Automata Theory (2C)

- Turing Machine

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Turing Machine

Turing Machine

The Turing machine mathematically models a machine that mechanically operates on a tape.

On this tape are symbols, which the machine can read and write, one at a time, using a tape head.

Operation is fully determined by a finite set of elementary instructions such as

> "in state 42, if the symbol seen is 0, write a 1;
> if the symbol seen is 1, change into state 17;
> in state 17, if the symbol seen is 0,
> write a 1 and change to state 6;" etc.

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – Tape

A **tape** divided into **cells**, one next to the other.
Each cell contains a **symbol** from some finite **alphabet**.
The alphabet contains a **special blank** symbol
(here written as '0') and one or more other symbols.

The tape is assumed to be arbitrarily extendable
to the left and to the right, i.e.,
the Turing machine is always supplied with
as much tape as it needs for its computation.

Cells that have not been written before are assumed
to be filled with the **blank symbol**.

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – Head, State Register

A **head** that can read and write symbols on the tape and move the tape left and right one (and only one) cell at a time.
In some models the head moves and the tape is stationary.

A **state register** that stores the state of the Turing machine, one of finitely many.
Among these is the special **start state** with which the state register is initialized.
These states, writes Turing, replace the "state of mind" a person performing computations would ordinarily be in.

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – Table of Instruction

A **finite table** of **instructions** that,
given the **state**(qi) the machine is currently in
and the **symbol**(aj) it is reading on the tape
(symbol currently under the head),
tells the machine to do the following in sequence
(for the 5-tuple models):

1. Either erase or write a symbol (replacing aj with aj1).

2. Move the head (which is described by dk and can have values: 'L' for one step left or 'R' for one step right or 'N' for staying in the same place).

3. Assume the same or a new state as prescribed (go to state qi1).

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – unlimited amount

Note that every part of the machine
(i.e. its state, symbol-collections,
and used tape at any given time) and
its actions (such as printing, erasing and tape motion) is
finite, discrete and distinguishable;

it is the <u>unlimited</u> amount of tape and runtime
that gives it an <u>unbounded</u> amount of storage space.

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – 4 tuple models

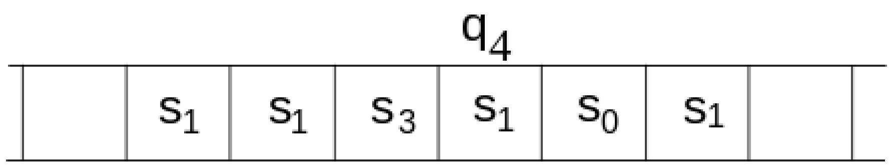
In the 4-tuple models,
erasing or writing a symbol (aj1) and
moving the head left or right (dk) are
specified as separate instructions.

Specifically, the table tells the machine to (ia) erase or write a symbol or (ib) move the head left or right, and then (ii) assume the same or a new state as prescribed, but not both actions (ia) and (ib) in the same instruction. In some models, if there is no entry in the table for the current combination of symbol and state then the machine will halt; other models require all entries to be filled.

Note that every part of the machine (i.e. its state, symbol-collections, and used tape at any given time) and its actions (such as printing, erasing and tape motion) is finite, discrete and distinguishable; it is the unlimited amount of tape and runtime that gives it an unbounded amount of storage space.

https://en.wikipedia.org/wiki/Turing_machine

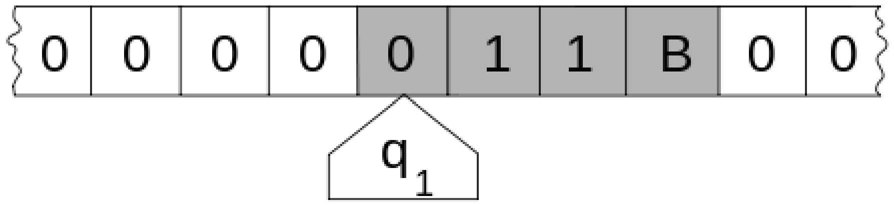
Turing Machine – head, instruction



The **head** is always over a particular square of the tape;
only a finite stretch of squares is shown.
The **instruction** to be performed (q4) is shown
over the scanned square.

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine – internal state, blank



Here, the **internal state** (q1) is shown inside the head,
and the illustration describes the **tape** as being infinite and **pre-filled** with "**0**", the symbol serving as **blank**.

The system's full state (its complete configuration) consists of the **internal state**, any **non-blank symbols** on the tape (in this illustration "11B"), and the **position** of the head relative to those symbols including blanks, i.e. "011B".

https://en.wikipedia.org/wiki/Turing_machine

Turing Machine

Turing machine as a 7-tuple $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ where

$\setminus$ set minus

Q is a finite, non-empty set of **states**;

Γ is a finite, non-empty set of tape **alphabet symbols**;

$b \in \Gamma$ is the **blank symbol** (the only symbol allowed to occur on the tape infinitely often at any step during the computation);

$\Sigma \subseteq \Gamma \setminus \{ b \}$ is the set of **input symbols**, that is, the set of symbols allowed to appear in the initial tape contents;

$q_0 \in Q$ is the **initial state**;

$F \subseteq Q$ is the set of **final states** or **accepting states**. The initial tape contents is said to be accepted by M if it eventually halts in a state from F .

$\delta : (Q \setminus F) \times \Gamma \rightarrow Q \times \Gamma \times \{ L , R \}$ is a partial function called the **transition function**, where **L** is **left shift**, **R** is **right shift**. (A relatively uncommon variant allows "**no shift**", say **N**, as a third element of the latter set.) If δ is not defined on the current state and the current tape symbol, then the machine halts;

https://en.wikipedia.org/wiki/Turing_machine

3-State Busy Beaver

The 7-tuple for the 3-state busy beaver looks like this (see more about this busy beaver at Turing machine examples):

$Q = \{ A , B , C , HALT \}$	(states);
$\Gamma = \{ 0 , 1 \}$	(tape alphabet symbols);
$b = 0$	(blank symbol);
$\Sigma = \{ 1 \}$	(input symbols);
$q\ 0 = A$	(initial state);
$F = \{ HALT \}$	(final states);
$\delta =$ see state-table below	(transition function).

Initially all tape cells are marked with 0

https://en.wikipedia.org/wiki/Turing_machine

3-State Busy Beaver

The 7-tuple for the 3-state busy beaver looks like this (see more about this busy beaver at Turing machine examples):

$Q = \{ A , B , C , HALT \}$	(states);
$\Gamma = \{ 0 , 1 \}$	(tape alphabet symbols);
$b = 0$	(blank symbol);
$\Sigma = \{ 1 \}$	(input symbols);
$q\ 0 = A$	(initial state);
$F = \{ HALT \}$	(final states);
$\delta =$ see state-table below	(transition function).

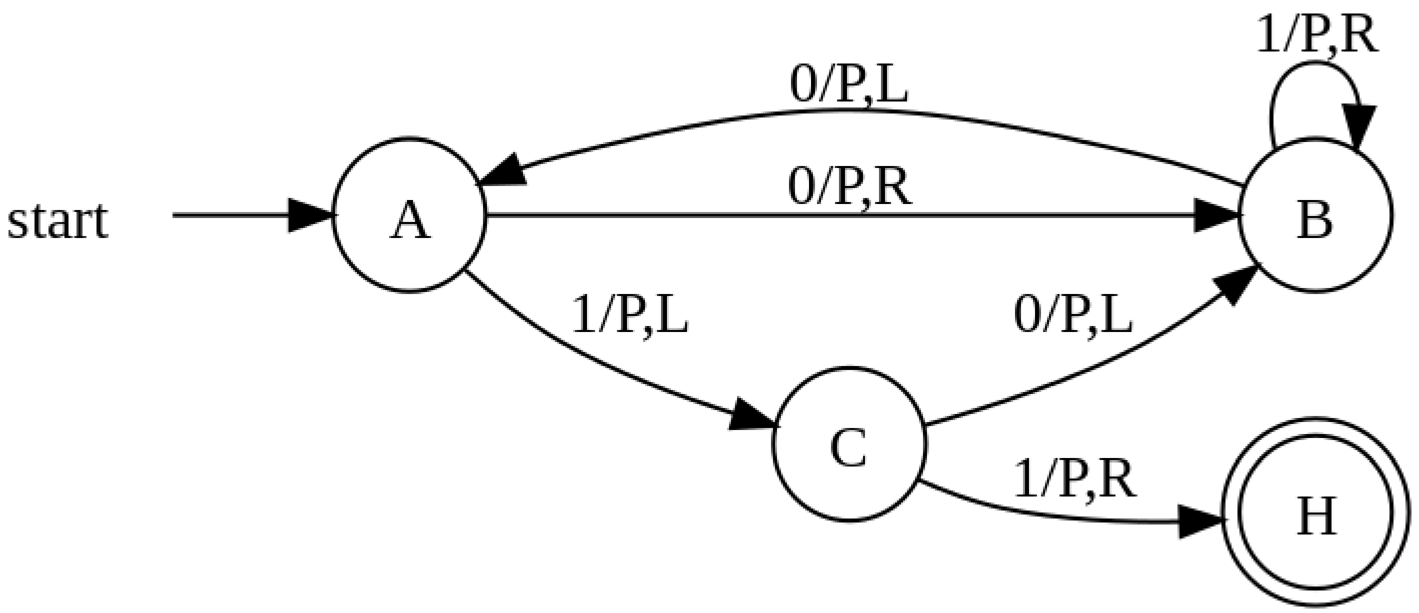
Initially all tape cells are marked with 0

https://en.wikipedia.org/wiki/Turing_machine

3-State Busy Beaver

State table for 3 state, 2 symbol busy beaver

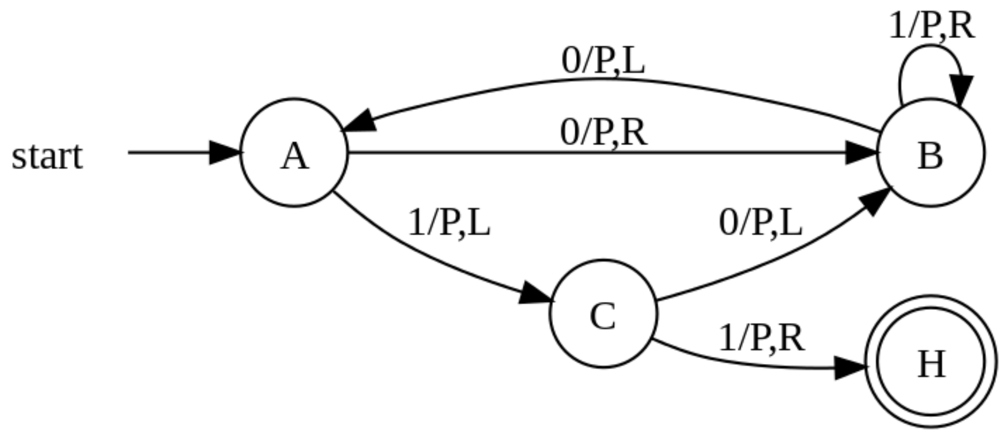
Tape symbol	**Current state A**			**Current state B**			**Current state C**		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state
0	1	R	**B**	1	L	**A**	1	L	**B**
1	1	L	**C**	1	R	**B**	1	R	**HALT**



https://en.wikipedia.org/wiki/Turing_machine

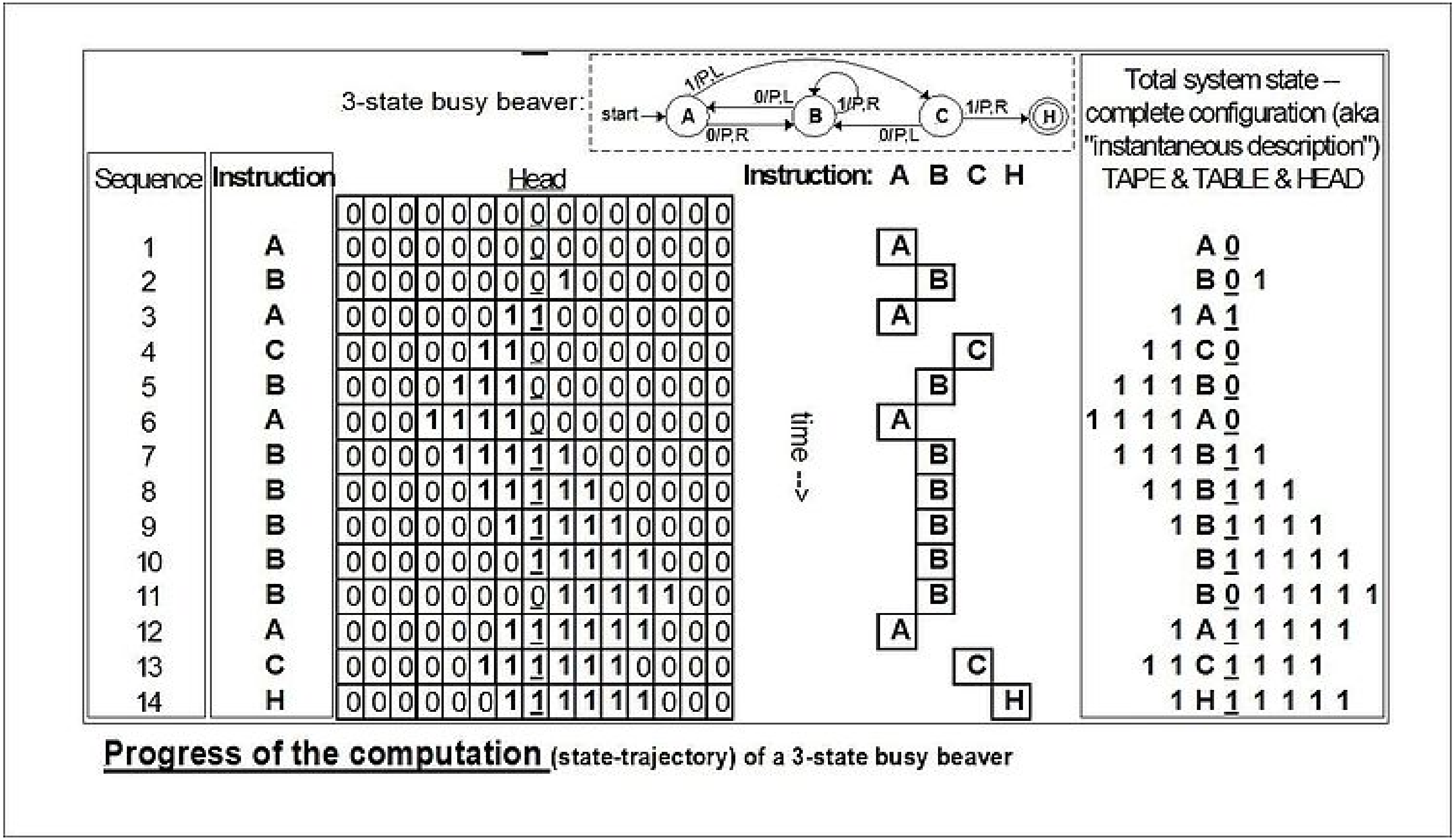
3-State Busy Beaver

Each circle represents a "**state**" of the table
—an "**m-configuration**" or "**instruction**".
"**Direction**" of a state transition is shown by an **arrow**.
The **label** (e.g. 0/P,R) near the outgoing state
(at the "tail" of the arrow) specifies the **scanned symbol**
that causes a particular transition (e.g. 0) followed by a slash /,
followed by the subsequent "**behaviors**" of the machine,
e.g. "P Print" then move tape "R Right".



https://en.wikipedia.org/wiki/Turing_machine

3-State Busy Beaver



3-state busy beaver:

$$\begin{tikzcd}
\text{start} \arrow[r] & A \arrow[r, bend right, "0/P,R"'] \arrow[rr, bend left, "1/P,L"] & B \arrow[l, "0/P,L"'] \arrow[r, "1/P,R"] \arrow[loop above, "1/P,R"] & C \arrow[l, "0/P,L"] \arrow[r, "1/P,R"] & H
\end{tikzcd}$$

Sequence	Instruction								Head								Instruction: A	B	C	H	Total system state – complete configuration (aka "instantaneous description") TAPE & TABLE & HEAD
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
1	A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A				A 0
2	B	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0		B			B 0 1
3	A	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	A				1 A 1
4	C	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0			C		1 1 C 0
5	B	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0		B			1 1 1 B 0
6	A	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	A				1 1 1 1 A 0
7	B	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0		B			1 1 1 B 1 1
8	B	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0		B			1 1 B 1 1 1
9	B	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0		B			1 B 1 1 1 1
10	B	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0		B			B 1 1 1 1 1
11	B	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0		B			B 0 1 1 1 1 1
12	A	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	A				1 A 1 1 1 1 1
13	C	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0			C		1 1 C 1 1 1 1
14	H	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0				H	1 H 1 1 1 1 1

Progress of the computation (state-trajectory) of a 3-state busy beaver

https://en.wikipedia.org/wiki/Turing_machine

n-State Busy Beaver

the design specifications:

1. The machine has **n** "**operational**" **states** plus a **Halt state**, where **n** is a positive integer, and one of the **n** states is distinguished as the **starting state**.
2. The machine uses a single two-way infinite (or unbounded) **tape**.
3. The **tape alphabet** is {0, 1}, with **0** serving as the **blank symbol**.
4. The machine's **transition function** takes two inputs:
    - the **current** non-Halt state,
    - the **symbol** in the current tape cell,

   and produces three outputs:
    - a **symbol** to write over the symbol in the current tape cell
      (it may be the same symbol as the symbol overwritten),
    - a **direction** to move (**left** or **right)**
    - a **state** to **transition** into (which may be the Halt state).

https://en.wikipedia.org/wiki/Turing_machine

n-State Busy Beaver

"**Running**" the machine consists of starting in the **starting state**, with the current tape cell being any cell of a **blank** (all-0) tape, and then iterating the **transition function** until the **Halt** state is entered (if ever).

If, and only if, the machine eventually halts, then the number of 1s finally remaining on the tape is called the machine's score.

The n-state busy beaver (BB-n) game is a contest to find such an n-state Turing machine having the largest possible score
— the largest number of 1s on its tape after halting.
A machine that attains the largest possible score among all n-state Turing machines is called an n-state busy beaver, and a machine whose score is merely the highest so far attained (perhaps not the largest possible) is called a champion n-state machine.

https://en.wikipedia.org/wiki/Turing_machine

References

[1] http://en.wikipedia.org/
[2]