CHAITANYA
WOMEN'S COLLEGE
DEGREE & PG
AFFILIATED TO ANDHRA UNIVERSITY

TURING MACHINES

TMs model the computing capability of a general purpose computer, which informally can be described as:

- Effective procedure
  - Finitely describable
  - Well defined, discrete, "mechanical" steps
  - Always terminates
- Computable function
  - A function computable by an effective procedure

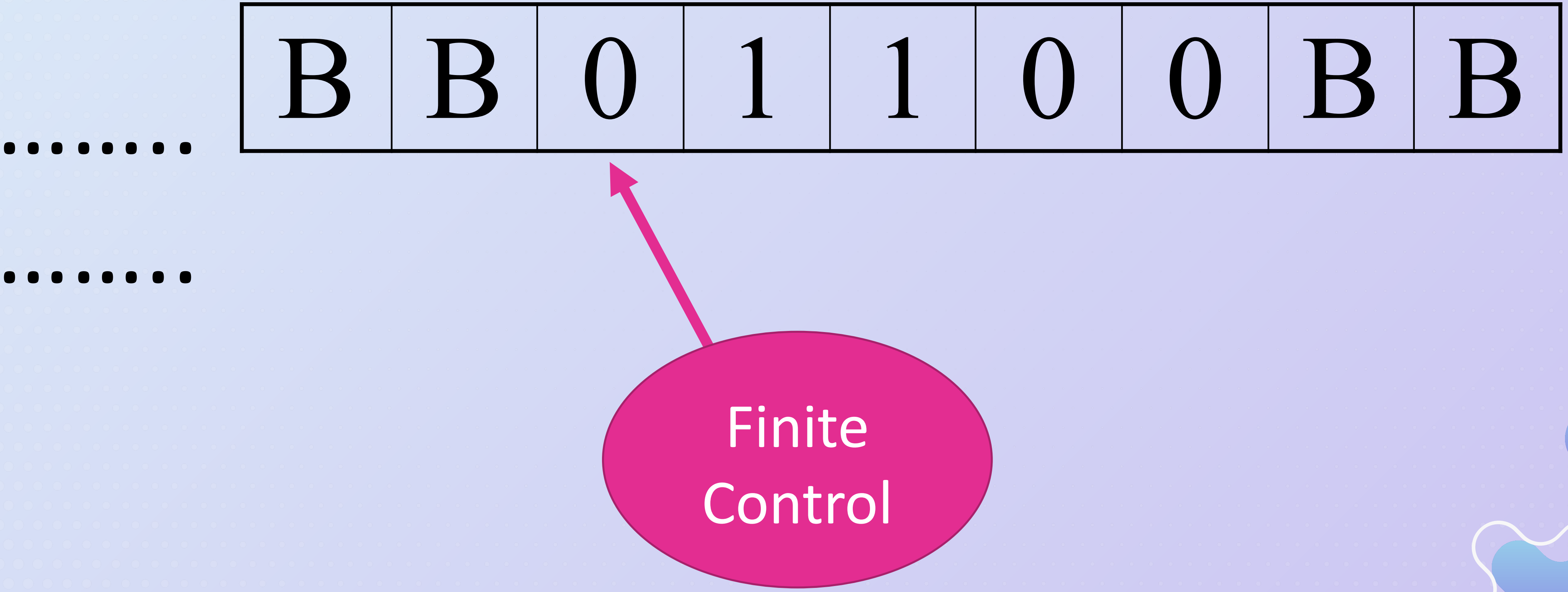
TMs formalize the above notion.

Church-Turing Thesis: There is an effective procedure for solving a problem if and only if there is a TM that halts for all inputs and solves the problem.

- There are many other computing models, but all are equivalent to or subsumed by TMs. *There is no more powerful machine* (Technically cannot be proved).

DFAs and PDAs do not model all effective procedures or computable functions, but only a subset.

Deterministic Turing Machine (DTM)



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

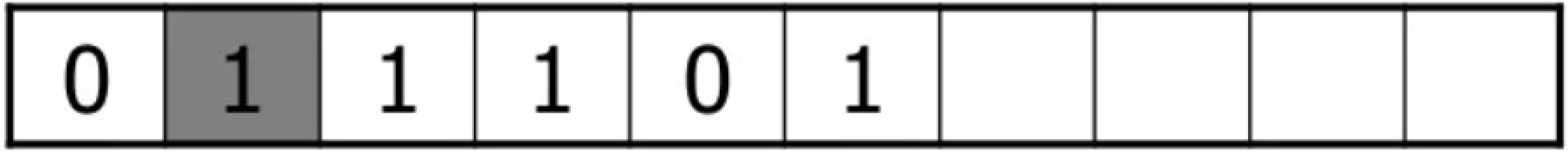
If read •, write 1, HALT!

1	1	1	1	0	1				

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!

0	0	1	1	0	1				

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!

0	0	0	1	0	1				

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!

0	0	0	0	0	1				

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!

0	0	0	0	1	1				

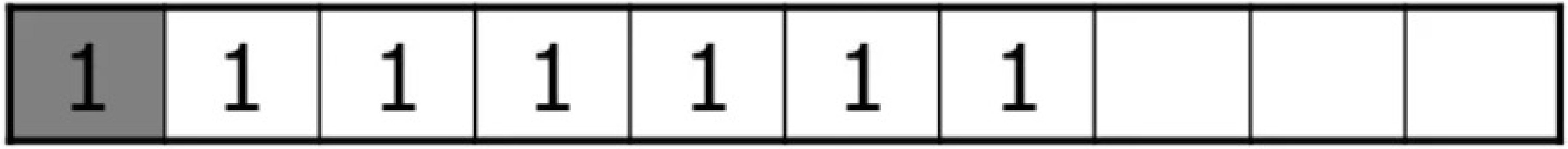
So the successor's output on 111101 was 000011 which is the reverse binary representation of 48.

Similarly, the successor of 127 should be 128:

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

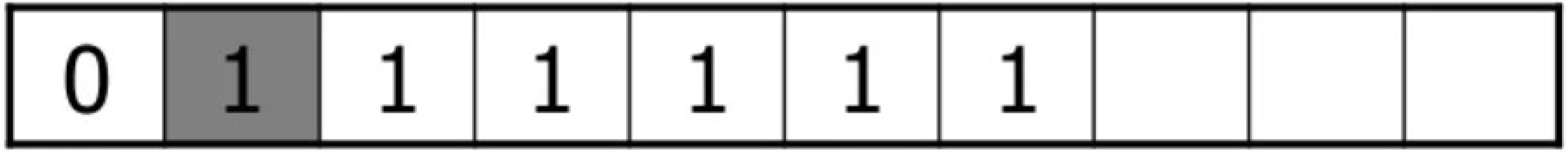
If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

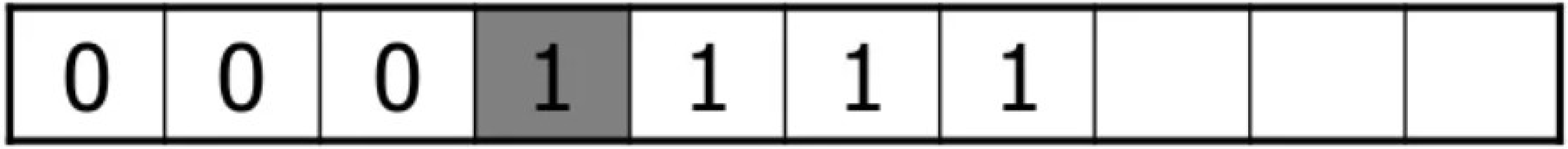
If read •, write 1, HALT!

0	0	1	1	1	1	1			

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

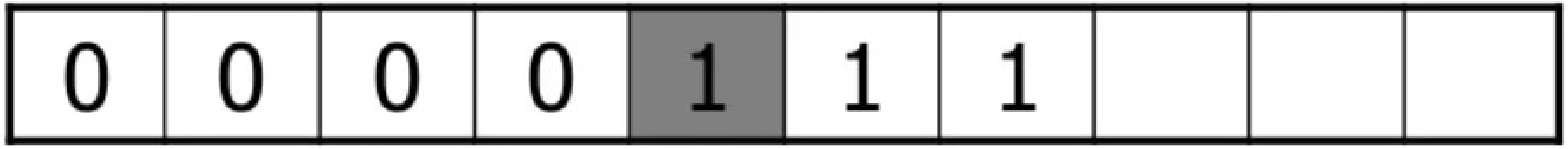
If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

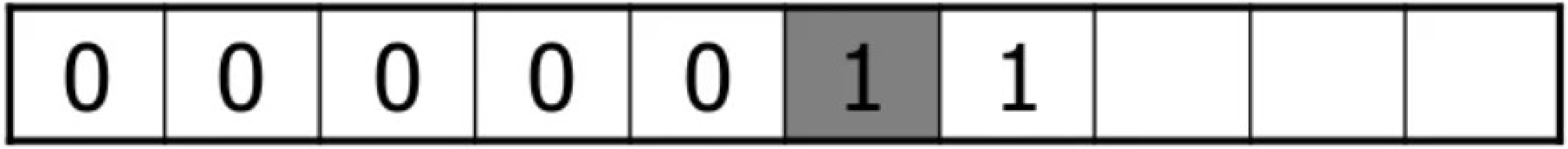
If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

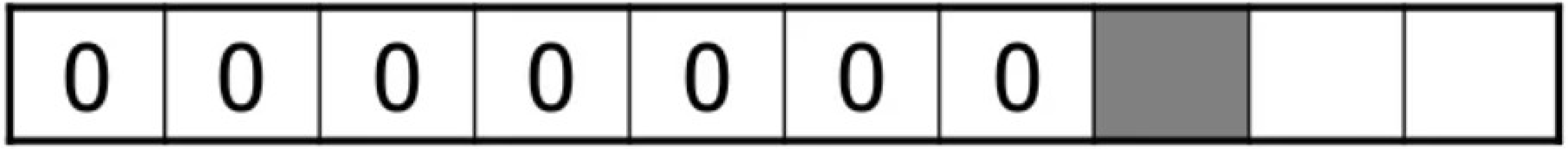
If read •, write 1, HALT!

0	0	0	0	0	0	1			

If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!



If read 1, write 0, go right, repeat.

If read 0, write 1, HALT!

If read •, write 1, HALT!

0	0	0	0	0	0	0	1		

- Two-way, infinite tape, broken into cells, each containing one symbol.
- Two-way, read/write tape head.
- An input string is placed on the tape, padded to the left and right infinitely with blanks, read/write head is positioned at the left end of input string.
- Finite control, i.e., a program, containing the position of the read head, current symbol being scanned, and the current state.
- In one move, depending on the current state and the current symbol being scanned, the TM

  1) changes state,

  2) prints a symbol over the cell being scanned, and

  3) moves its’ tape head one cell left or right.
- Many modifications possible, but Church-Turing declares equivalence of all.

Formal Definition of a DTM

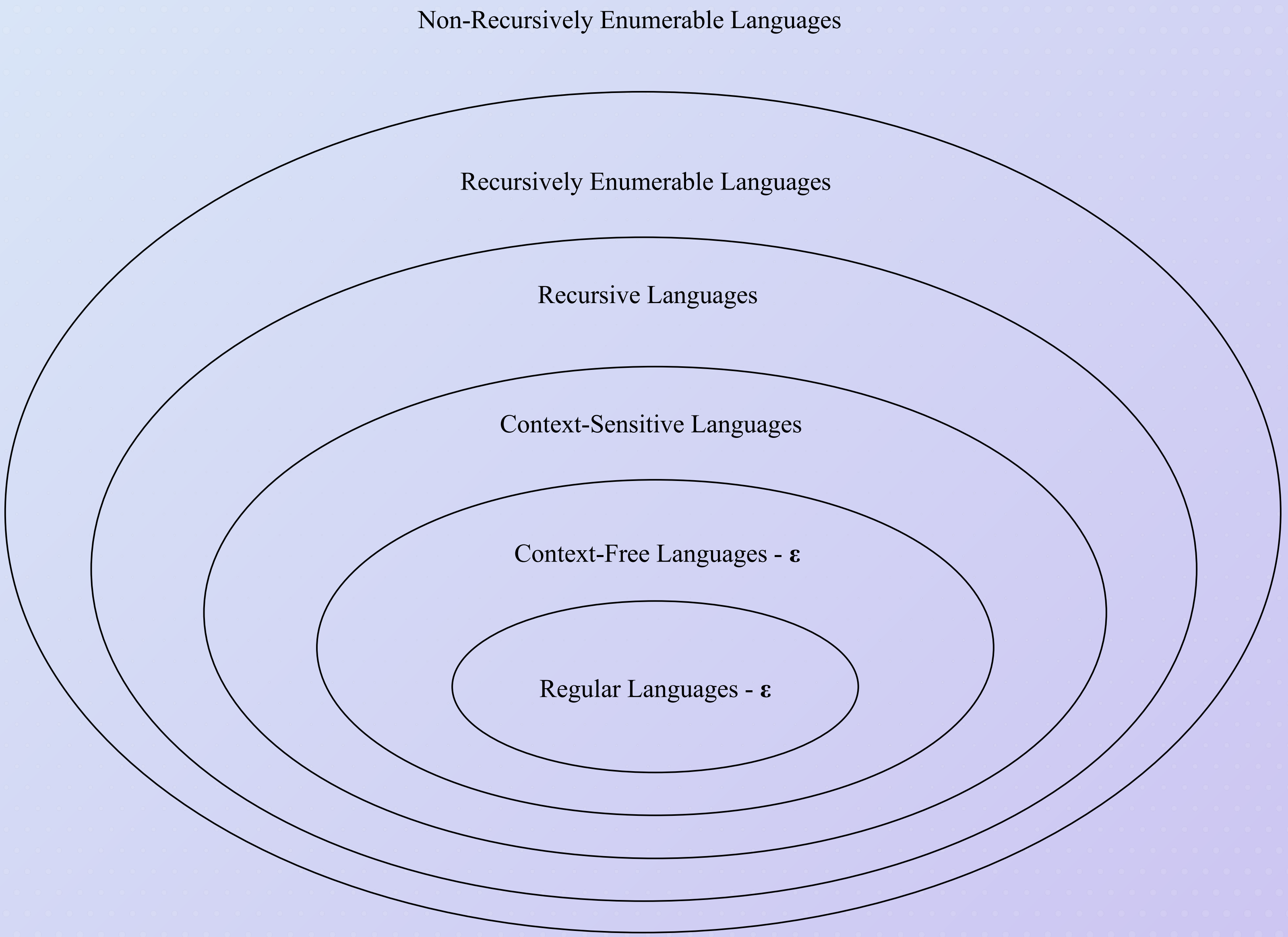
- A DTM is a seven-tuple:

  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Q	A finite set of states
Σ	A finite input alphabet, which is a subset of $\Gamma - \{B\}$
Γ	A finite tape alphabet, which is a strict superset of Σ
B	A distinguished blank symbol, which is in Γ
q_0	The initial/starting state, q_0 is in Q
F	A set of final/accepting states, which is a subset of Q
δ	A next-move function, which is a *mapping* (i.e., may be undefined) from

  $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$

  Intuitively, $\delta(q,s)$ specifies the next state, symbol to be written, and the direction of tape head movement by M after reading symbol s while in state q.


Non-Recursively Enumerable Languages
Recursively Enumerable Languages
Recursive Languages
Context-Sensitive Languages
Context-Free Languages - ε
Regular Languages - ε

The Halting Problem

- **Definition:** A decision problem is a problem having a yes/no answer (that one presumably wants to solve with a computer). Typically, there is a list of parameters on which the problem is based.
  - Given a list of numbers, is that list sorted?
  - Given a number x, is x even?
  - Given a C program, does that C program contain any syntax errors?
  - Given a TM (or C program), does that TM contain an infinite loop?

From a practical perspective, many decision problems do not seem all that interesting. However, from a theoretical perspective they are for the following two reasons:

- Decision problems are more convenient/easier to work with when proving complexity results.
- Non-decision *counter-parts* can always be created & are typically at least as difficult to solve.