Universal Turing Machine

Ertug Umsur & Joshua Espinoza

Introduction to Turing Machine

Key Concepts:

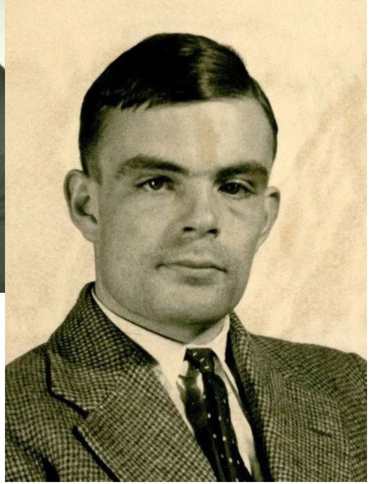
- **Transition Function:** Dictates what happens based on the current state and symbol
- **Tape:** Infinite memory tape divided into cells
- **Head:** Reads and writes symbols on the tape
- **State:** The Machine's current status

Konrad Zuse's Z3

History of the Turing Machine

First Mentioned In: Turing's 1936 paper "On Computable Numbers, with an Application to the Entscheidungsproblem

Church-Turing Thesis: Postulates that anything computable by an algorithm can be computable by Turing Machines

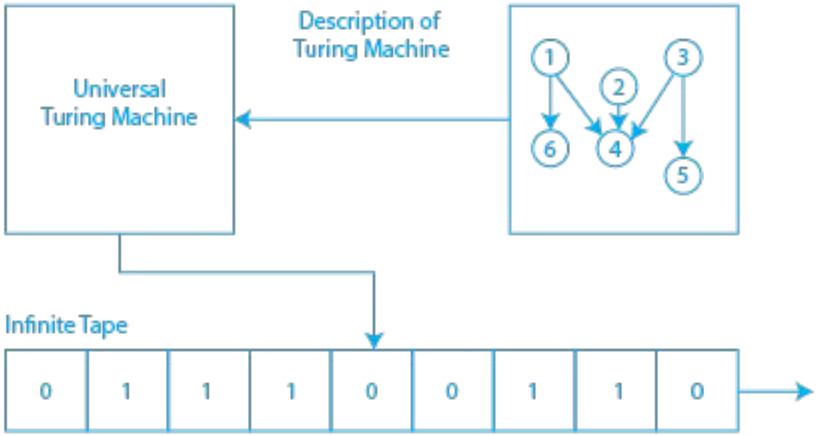


Alan Turing & Alonzo Church

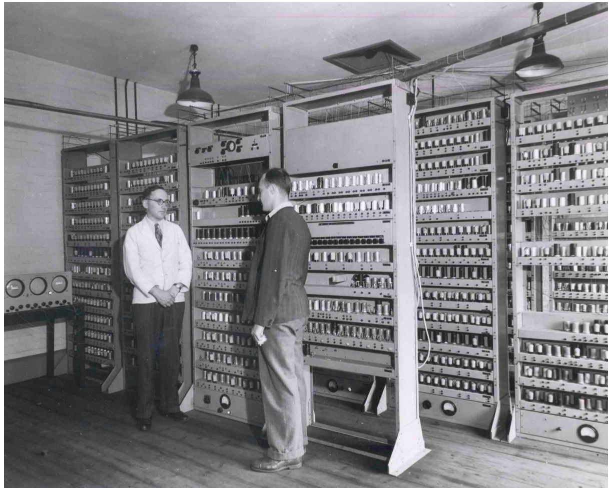
The Universal Turing Machine

A **Universal Turing Machine** is a special type of Turing Machine that can simulate any other Turing Machine.

Normal Turing Machines are designed to perform one specific task but **UTMs** can perform any Turing Machine's tasks



The First Universal Turing Machine



- Mentioned in the same 1936 paper by Turing
- The first physical copies were made in 1950s and 60s
- In US, John von Neumann's contributions created ENIAC and EDVAC. US's first programmable computers.

Importance of Universal Turing Machines

Universal Turing Machines demonstrate the universality of the Turing Machines and it lays the foundation of **modern computers**.

Introduces the idea that all computations can be **reduced to same basic operations**, regardless of the specific algorithms.

Introduced the concept of undecidability and the **Halting Problem**

Our Universal Turing Machine

Part I: The Encoding Scheme

The Encoding Scheme

Our encoding [M] for a TM M with input string w

$[M] = \langle \text{current state}/\text{Buffer} \rangle\, 0000 \langle M \rangle\, 0000 \langle w \rangle$

$M = \{Q, \Sigma, \Gamma, \delta, q_{start}, q_{acc}, q_{rej}\}$

$Q = \{1, 2, 3, \ldots$ (1 = start state, 2 = accept state, 3 = reject state)

$\langle Q \rangle = \underbrace{1\,0}_{q_s}\underbrace{11\,0}_{q_{acc}}\underbrace{111\,0}_{q_{rej}} \ldots$

ends with the final state (no final 0)

$\Sigma = \{a, b, c \ldots$ ($a \to 1$, $b \to 2$, $c \to 3$)

$\langle \Sigma \rangle = \underbrace{1\,0}_{a}\underbrace{11\,0}_{b}\underbrace{111\,0}_{c} \ldots$

$\Gamma = \{\Sigma, x, y, \ldots$

$\langle \Gamma \rangle = \langle \Sigma \rangle\, 0\, \underbrace{111 \ldots 1\, 0}_{x}\, \underbrace{11 \ldots 1\, 0}_{y} \ldots$

Machine alphabet first, then the letters of the tape alphabet

$\delta = \{\underbrace{p, a, q, b, \pm 1}_{d_1}\} \ldots$ (more transitions)

$d_n = \{p, a, q, b, \pm 1\}$

$\langle d_n \rangle = \underbrace{1 \cdots 1}_{\substack{\text{encoding of state } p \\ \text{(from } Q)}}\, 0\, \underbrace{1 \cdots 1}_{\substack{\text{encoding of letter } a \\ \text{(from } \Gamma)}}\, 0\, \underbrace{1 \cdots 1}_{\substack{\text{encoding of state } q \\ \text{(from } Q)}}\, 0\, \underbrace{1 \cdots 1}_{\substack{\text{encoding of letter } b \\ \text{(from } \Gamma)}}\, 0\, \underbrace{11}_{\text{Move pointer } \pm 1}$ (or 1)

1 for +1

11 for −1

$\langle \delta \rangle = \langle d_1 \rangle 00 \langle d_2 \rangle 00 \cdots 00 \langle d_n \rangle$

$\{q_{start} = 1 \quad \langle q_{start} \rangle = 1$

$\{q_{acc} = 2 \quad \langle q_{acc} \rangle = 11$

$\{q_{rej} = 3 \quad \langle q_{rej} \rangle = 111$

Final encoding of $\langle M \rangle$

$\langle M \rangle = \langle Q \rangle\, 00 \langle \Sigma \rangle\, 00 \langle \Gamma \rangle\, 000 \langle \delta \rangle\, 000 \langle q_{start} \rangle\, 00 \langle q_{acc} \rangle\, 00 \langle q_{rej} \rangle$

Demonstration

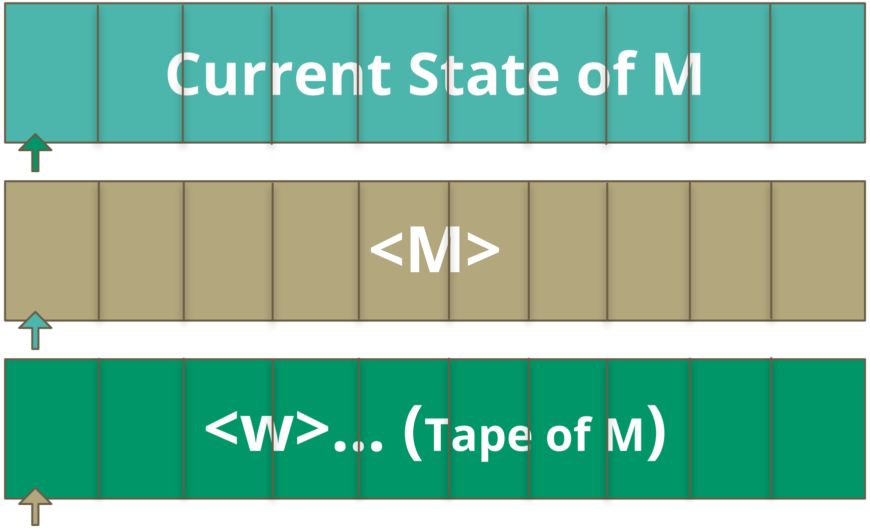
How Does It Work?

UTM Overview

Base scenario: 3-Tape UTM.

Given a Turing Machine M, develop a UTM to simulate it using 3 tapes.

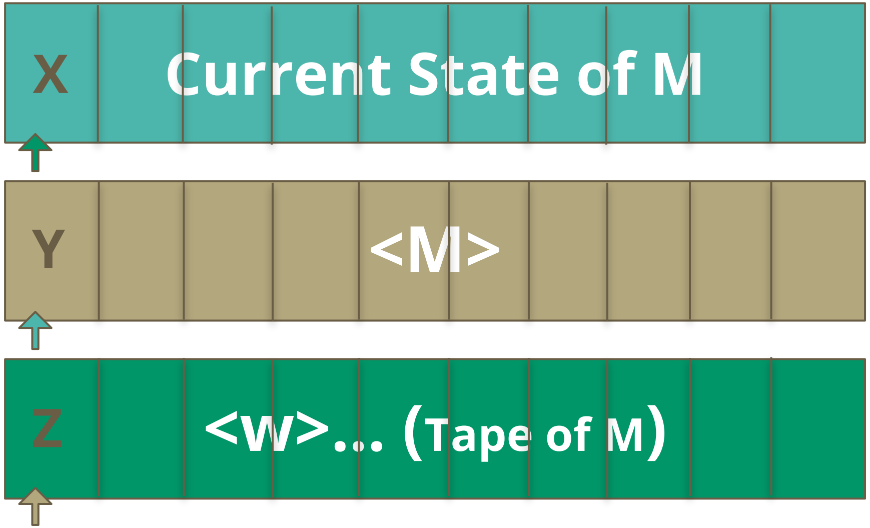
In a 3-Tape UTM, **tape 1** is allocated for the **current state of M**. **Tape 2** is for the TM **M encoding** (**<M>**). **Tape 3** is for the string **w** (**also the tape of M**).



3-Tape to 1 Tape UTM transformation

a) The **tape heads** will be **replaced** with special tape symbols

The symbols **X, Y** and **Z** of the tape alphabet for the 1 tape UTM will serve as the tape head of the three tapes.

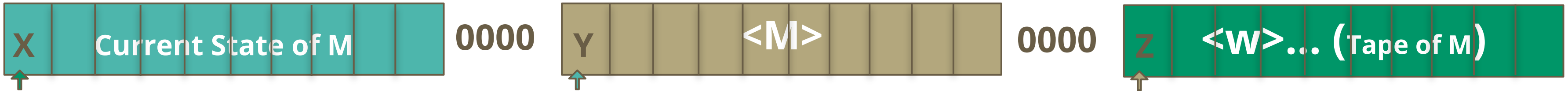


* We will use the word pointer to refer to these symbols.

3-Tape to 1 Tape UTM transformation

b) The three tapes will be **concatenated** and a series of 0000 will be added to separate each tape.

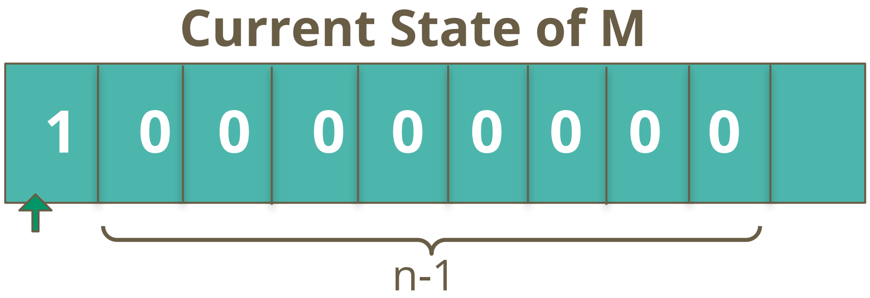
This is a process was performed in the encoding of M



3-Tape to 1 Tape UTM transformation

c) For simplicity, the tape 1 will start with a 1 and a sequence of n-1 0's.

Where n is the state with the greatest value for the encoding. This will allow the tape 1 to hold any state without requiring to move the contents of the UTM tape.



* For this reason, we will call tape 1 also as the buffer to hold the current state of M.

Gadgets to build the UTM

We developed small portions of a TM to simplify the process or implementing the 1-Tape UTM

Definitions:

A,B: Pointers (the actions apply to the sequence of 1's next to them.

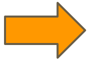
a,b: Symbols (the actions apply to the symbol itself).

P: Placeholder symbol

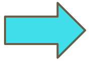
C: Any other symbol that is not A, B, P, 1 or 0

Symbology:

End state of a gadget that can be connected to the rest of the TM



Normal connections with a different color to differentiate when multiple lines cross

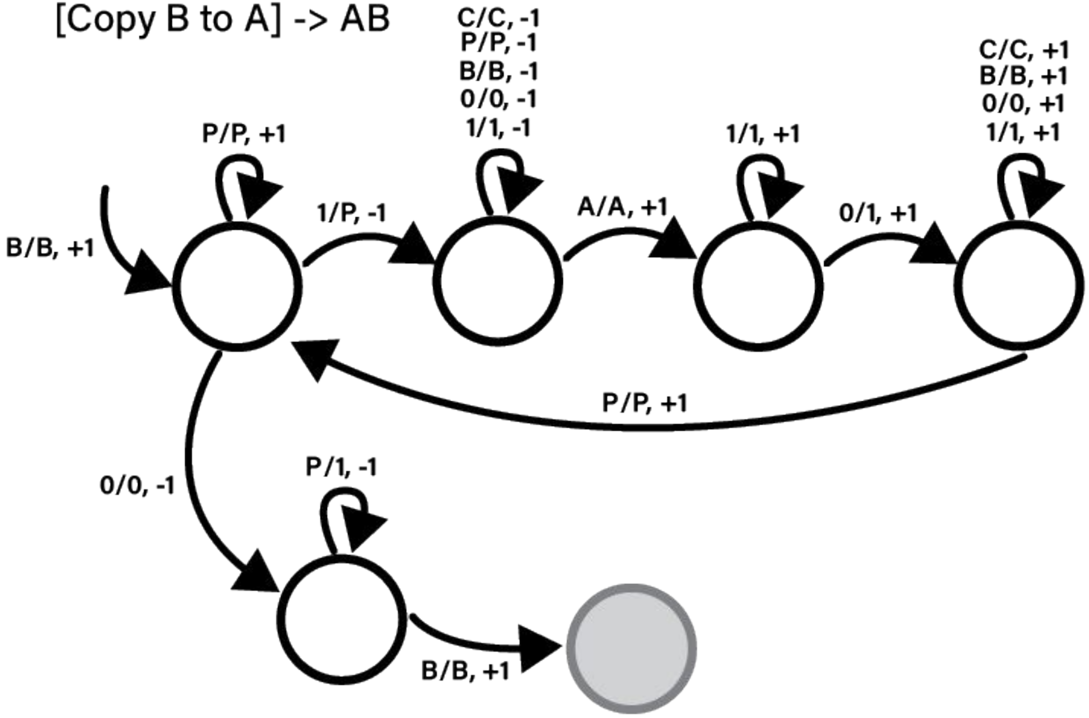


Expanded form of the gadget

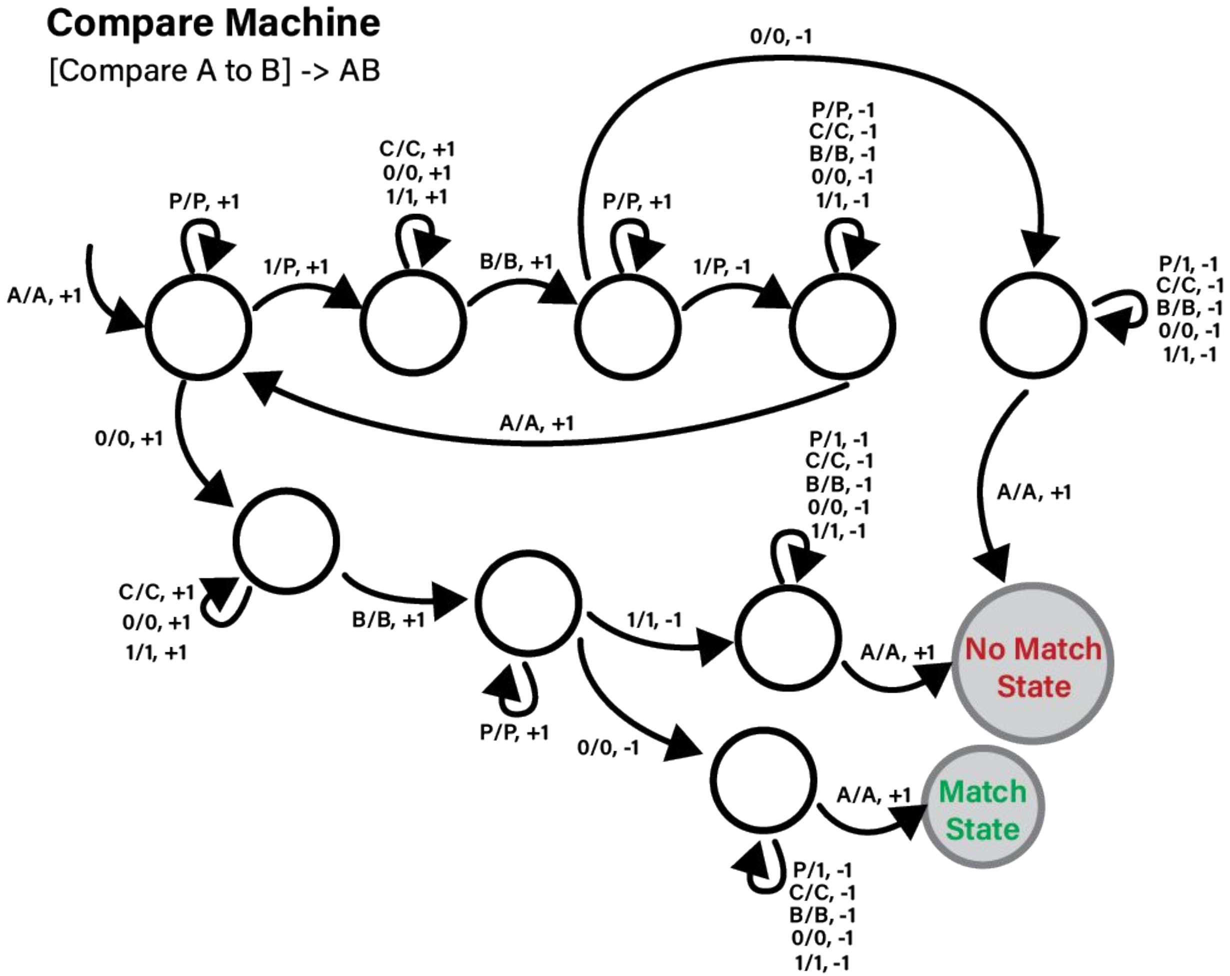
Gadgets

Copy Machine

[Copy B to A] -> AB



* Used to copy the elements of a pointer B to the pointer A (Assumes that A has enough 0's to allocate B)

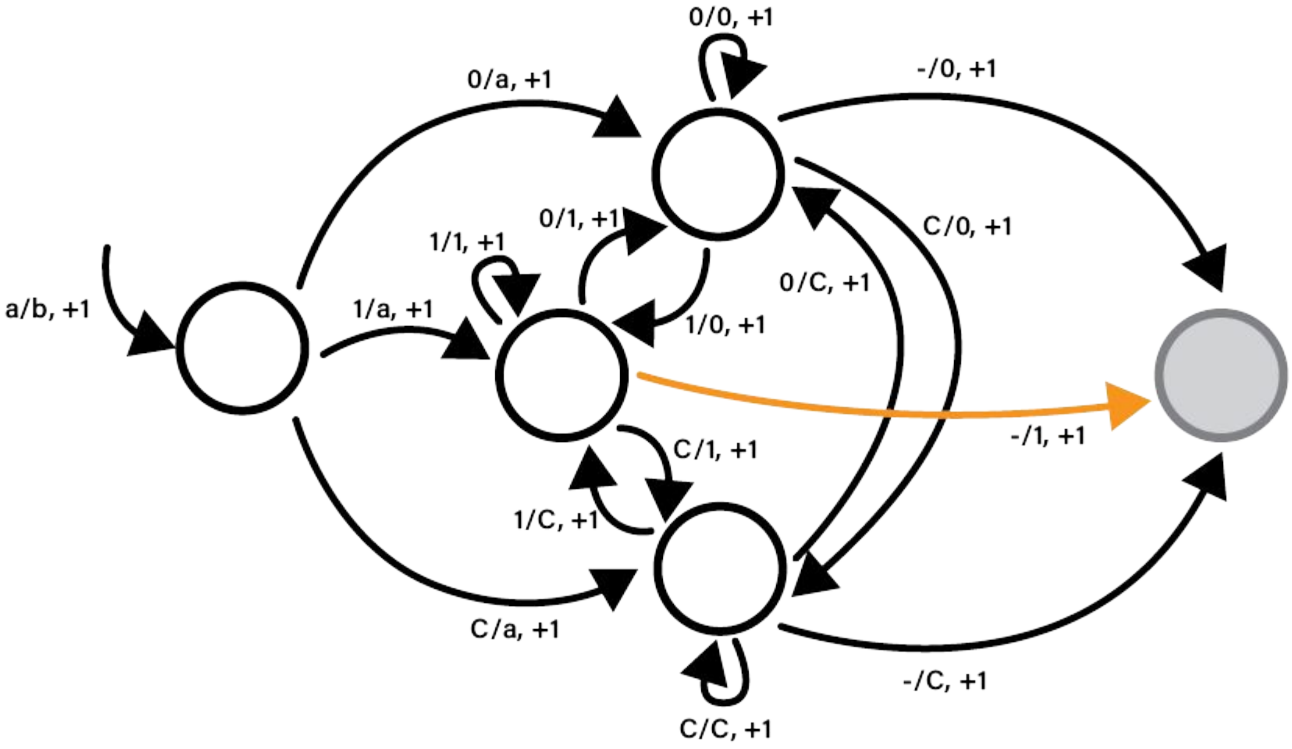


Gadgets

* Compare the content of pointer A with pointer B

Gadgets

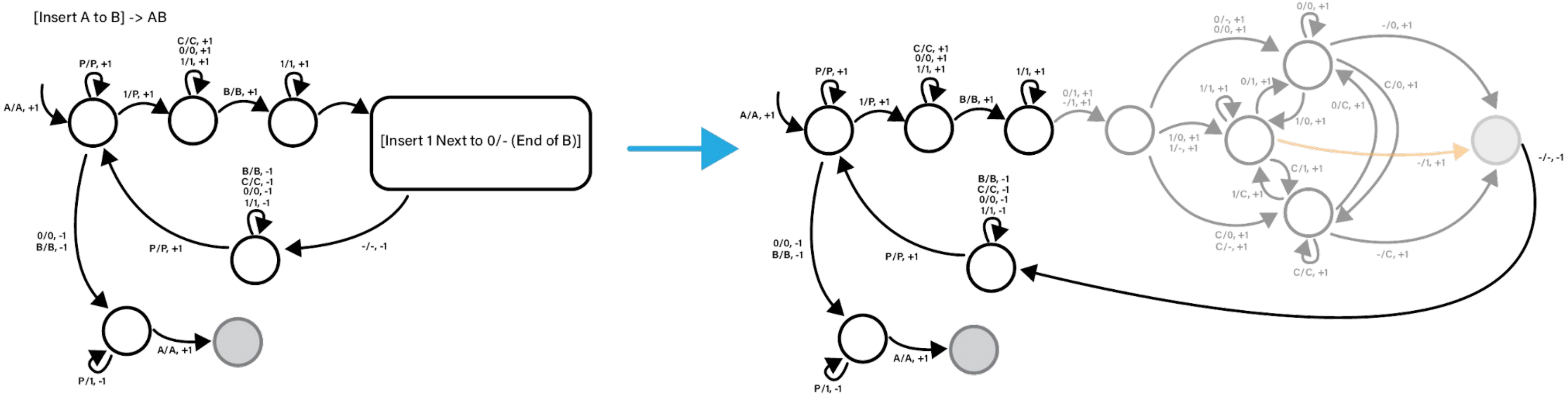
[Insert symbol b next to a] -> ba, pointer starts on a

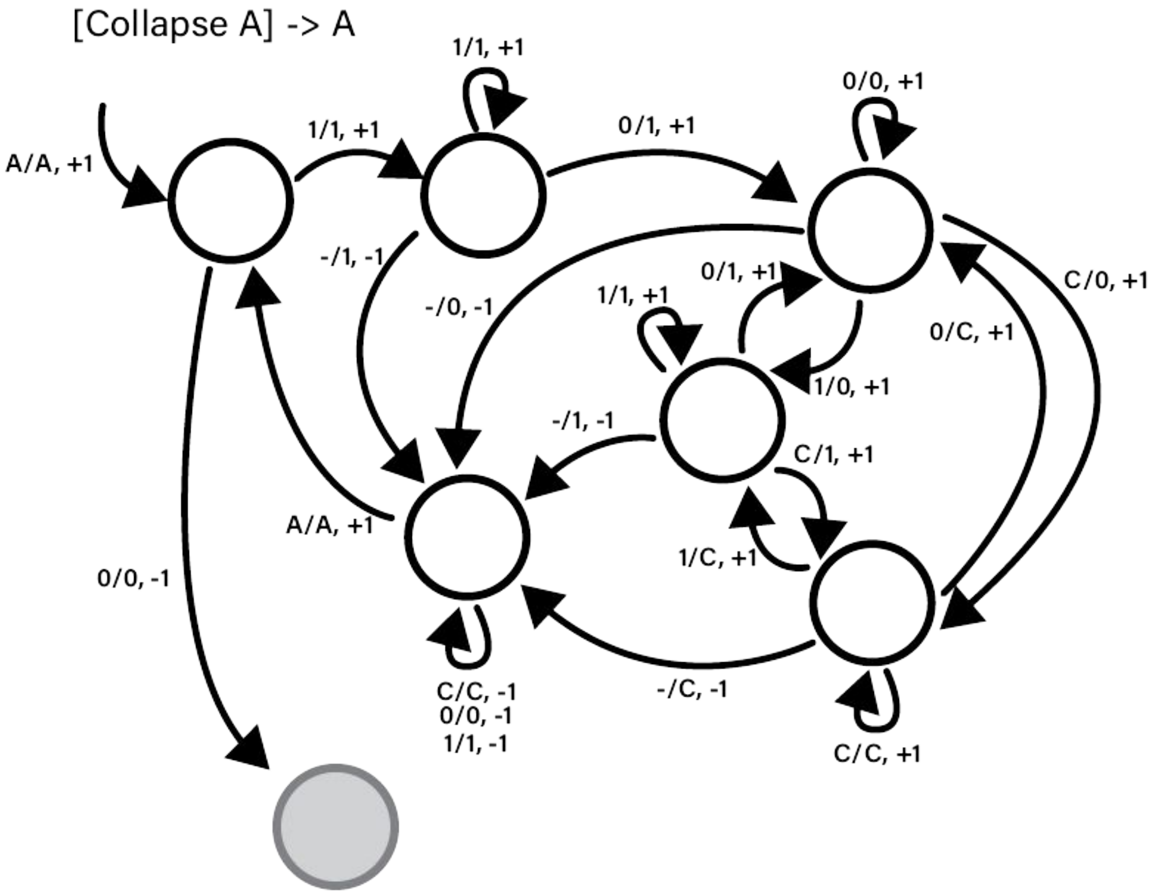


* Inserts symbol b into the tape at the position a, then moves the rest of the tape 1 cell to the right.

Gadgets

Inserts the contents of pointer A into pointer B





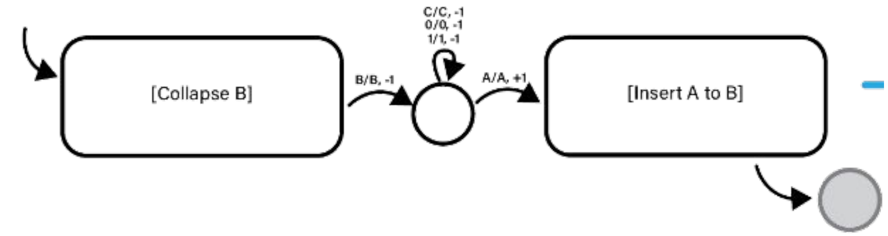
Gadgets

*** Deletes the string of 1's next to pointer A (If there is one)**

Gadgets

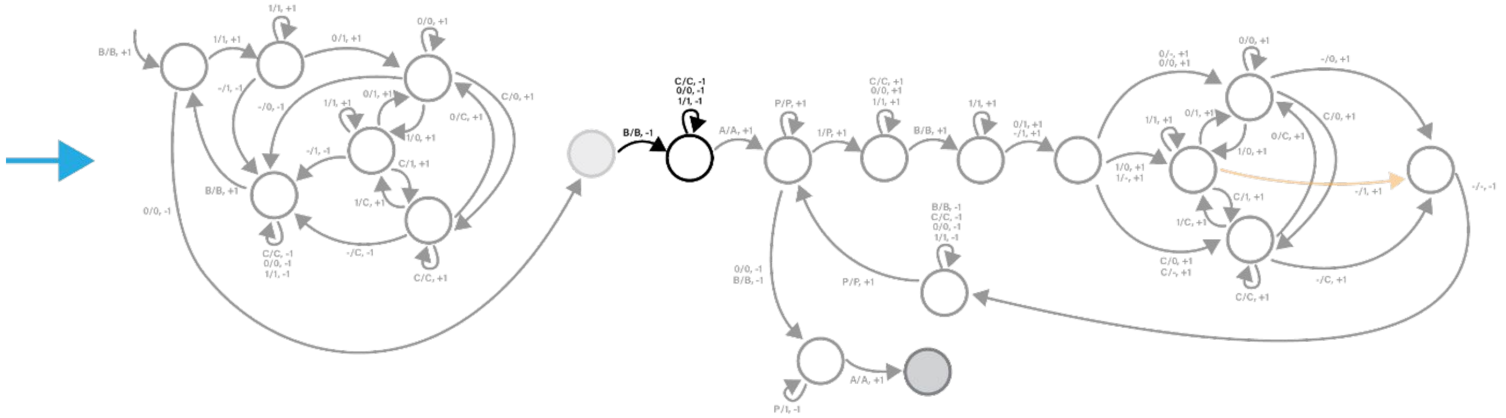
Replace Machine

[Replace B for A] -> AB, pointer starts on B



*** Replaces the content of pointer B with the content in pointer A.**

First it collapses pointer B and then inserts the content of pointer A into pointer B

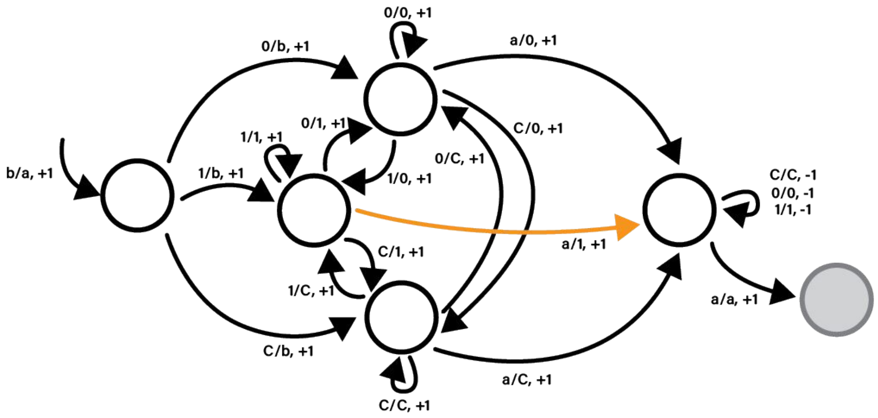


Gadgets

Moves the pointer (symbol) to a designated position

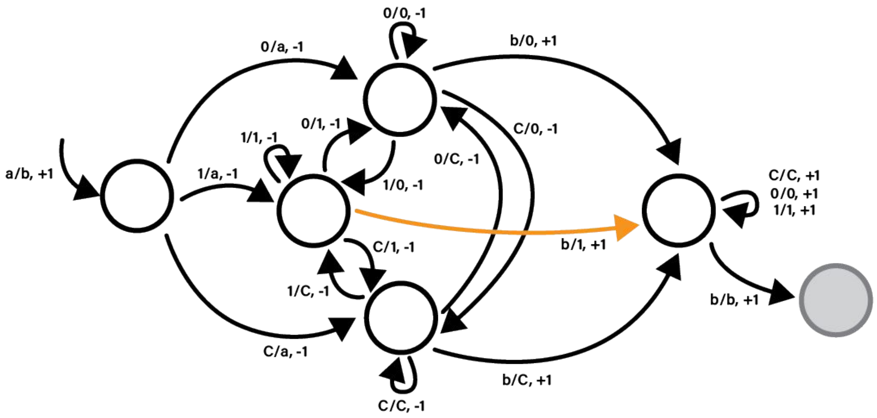
Move Pointer A to the Left

[Move a to position b] -> ba



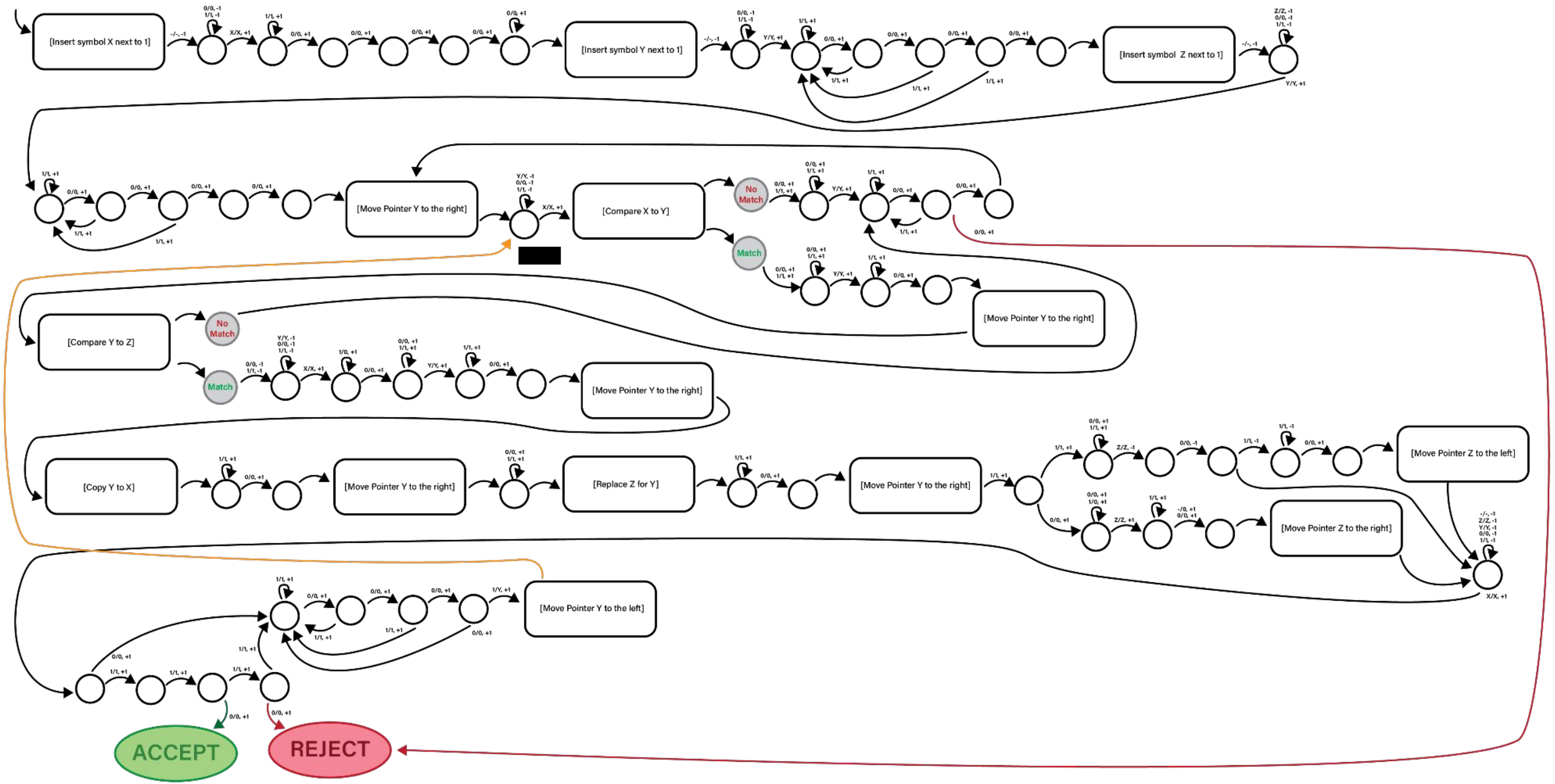
Move Pointer B to the Right

[Move b to position a] -> ba

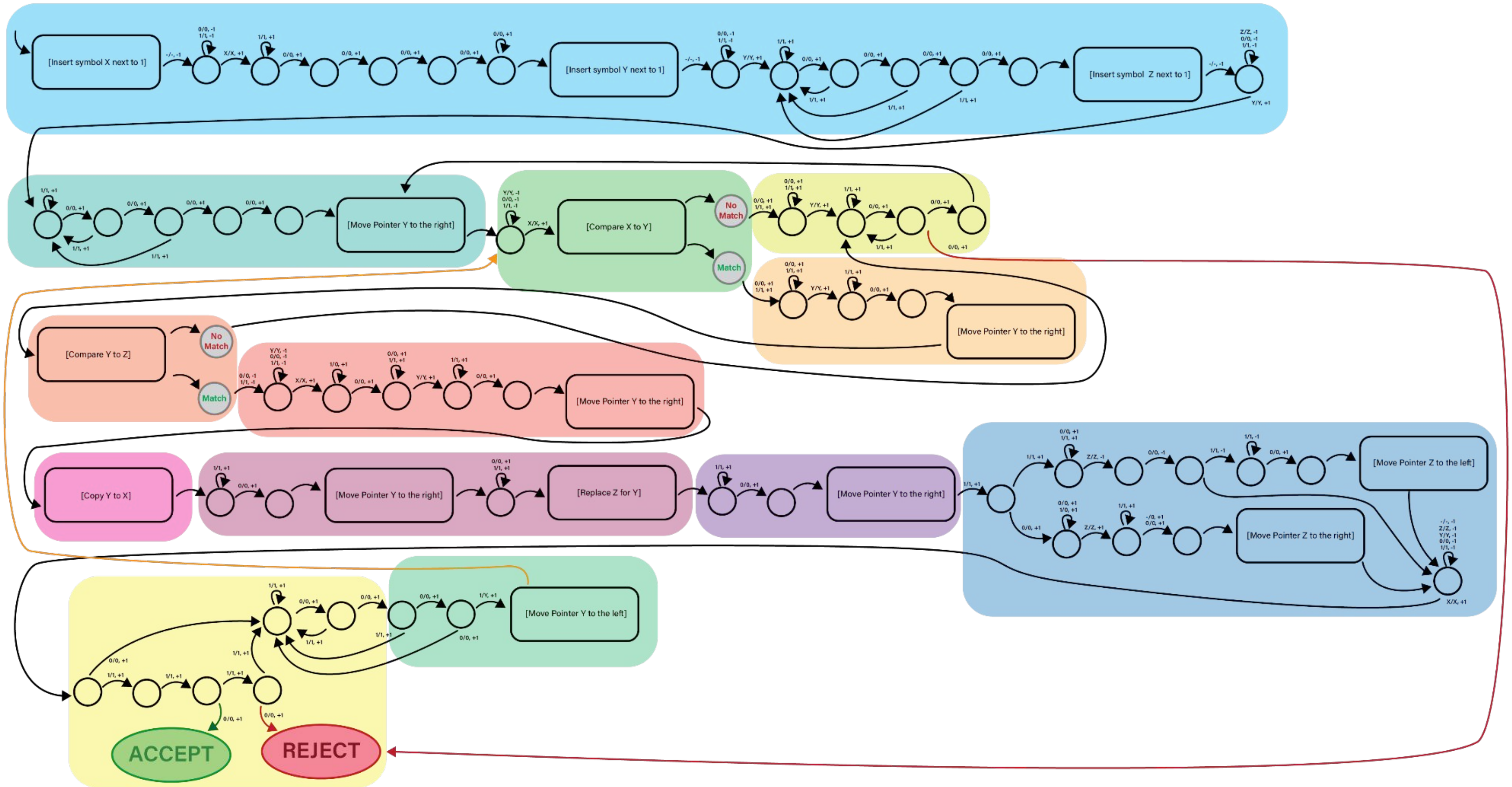


1-Tape UTM Implementation

Single Tape Universal Turing Machine

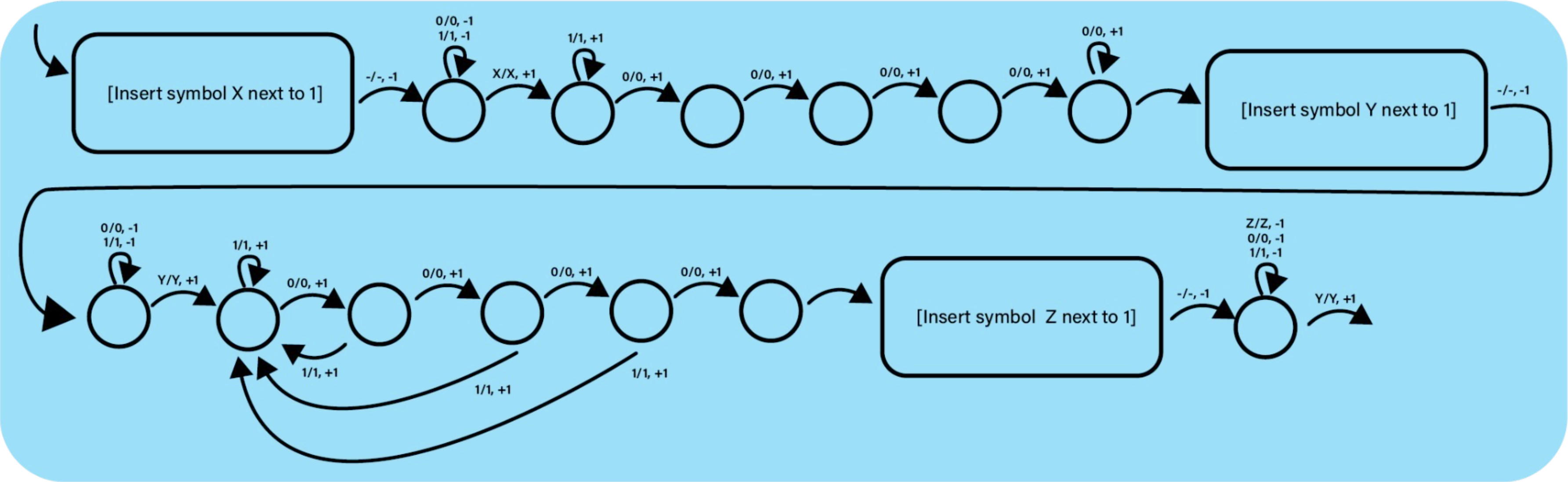


Single Tape Universal Turing Machine



UTM Algorithm

1. Add the pointers to the encoding of the Turing Machine M.



Example tape:

<buffer> 0000 <M> 0000 <w>

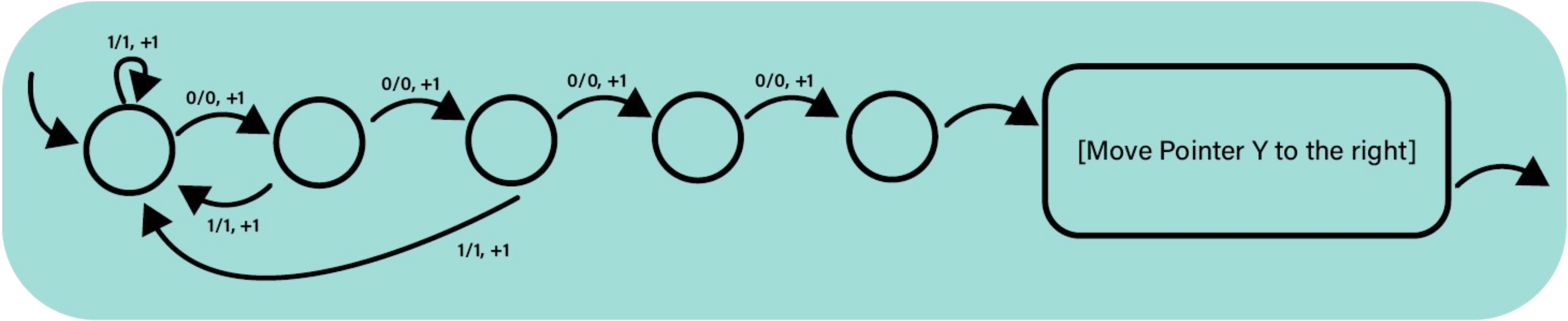
<u>X</u><buffer> 0000 <M> 0000 <w>

X<buffer> 0000 <u>Y</u><M> 0000 <w>

X<buffer> 0000 Y<M> 0000 <u>Z</u><w>

UTM Algorithm

2. Move the Y pointer to the first transition in the transition function "delta."



Example tape:

X<buffer> 0000 Y<Q> 00 <Σ> 00 <Γ> 000 **<δ>** 000 <qs> 00 <qacc> 00 <qrej> 0000 Z<w>

X<buffer> 0000 Y<Q> <Σ> <Γ> 000 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

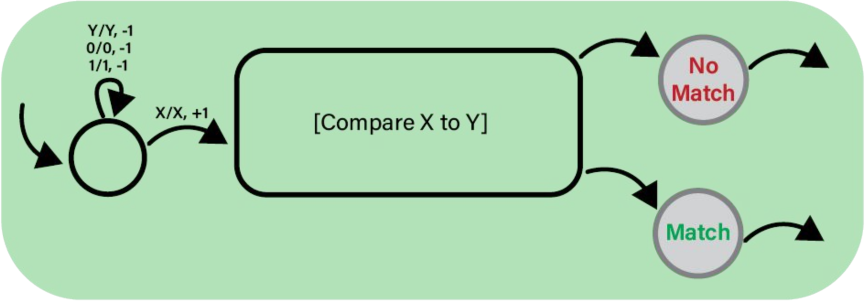
X<buffer> 0000 <<Q> <Σ> <Γ> 000 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

UTM Algorithm

3. Return to X and compare X and Y. If they are the same go to step 5 If they are not, go to step 4.

Compare the current state of M with the state p in the first transition of "delta."



Example tape:

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

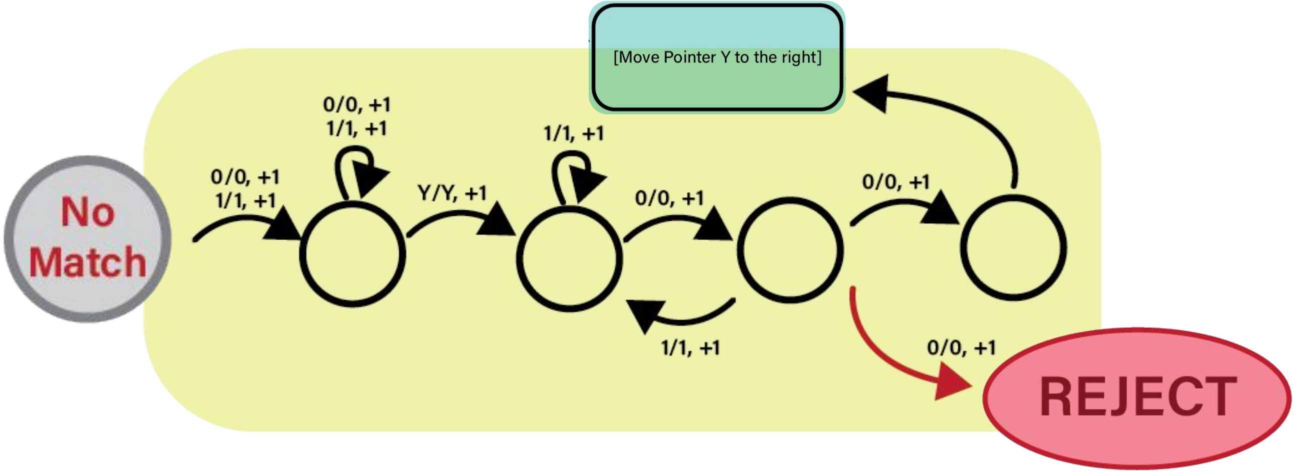
X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

UTM Algorithm

4. Move the Y pointer to the next transition of delta and return to step 3. If there are no transitions left, reject the string.



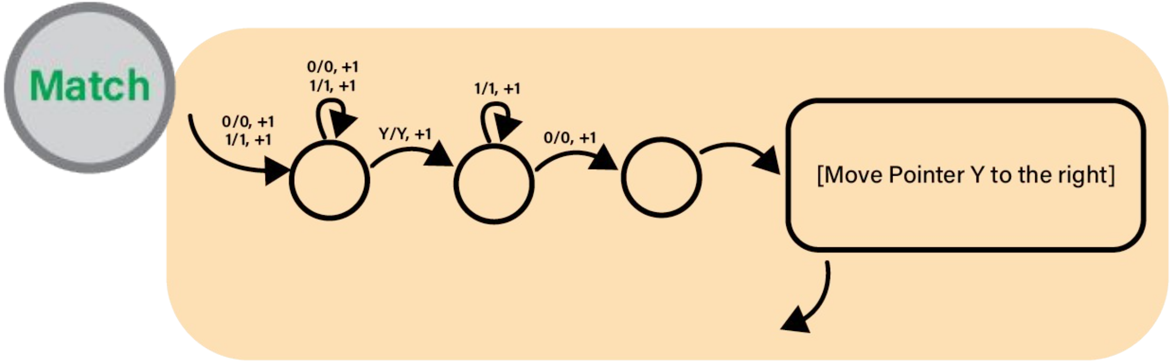
Example tape:

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

UTM Algorithm

5. Move the Y pointer to the next item in the current transition (symbol a).



Example tape:

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p>*** Y***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

X<buffer> 0000 <Q> <Σ> <Γ> 000 Y***<p*** *0* Y***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

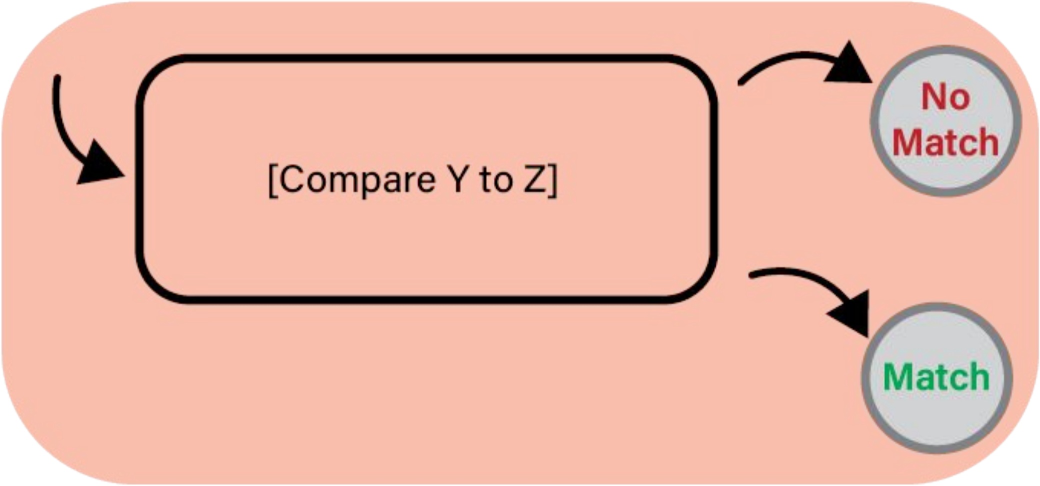
...

X<buffer> 0000 <Q> <Σ> <Γ> 000 ***<p>*** *0* Y***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** 00 ***<p>*** *0* ***<a>*** *0* ***<q>*** *0* ***<b>*** *0* ***<±1>*** ...

UTM Algorithm

6. Compare Y and Z. If they are the same, go to step 7, if they are not, return to step 4.

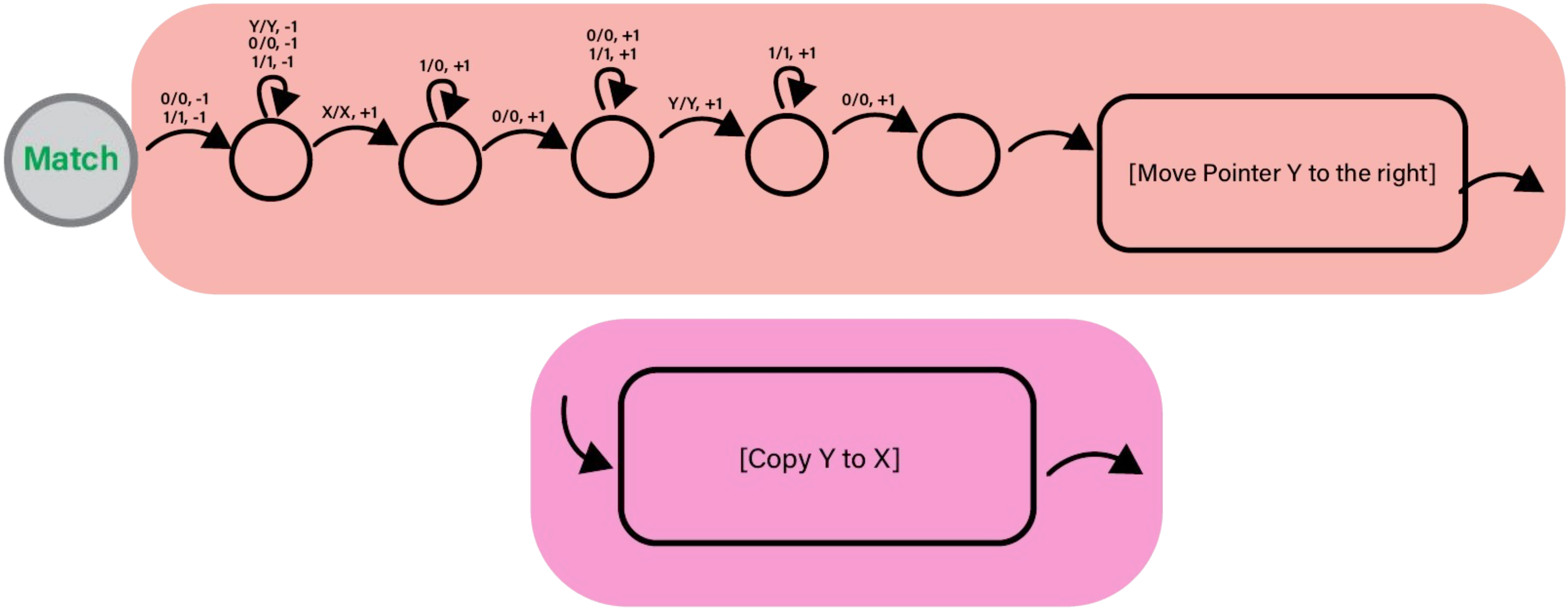
Compare the current symbol in the tape of M with the symbol in the transition



UTM Algorithm

7. Empty the buffer in X (turn the current state to a string of 0's) and move the Y pointer to the next item in the current transition (state q). Then, copy Y to X

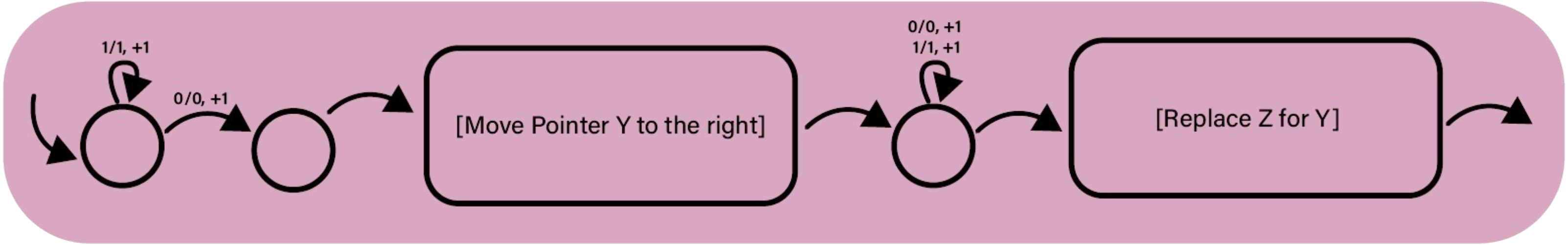
Copy the state q to the buffer to set it as the next state.

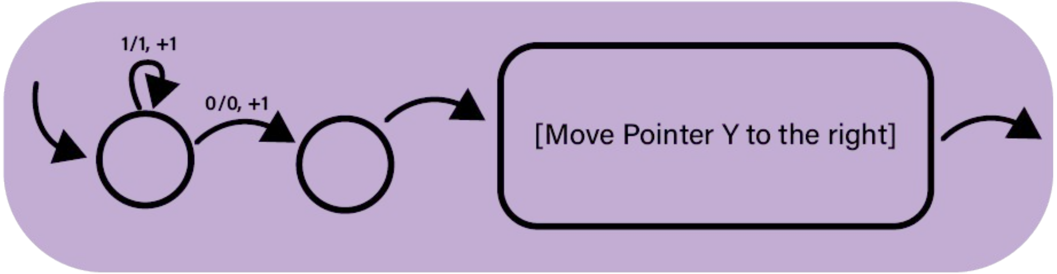


UTM Algorithm

8. Move the Y pointer to the next item in the current transition (symbol b) and replace Z with Y. Then move the Y pointer to the next item of the current transition (Movement of the tape head in M).

Replace the current symbol **a** in the tape for M with the new symbol **b** from the transition

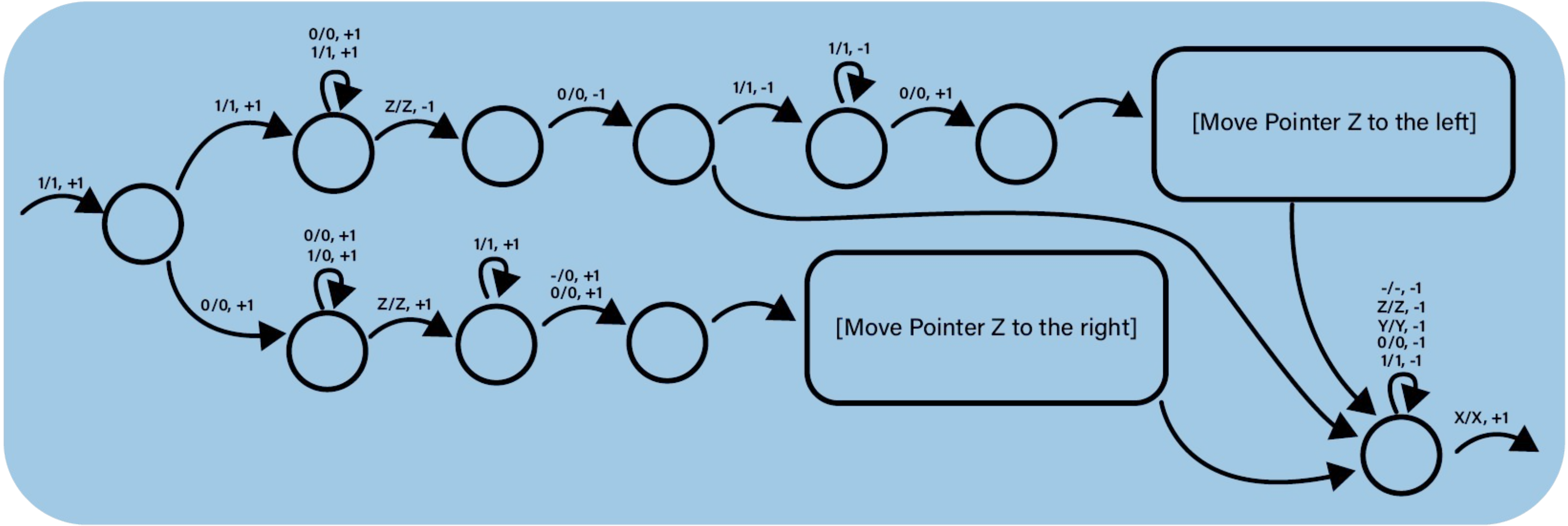




UTM Algorithm

9. In the next item of the current transition (Movement of the tape head in M).

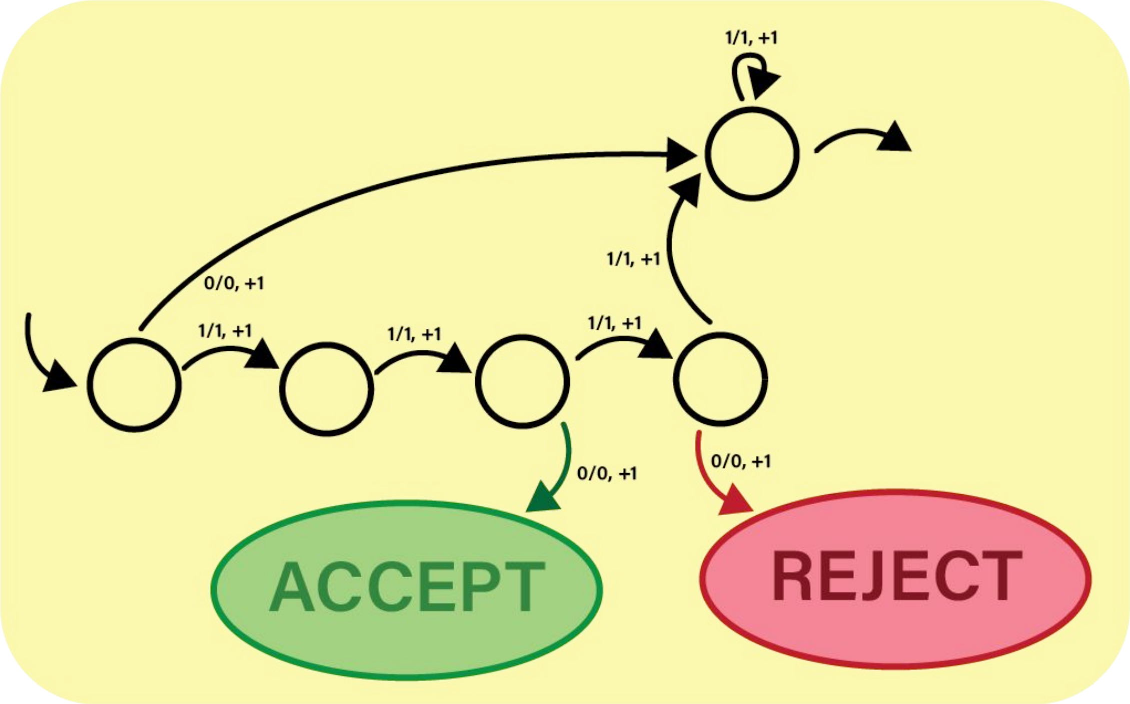
- If the transition indicates that the tape head has to move to the right, move the Z pointer to the right.
- If the transition indicates a movement to the left, move the Z pointer to the left.
- If the tape head is at the beginning of the tape, do not move it.



UTM Algorithm

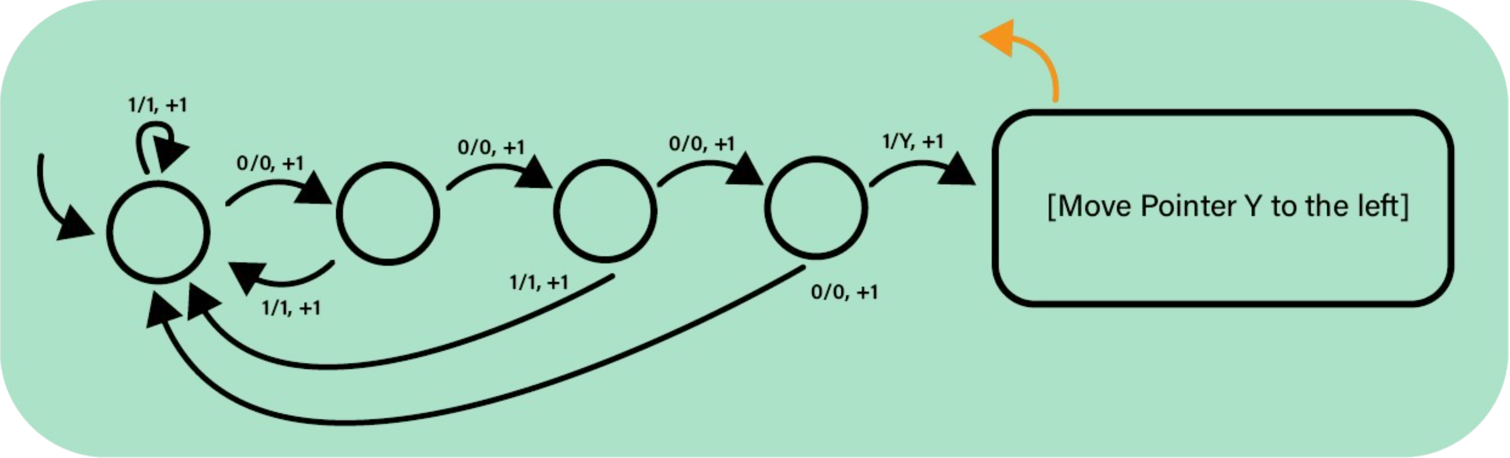
10. Return to X and read the current state of M.

- If the state is the accepting state, accept the string.
- If the state is in the rejecting state, reject the string.
- If the state is any other state, go to step 11.

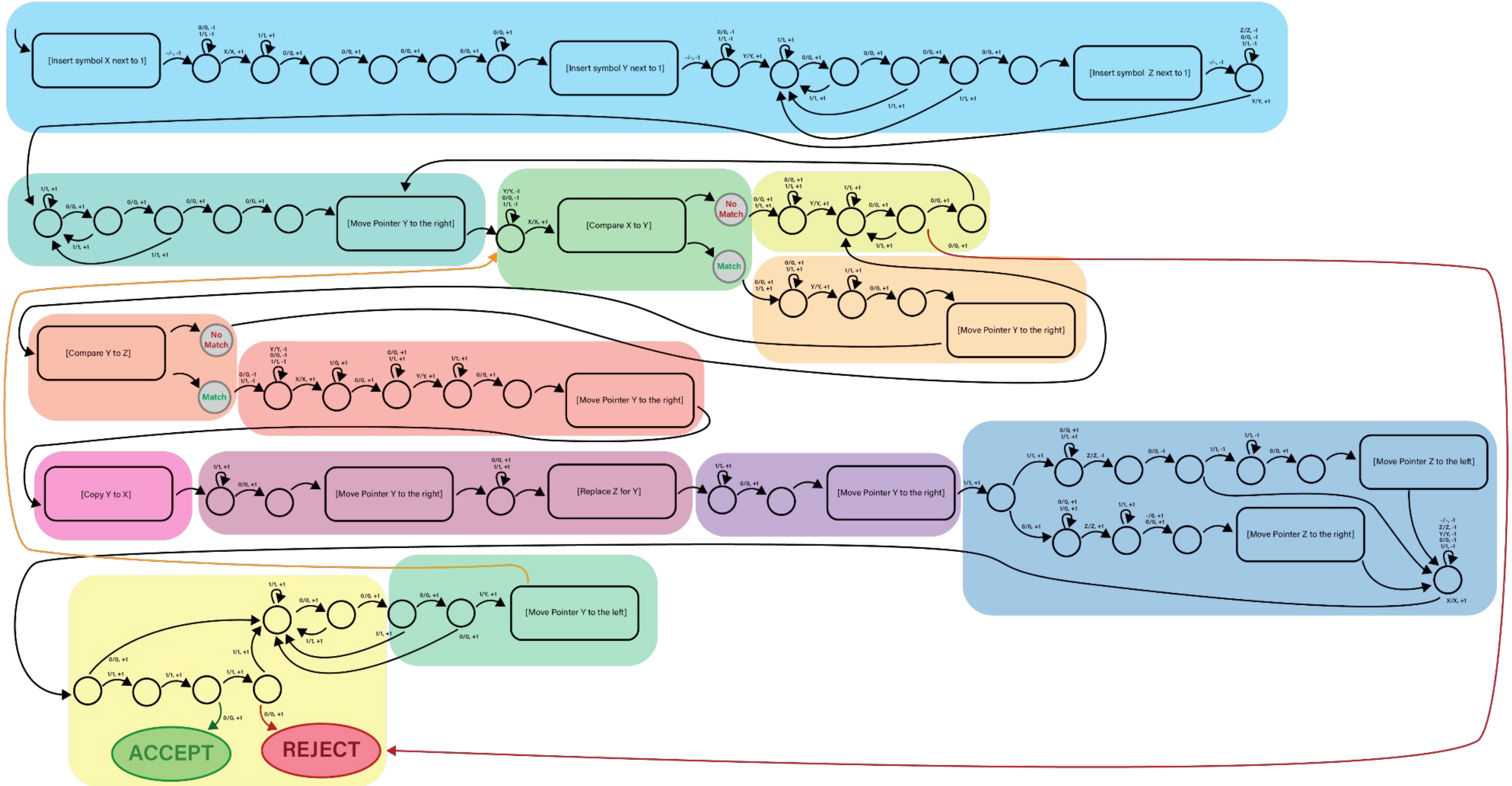


UTM Algorithm

11. Move the Y pointer to the first term of the transition function "delta" and loop back to step 2 to continue simulating the machine M.



Single Tape Universal Turing Machine



Thank you!