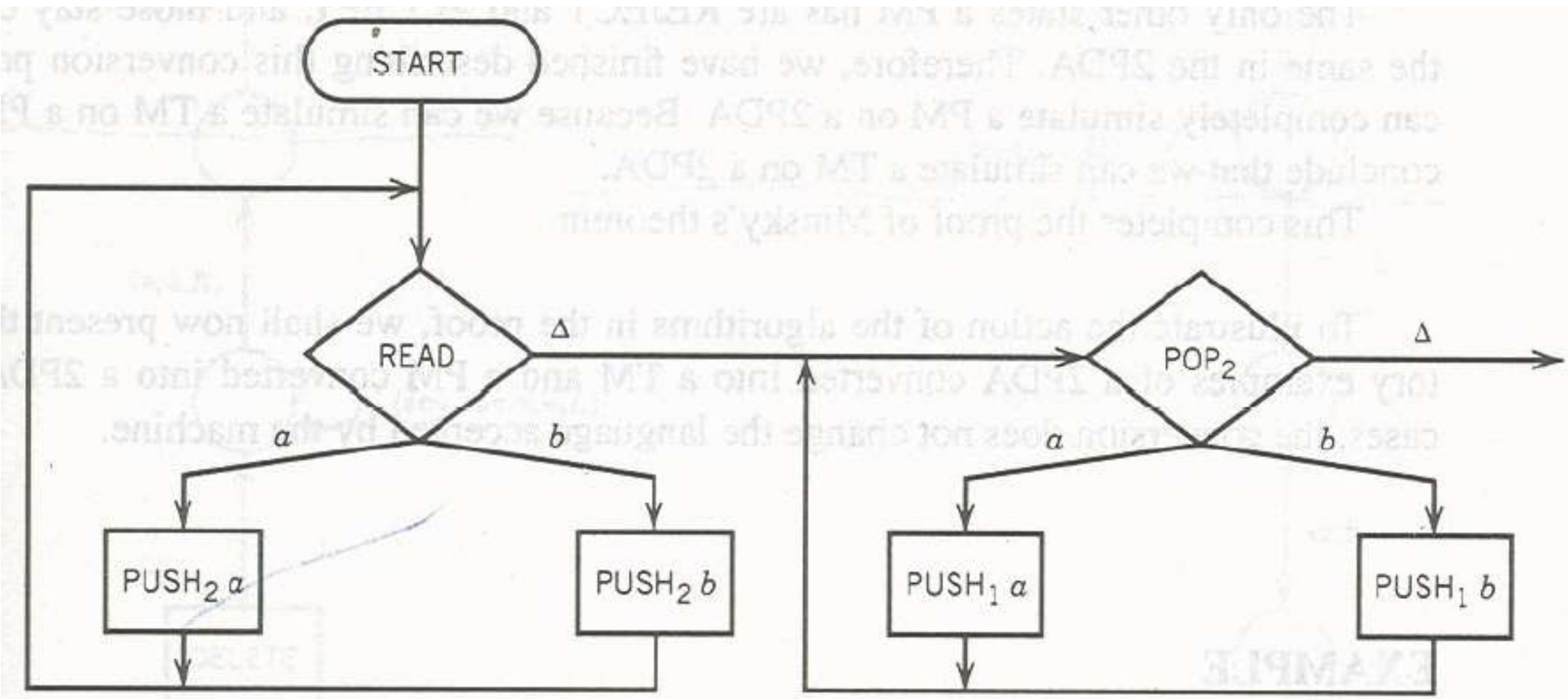
- This part will not be proved directly
- We will make a 2PDA for PM and as we have already proved that PM=TM,

So indirectly it is proven that 2PDA = TM

We will do the above step for convenience as constructing 2PDA for PM is easier as compared to constructing 2PDA for TM.

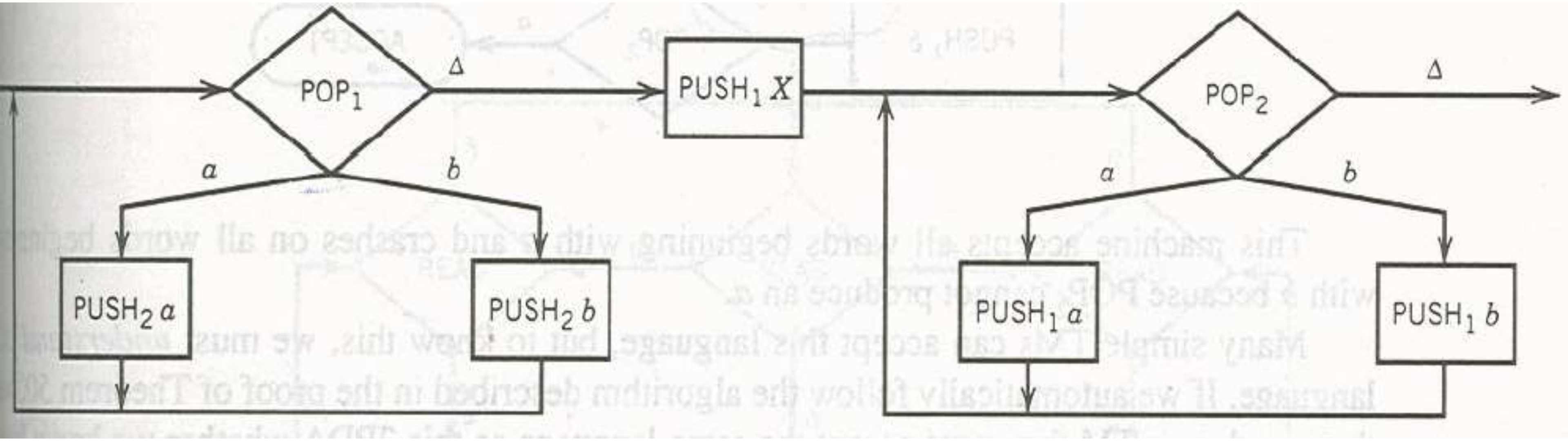
2PDA = PM

- PM starts with input string already in the STORE, so we will store the input strings over the stack1 as follows

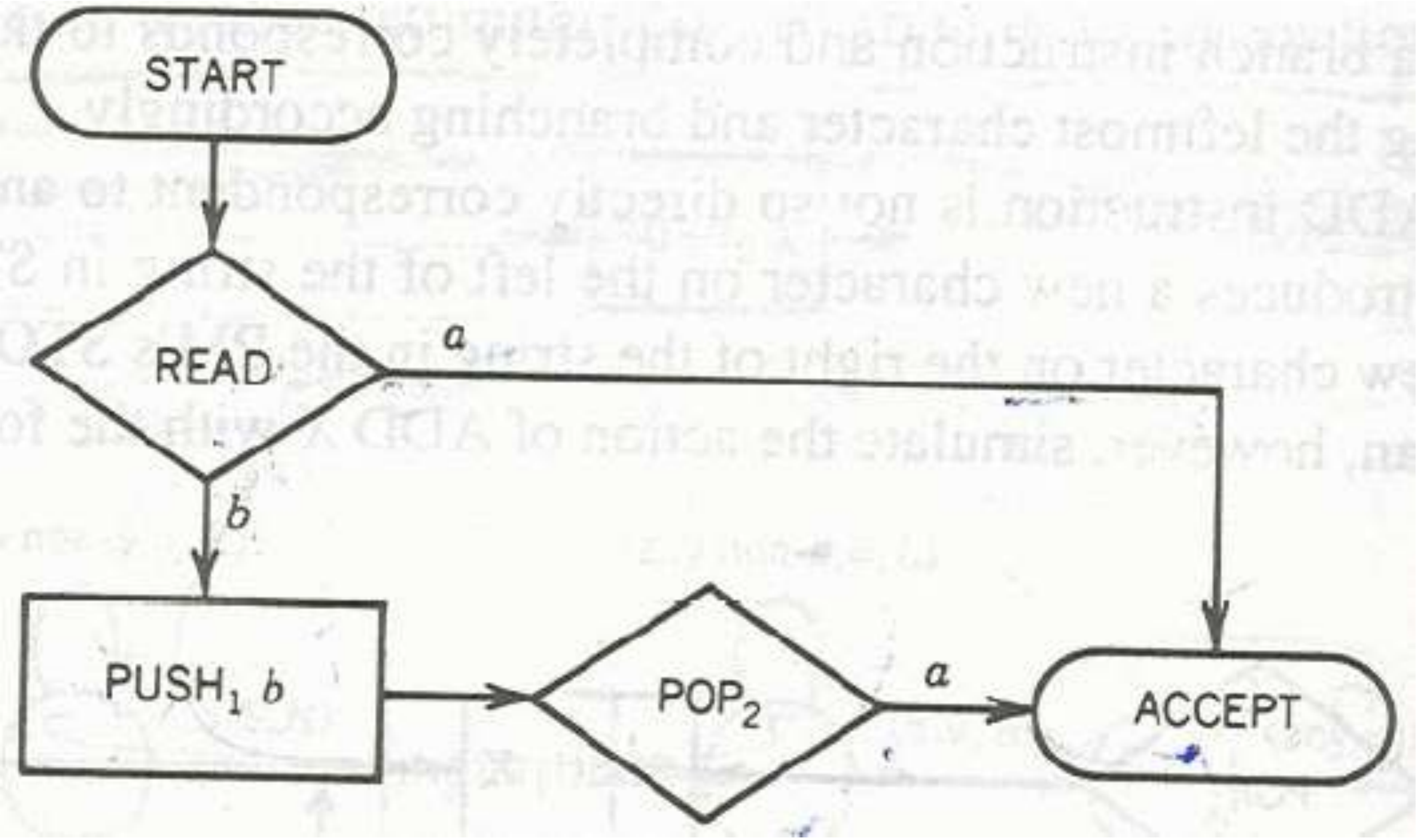


Simulating a ADD of PM

- ADD of PM will add to right of string while push of 2PDA adds to left of the string.



Construct a PM for following



nPDA = TM

Theorem 51

Any language accepted by a PDA with n STACKS (where n is 2 or more), called an **nPDA**, can also be accepted by some TM.

In power **nPDA = TM**

Theorem 51: PROOF

Suppose we have 3PDA (i.e. PDA with 3 stacks)

And the status is

TAPE $w_1w_2w_3w_4$

STACK$_1$ X_1X_2

STACK$_2$ Y_1 Y_2 Y_3 Y_4 Y_5

STACK$_3$ Z_1, Z_2Z_3

The input tape will be

Δ	Δ	Δ	$w_1w_2w_3w_4$	$\#_1$	x_1x_2	$\#_2$	$y_1y_2y_3y_4y_5$	$\#_2$	z_1z_2	Δ	…

nPDA = TM

Remaining proof is the same as for 2PDA. So

nPDA = TM for $n \le 2$

Symbolically we can represent the power comparison of our various mathematical models of machines as follows.

FA = TG = NFA < DPDA < NPDA < 2PDA = nPDA = PM = TM

Problems

2 EQUAL

3 EQUAL

MOREa

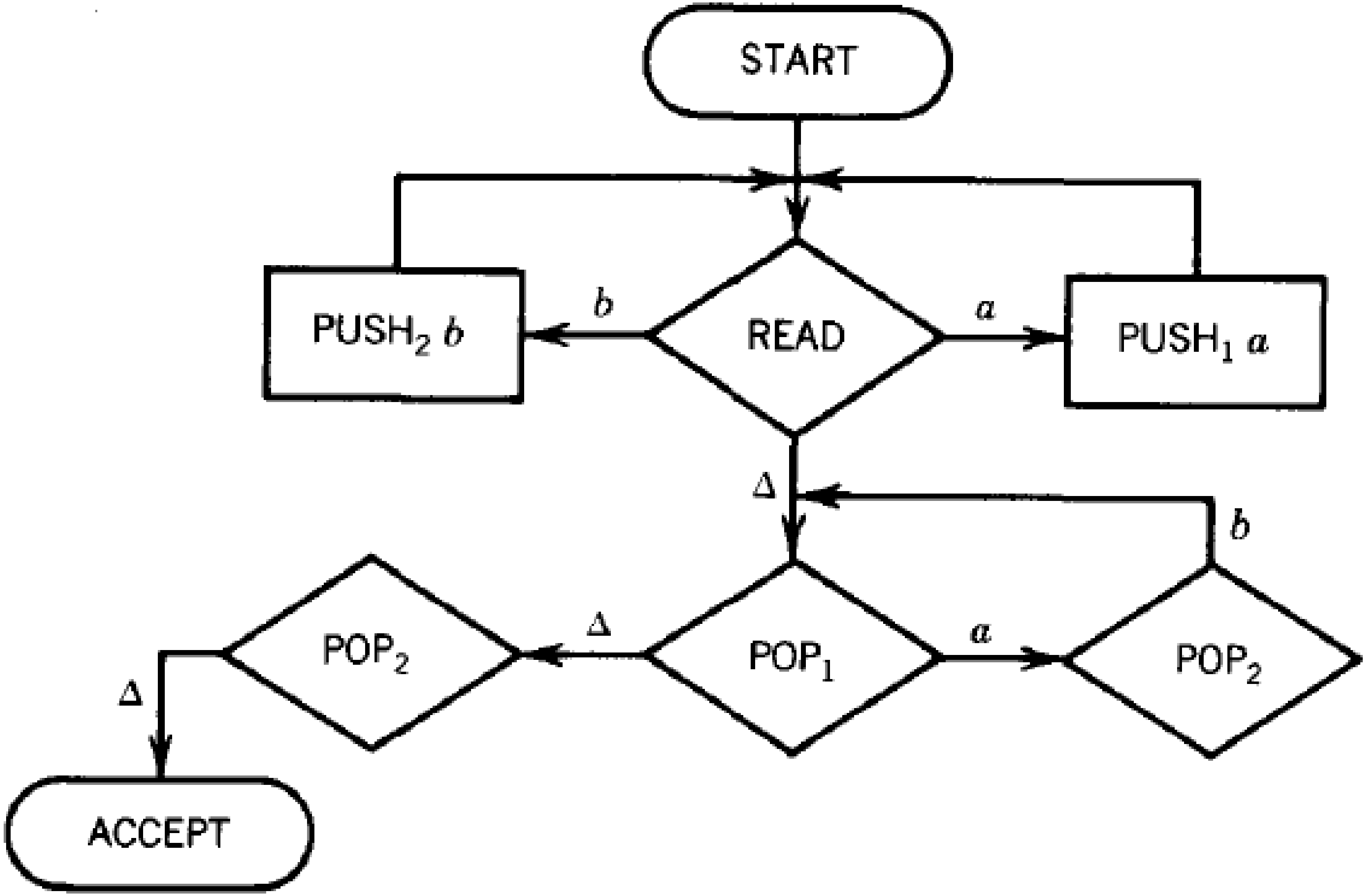
DoubleWord

$a^{n!}$

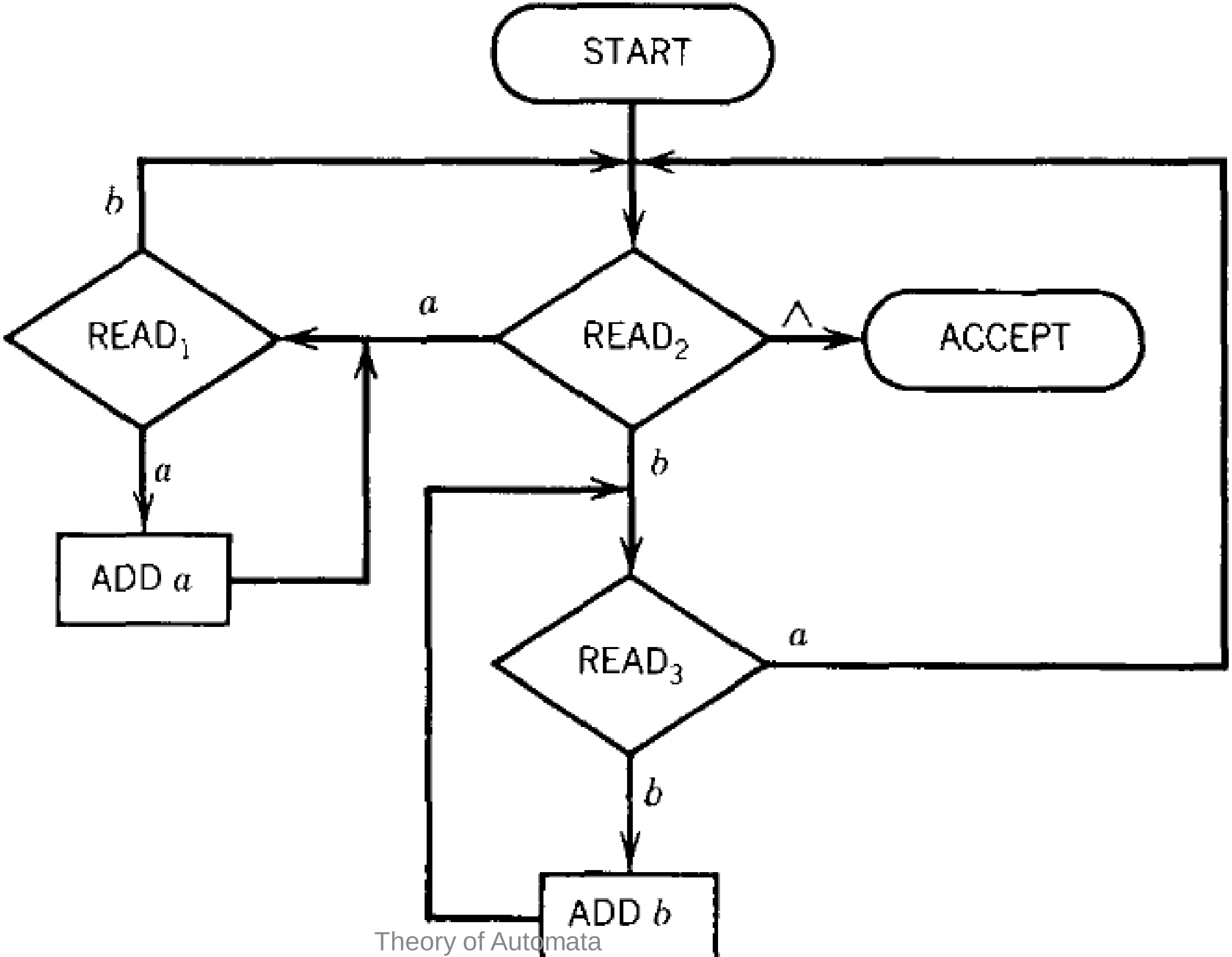
a^{2^n} hint (chop the input into half at every iteration)

Delete SubProgram

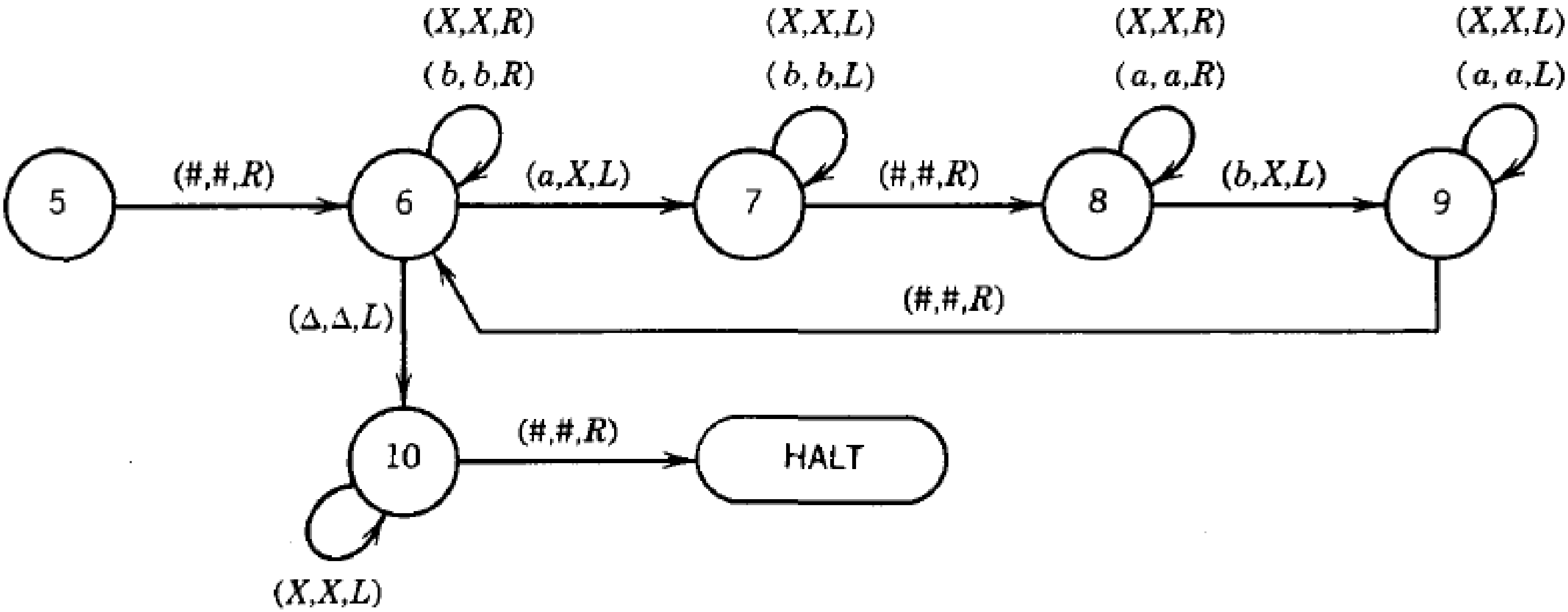
EQUAL EQUAL


START
READ
PUSH$_2$ b
PUSH$_1$ a
b
a
Δ
POP$_1$
POP$_2$
POP$_2$
b
a
Δ
Δ
ACCEPT

PM-EQUAL EQUAL



TM-EQUAL EQUAL

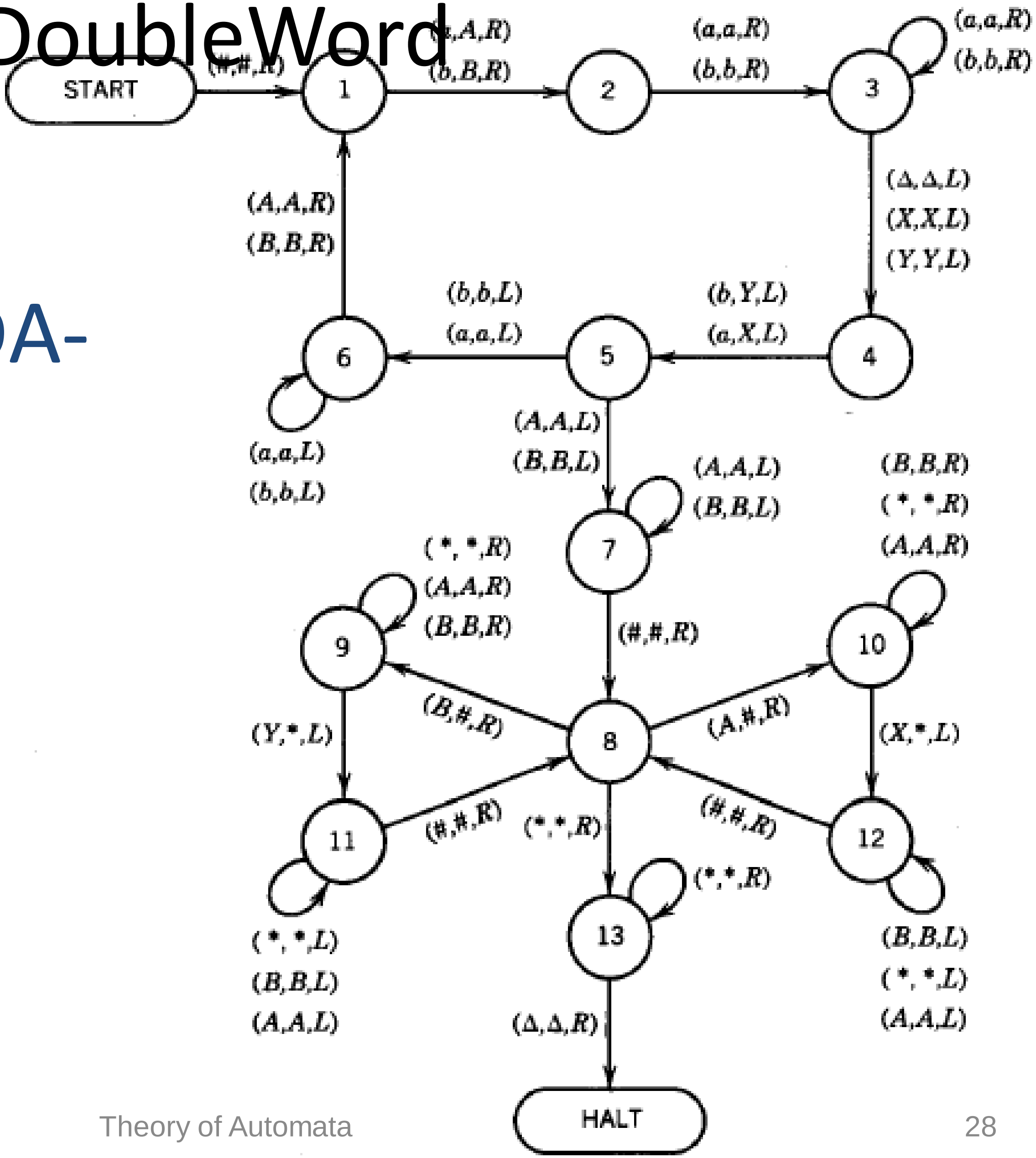


Build a TM

Language: 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 . . }

TM-DoubleWord

Build a PM-2PDA-3PDA for DoubleWord



Language: $a^n b^n c^n d^n$

- Design a PM
- Design a 2PDA
- Design a 3PDA

Language: 3EQUAL

Let us define the language 3EQUAL over the alphabet {a,b,c} as all strings that have as many total a's as total b's as total c's.

3EQUAL = *{abc acb bac bca cab cba aabbcc aabcbc* ***. . . }***

- Design a TM
- Design a PM
- Design a 2PDA
- Design a 3PDA