Theory of Computing

Assoc. Prof. Osama fathy

Turing Machines and Computability

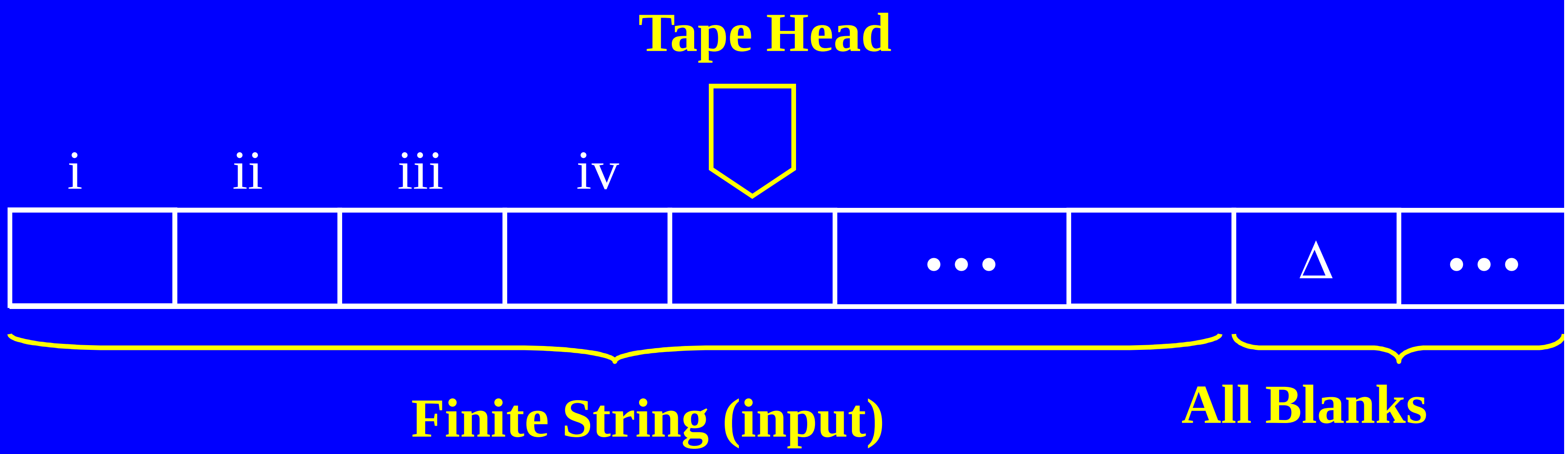
Overview

- Turing Machines
- Converting FA to TM
- Building Turing Machines
- Representing natural numbers
- Functions of one variable
- Tuples
- Functions of several variables
- Church's Thesis

TM Components

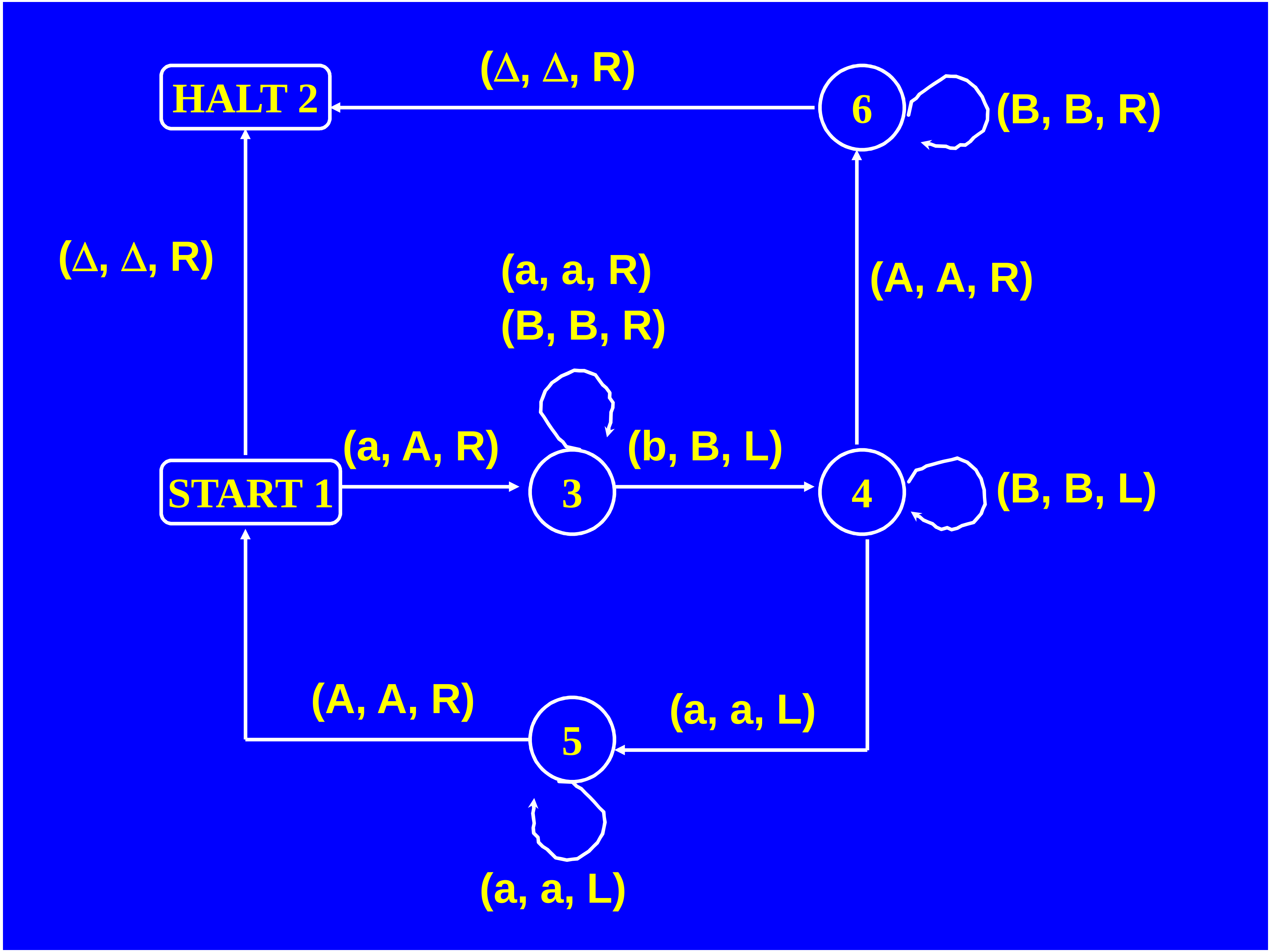
- A Tape and Tape Head
- An Input Alphabet (Σ)
- A Tape Alphabet (Γ)
- A finite set of states
  - Each state is numbered by an integer ≥ 1.
  - **Start State** (only one)
  - **Halt State** (one or more)
- A finite set of rules **(letter, letter, direction)** between states (it may be called program).

Turing Machine



Tape Head can:

- Move **left** and **right**.
- **Read letters** from the tape.
- **Write letters** onto the tape.



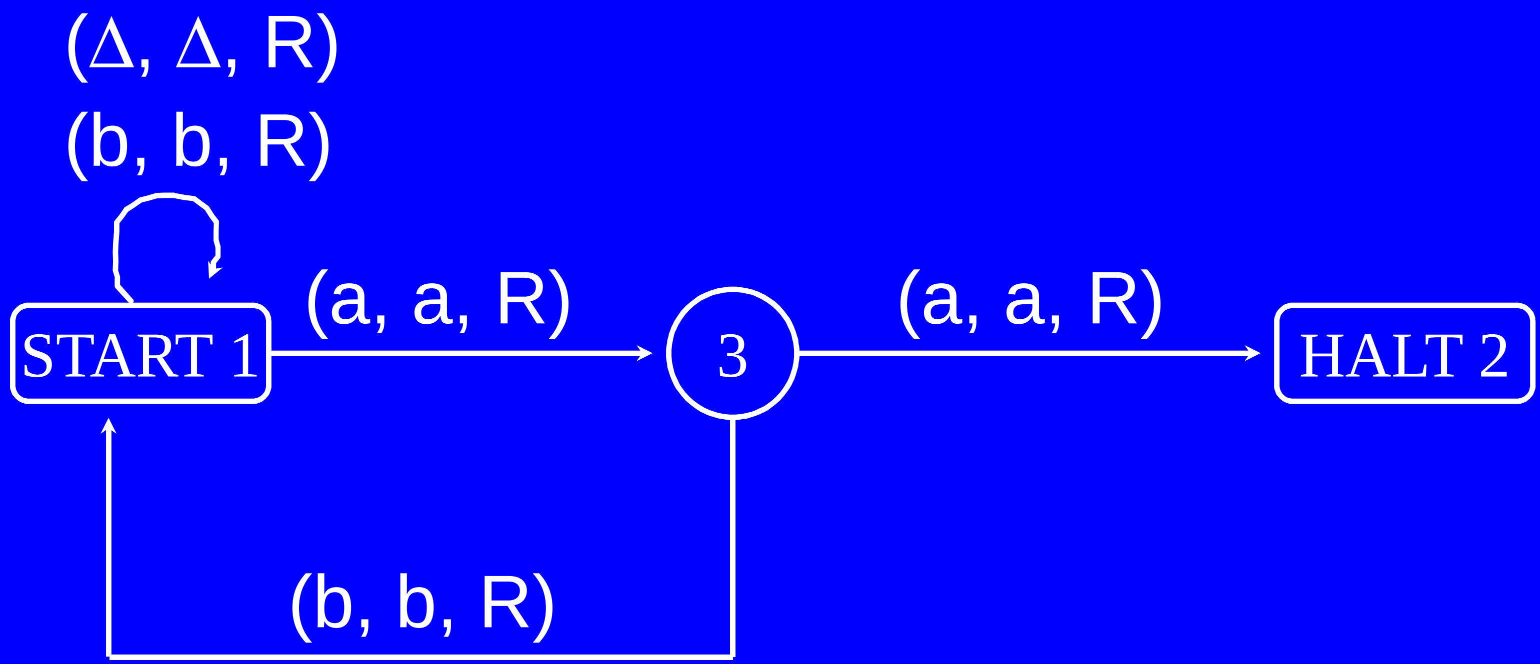
HALT 2
(Δ, Δ, R)
6
(B, B, R)
(Δ, Δ, R)
(a, a, R)
(B, B, R)
(A, A, R)
(a, A, R)
(b, B, L)
START 1
3
4
(B, B, L)
(A, A, R)
5
(a, a, L)
(a, a, L)

Definitions

For a Turing Machine T

- **Accept(T)**
  - The set of strings leading to a **HALT** state.
  - Called the **language accepted by T.**
- **Reject(T)**
  - The set of strings that crash during execution.
- **Loop(T)**
  - The set of strings that loop forever.

Example



- **Accept(T)** = strings with a double **aa**
- **Reject(T)** = strings without a double **aa** that end in **a**
- **Loop(T)** = **L** or strings without a double **aa** that end in **b**

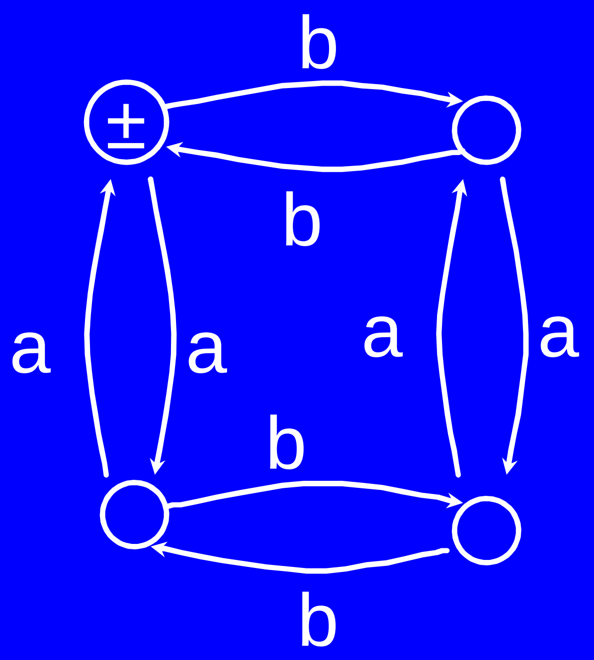
Regular Languages

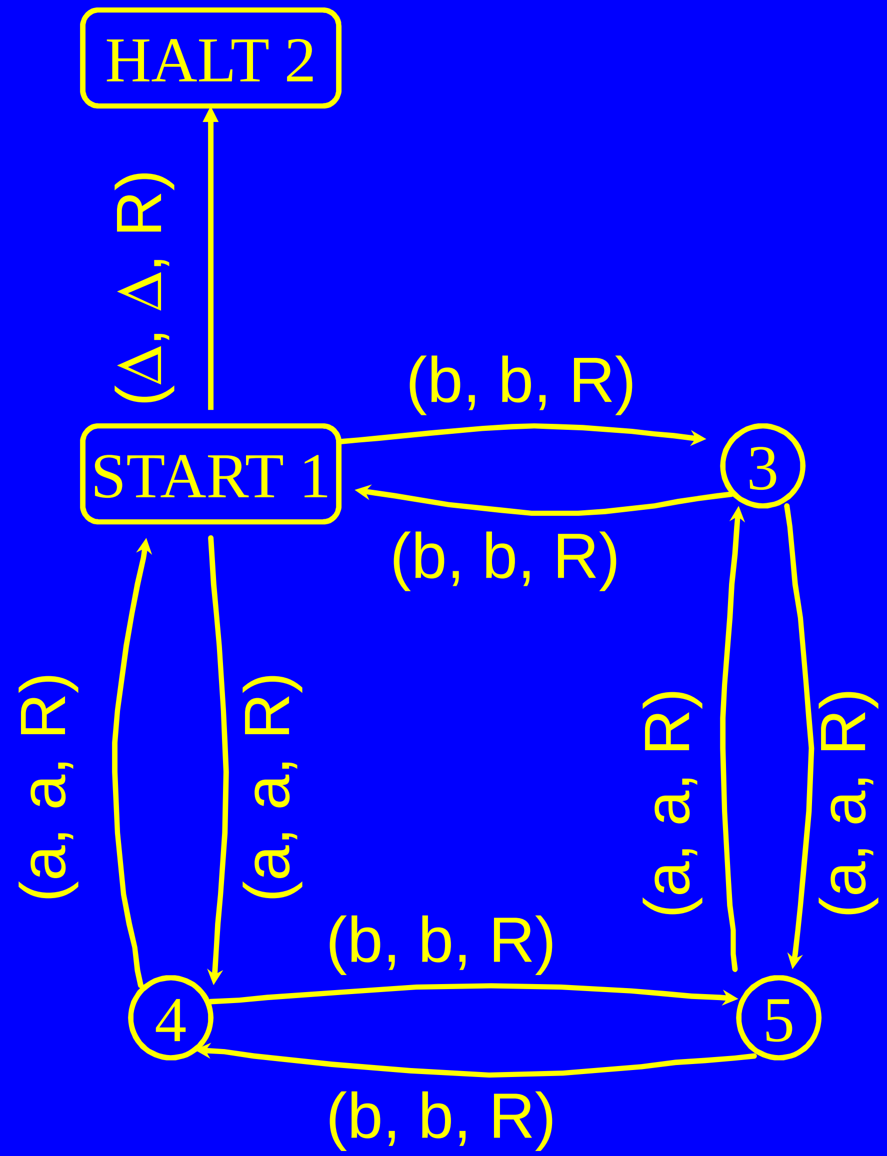
Every **Regular Language** can be accepted by a **Turing Machine**.

Convert FA to TM

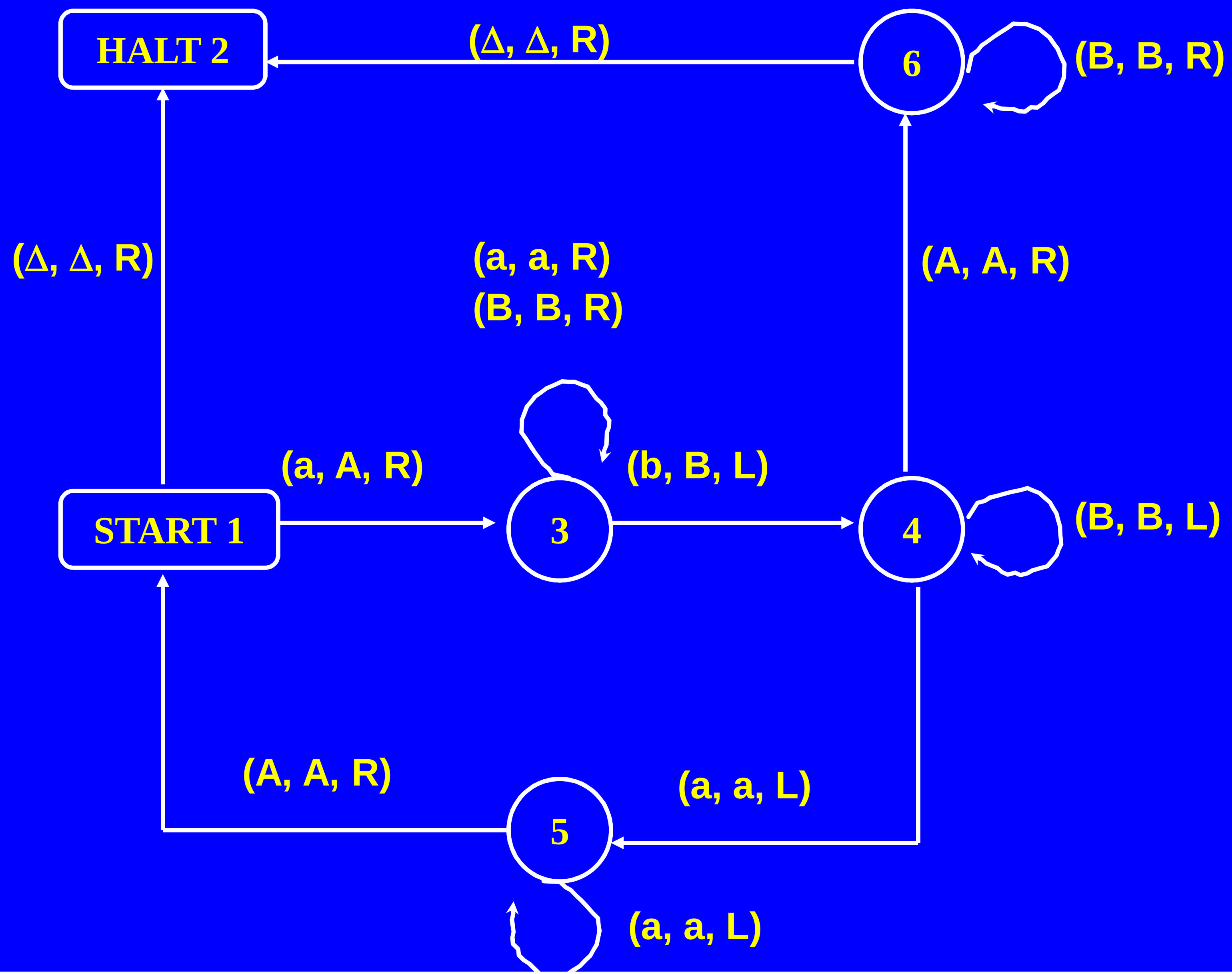
- Change the **-** to **START 1**.
- Label all other states with a integer ≥ 3.
- Change the edge labels.
  - **a** to **(a, a, R)**
  - **b** to **(b, b, R)**
- Delete the **+** from all the Final states, and add an edge from each Final state to **HALT 2**, labeled with **(Δ , Δ , R)**.

Problem: Convert the shown FA, that accepts the language **EVEN-EVEN** into the equivalent Turing Machine



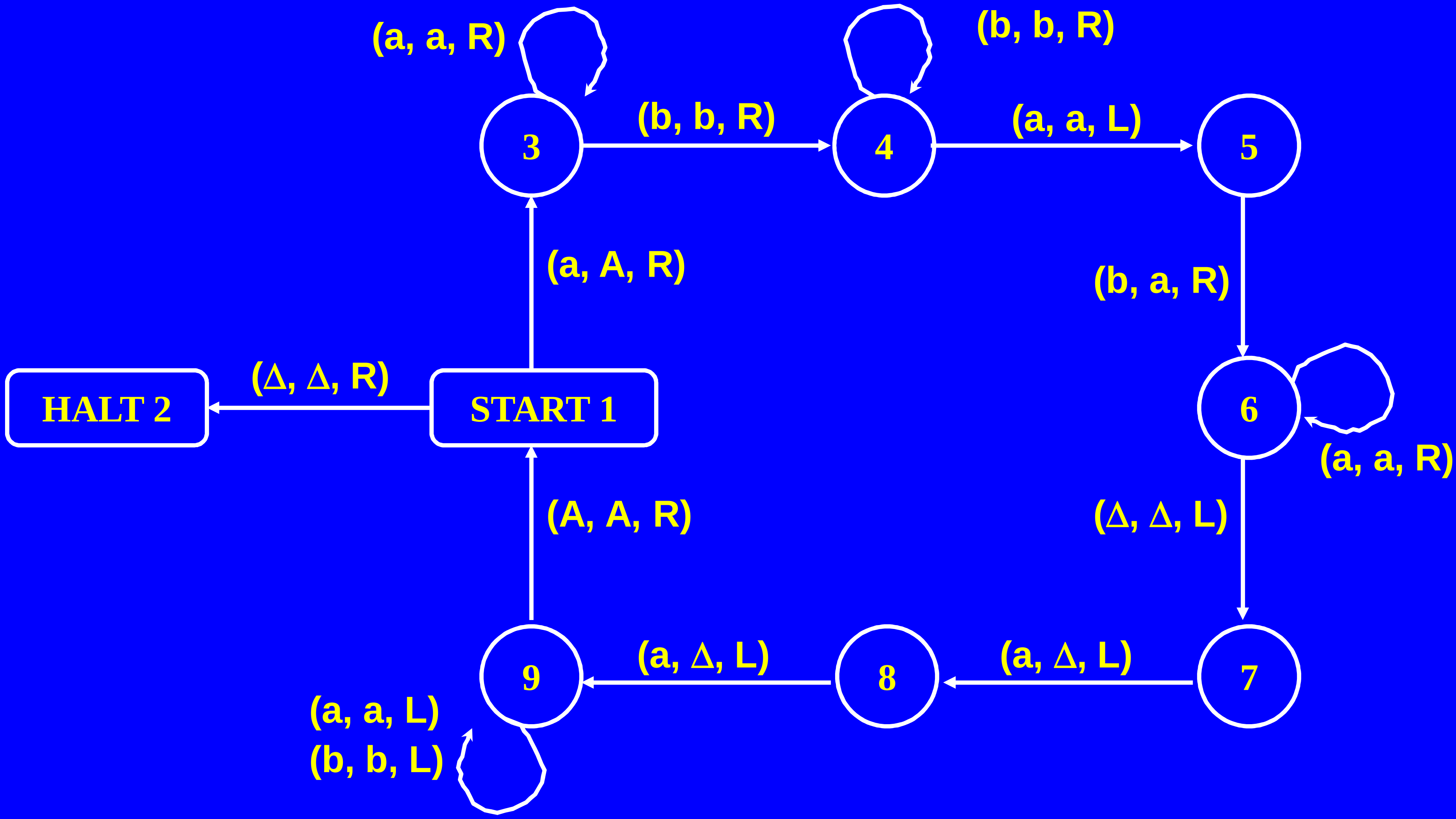


Problem: Build a Turing Machine that accepts the language $\{a^n b^n : n \geq 0\}$.



Problem: Build a Turing Machine that accepts the language

$\{a^n b^n a^n : n \geq 0\}$.



Representing natural numbers

Unary Code

0	Δ	Δ
1	a	a
2	aa	a^2
3	aaa	a^3
4	aaaa	a^4
5	aaaaa	a^5
6	aaaaaa	a^6
7	aaaaaaa	a^7
:	:	:

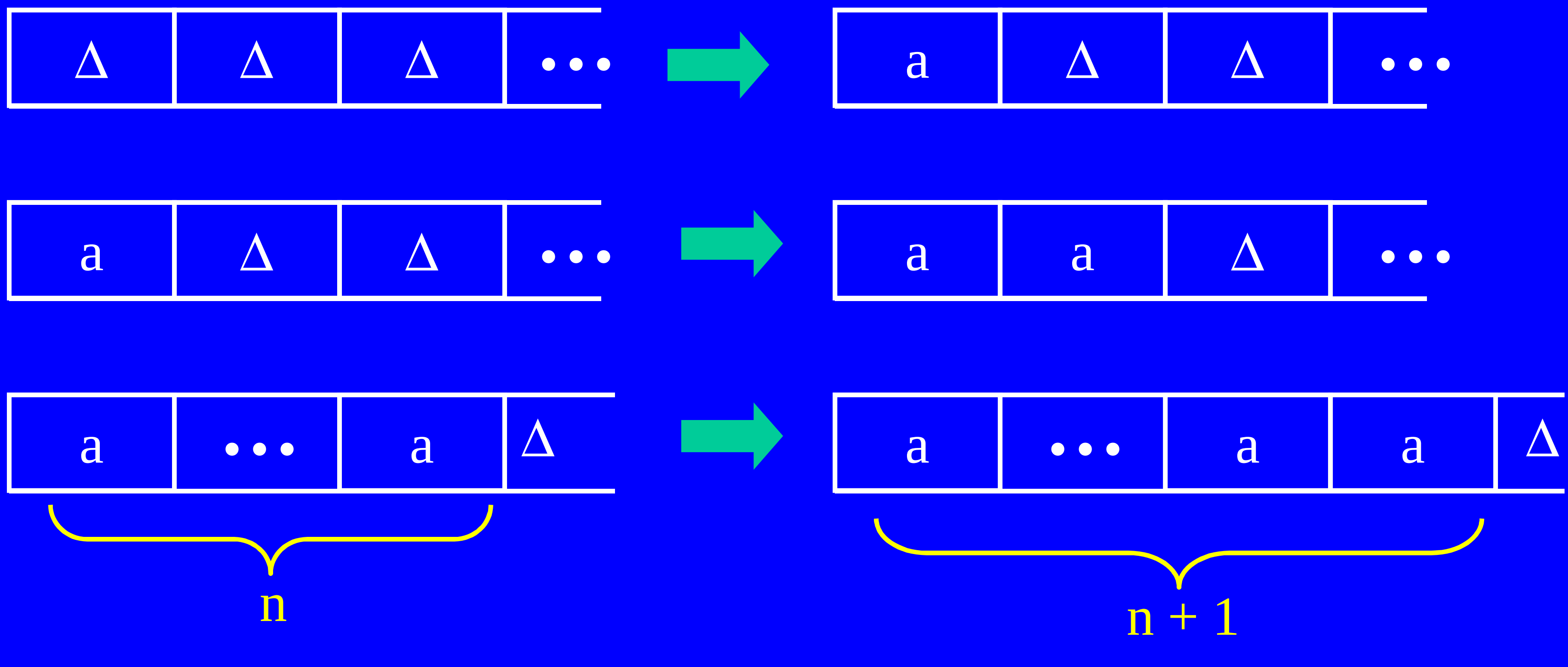
Binary Code

0	a
1	b
2	ba
3	bb
4	baa
5	bab
6	bba
7	bbb
:	:

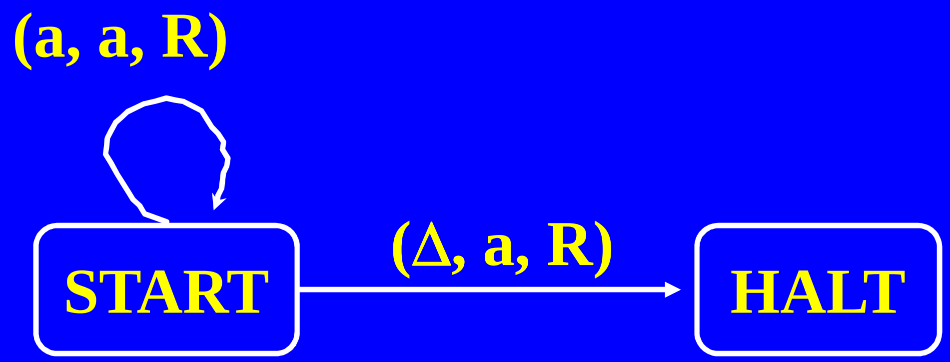
Representing functions of one variable by using TM

Successor

Using the unary code for natural numbers build a Turing Machine that represents the function **f(n) = n + 1**.



Successor



Unary Code for Tuples of Integers

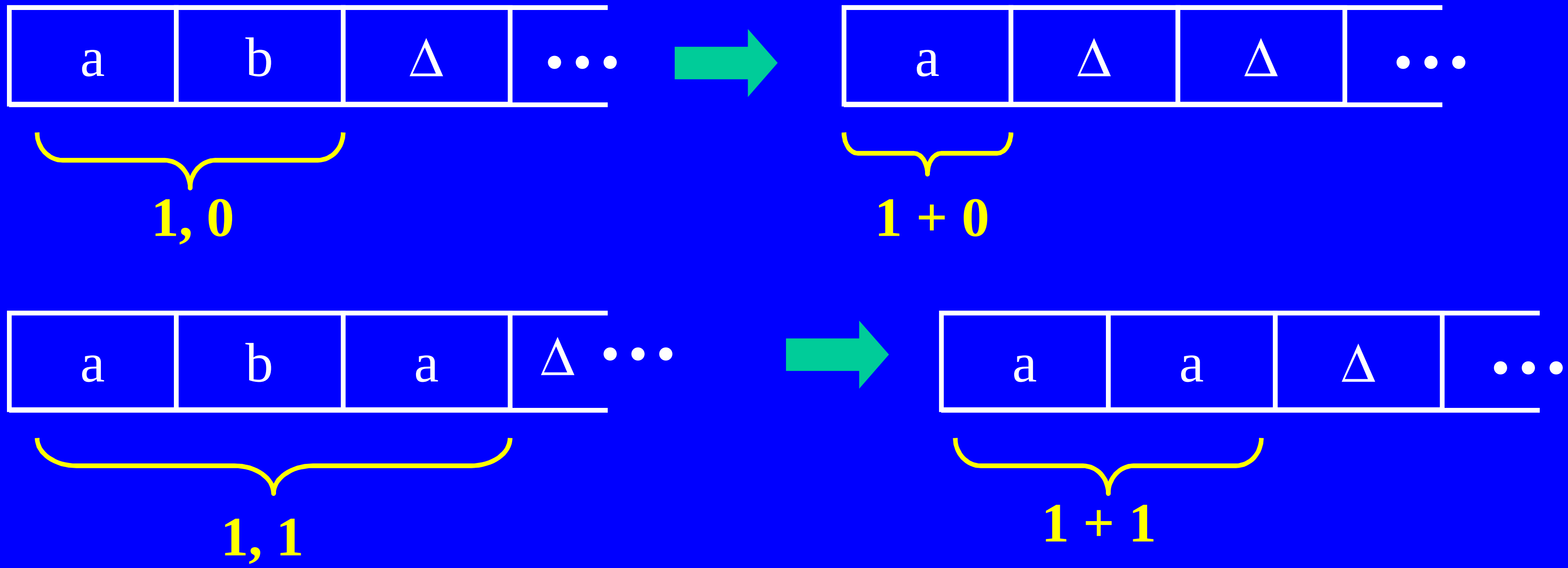
- Tuples of natural numbers
- Example: **1, 0, 2, 3**
- Encoding:
  - Each integer is coded using the unary code as a string of **a**'s
  - Integers are separated by a **b**.

- Example: **abbaabaaa**

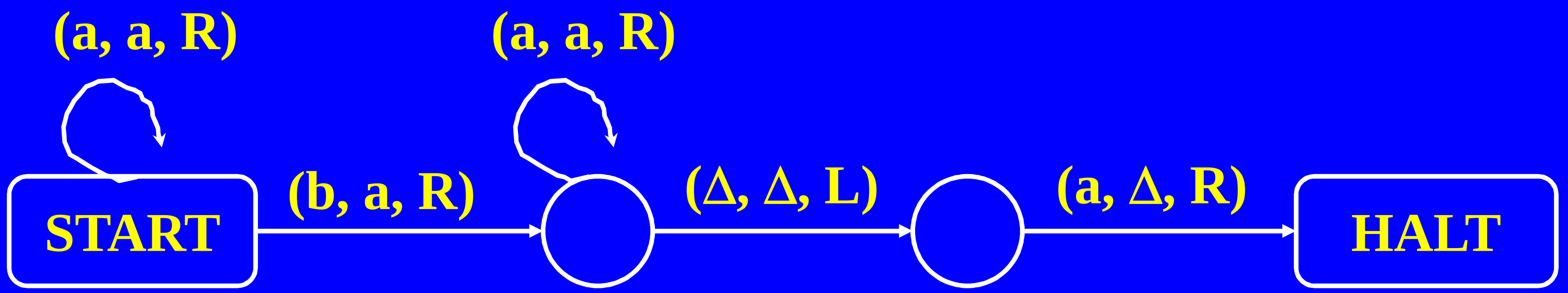
Addition

Using the unary code of natural numbers build a Turing Machine that represents the function

f(n, m) = n+m.



Addition



Definition

A **computable function** can be represented as:

- A Turing Machine
- Input
  - sequences of natural numbers
- Output
  - one natural number

Church's Thesis

Any function which can defined by an algorithm is a computable function.

Church's Thesis

Any function which can defined by an algorithm is a computable function.