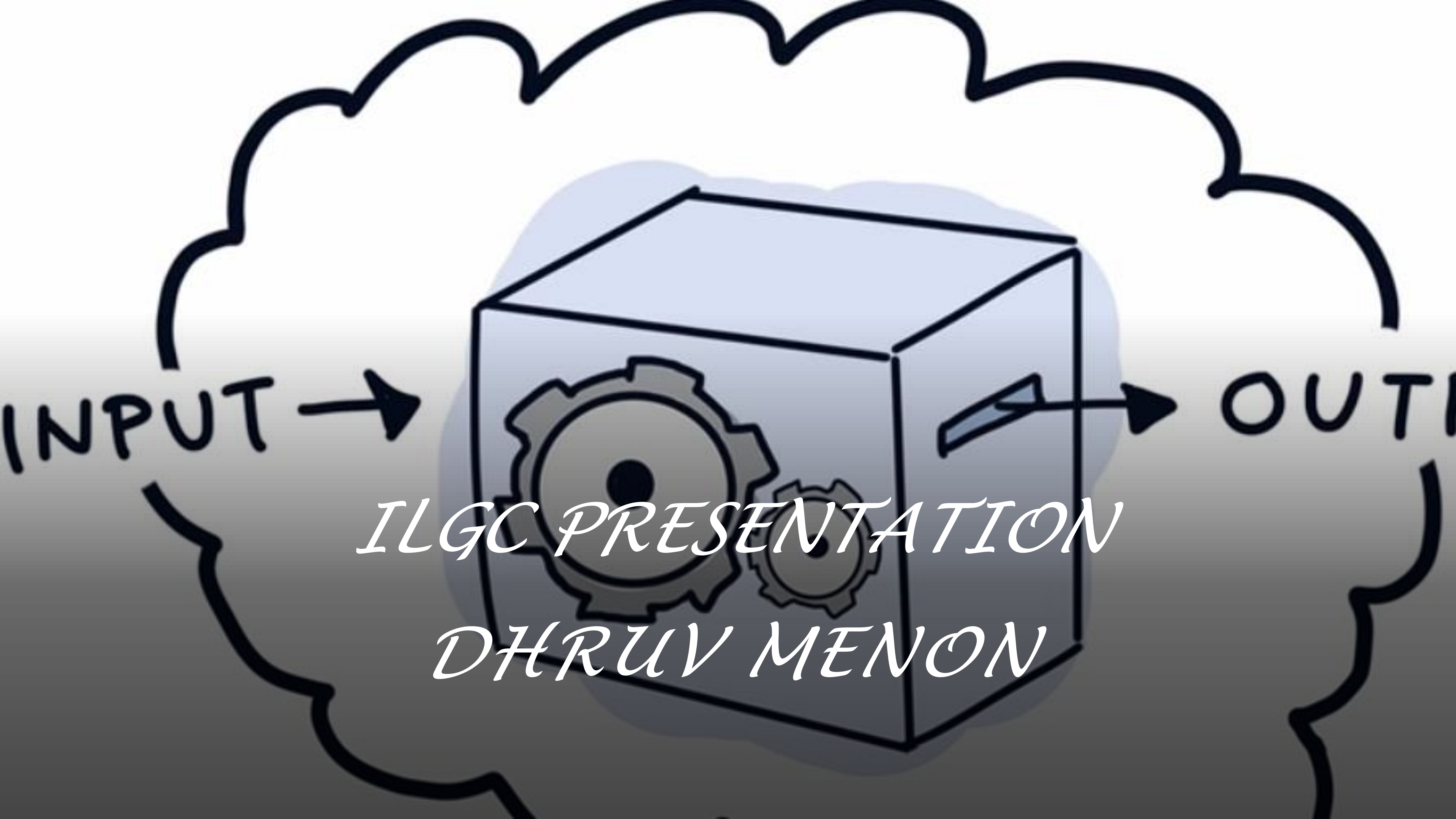


ILGC PRESENTATION

DHRUV MENON

Objectives

- Learn Finite Automata Theory
- Designing an algorithm to test finite automata

What are Languages?

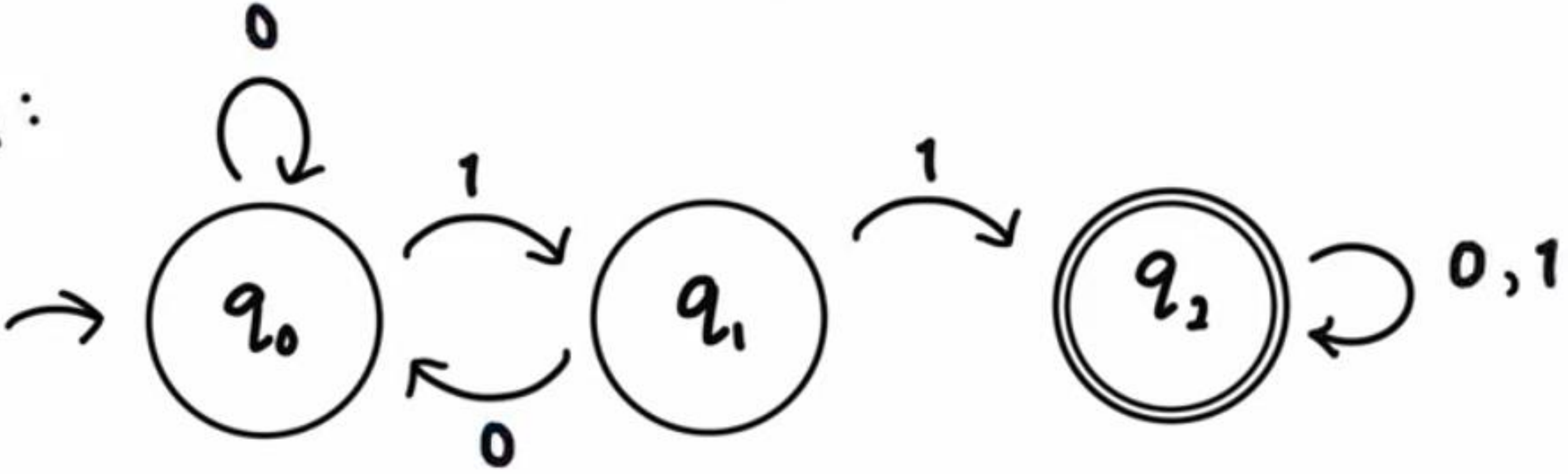
- Languages are strings of symbols that are inputs into the machine.
- Goal: Design a transition function so that the machine accepts only those languages that we want it to accept!

DID A HEATWAVE OCCUR?

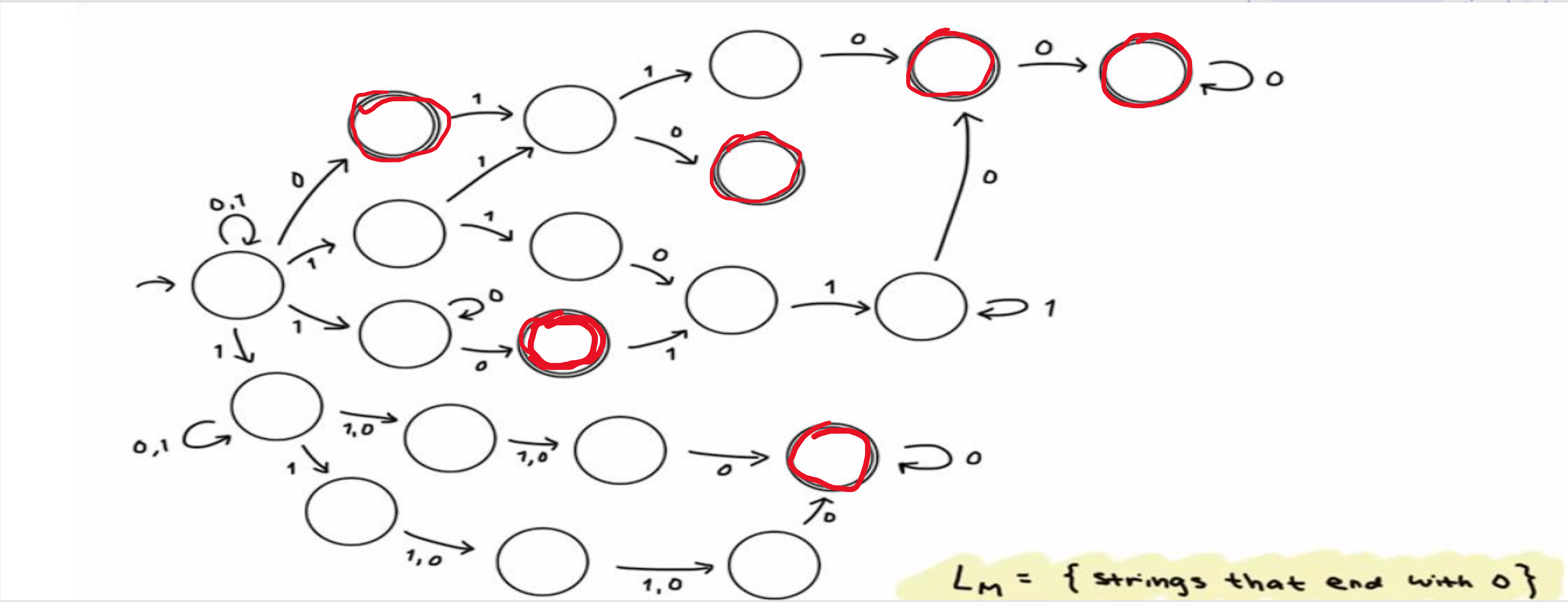
INPUT: STRING OF WEATHER DATA

* heatwave: temperature $\geq 45°C$ (113°F) for 2 consecutive days

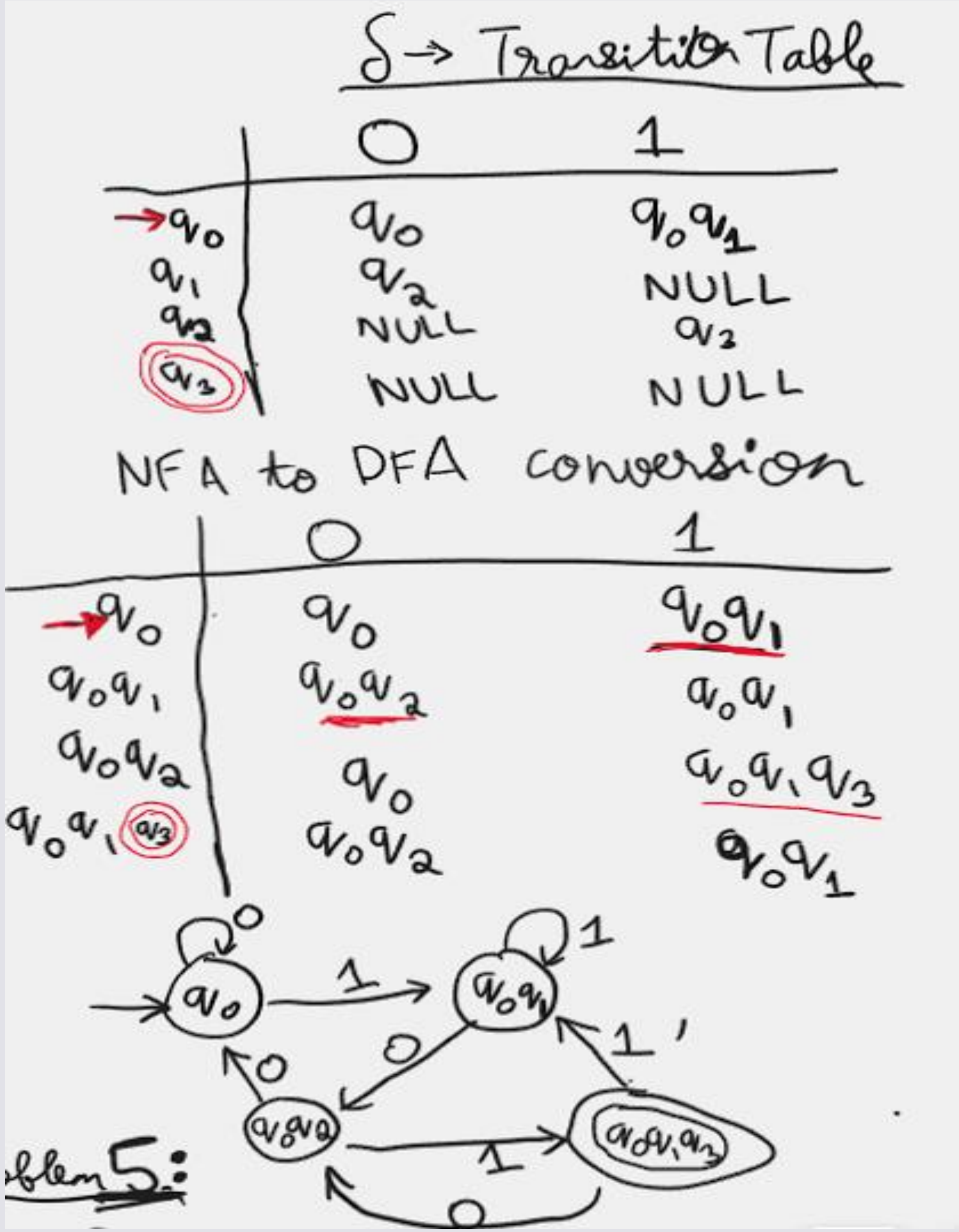
M:



Non-Deterministic Finite Automata



NFA=DFA(Hard to Believe?)



δ → Transition Table

	0	1
→q_0	q_0	q_0q_1
q_1	q_2	NULL
q_2	NULL	q_3
q_3	NULL	NULL

NFA to DFA conversion

	0	1
→q_0	q_0	q_0q_1
q_0q_1	q_0q_2	q_0q_1
q_0q_2	q_0	$q_0q_1q_3$
$q_0q_1q_3$	q_0q_2	q_0q_1

How do we automate the testing process?

- *There are infinitely many possible strings so its impossible manually! Answer:*
- Convert the regular expression pattern into a deterministic finite automaton (DFA).
- Construct a DFA from the user-defined transition table.
- Intersect the two DFAs to find any discrepancies.
- Check for language equivalence over strings of the specified length.

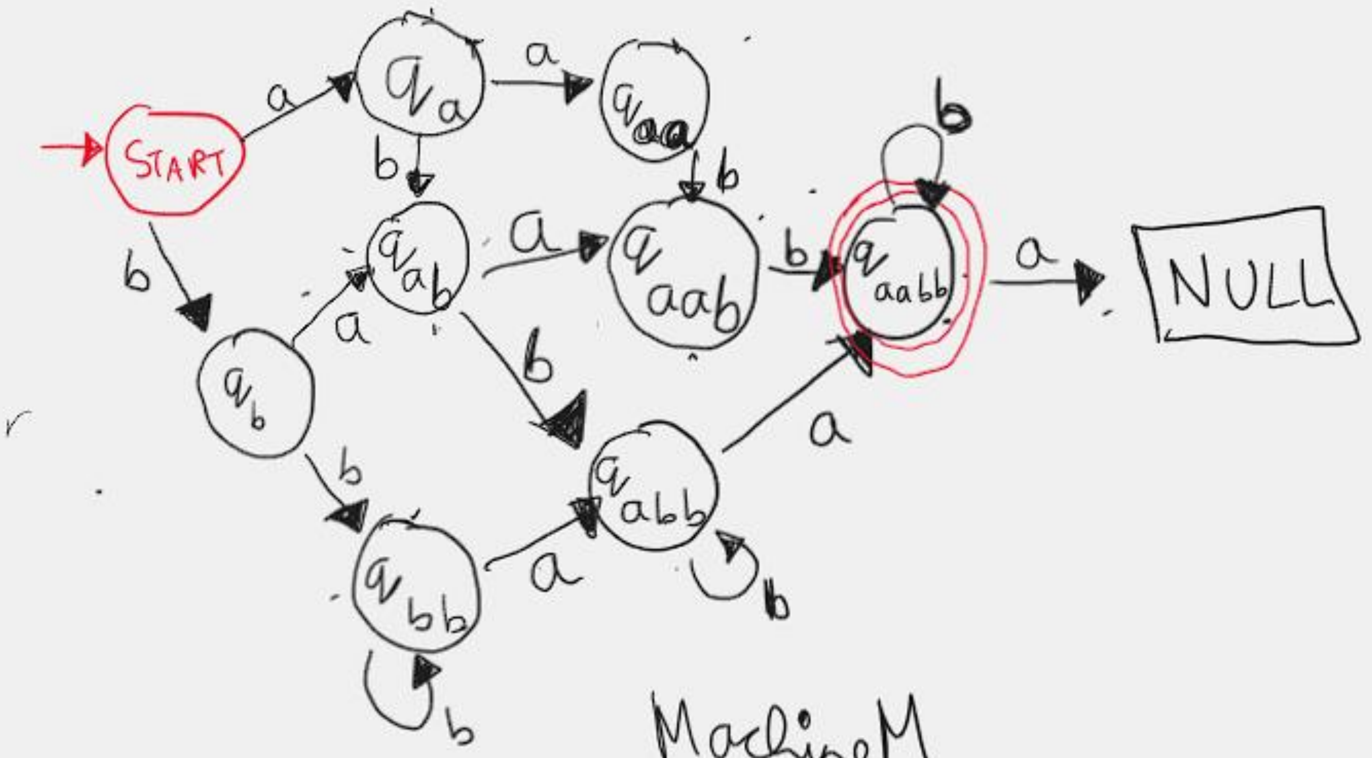
$\underline{M_1}$ $\underline{M_2}$

Goal: Combine these 2 Machines to create a new Machine M that can read both L_1 and L_2.

(i) $\{w \mid w$ has exactly 2 a's and atleast 2 b's $\}$

$q_a \rightarrow$ State that M has read 1 'a'.

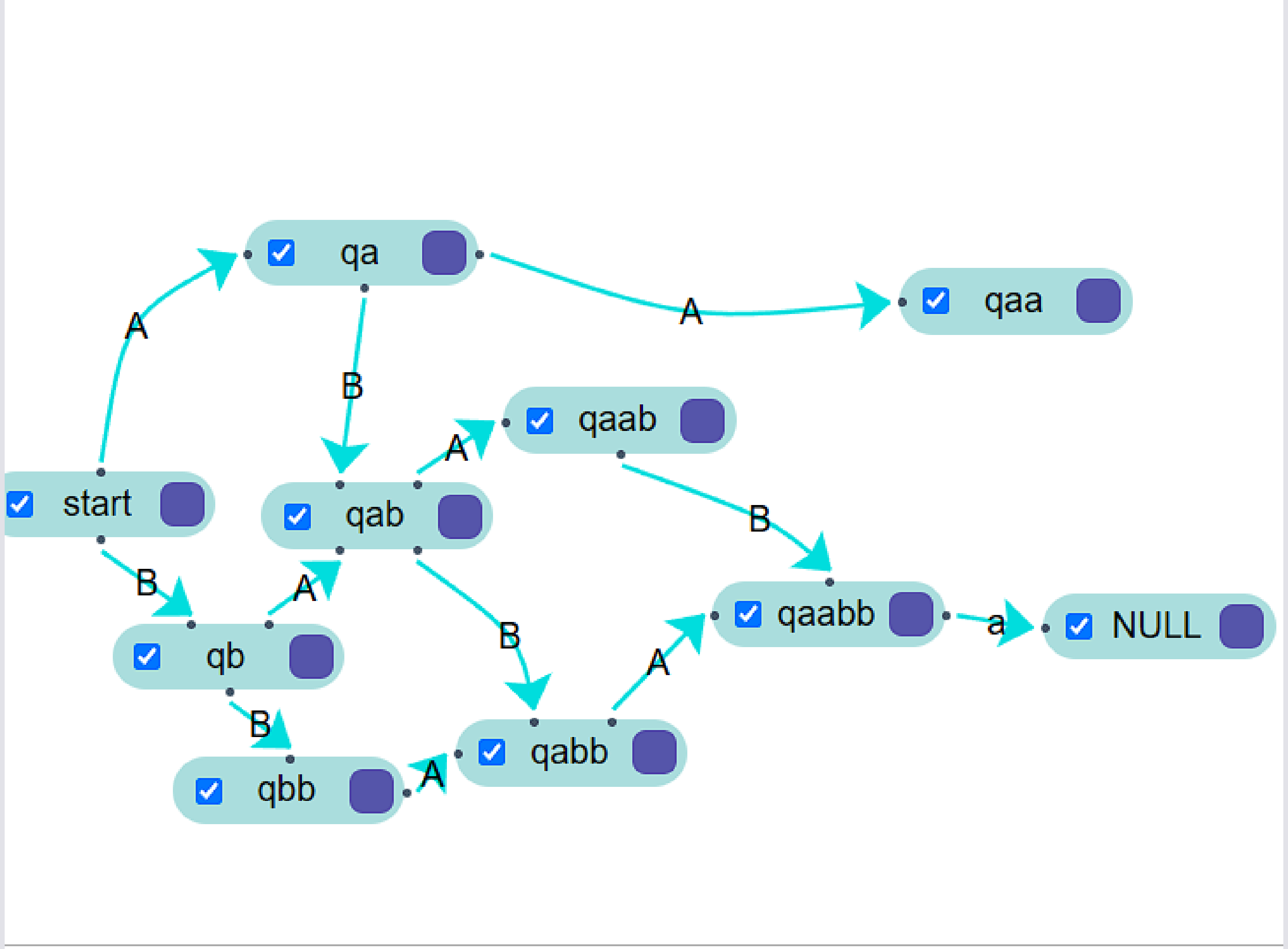
$q_b \rightarrow$ State that M has read 1 'b'.



Machine M

Results from Algorithm that I designed

Automaton Simulator:



est Results:

ccept: AA -- Pass
ccept: BB -- Pass
ccept: [Empty String] -- Pass
ccept: [Empty String] -- Pass
eject: [Empty String] -- Fail

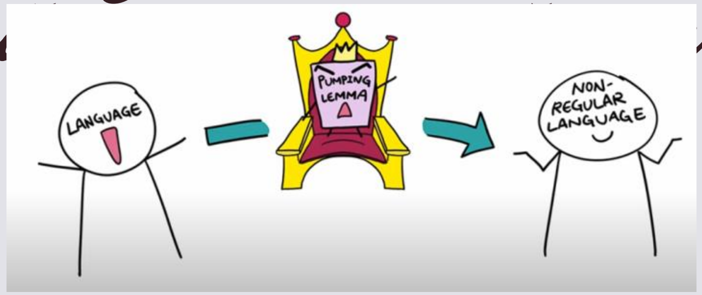
Limitations of Finite Automata

- Finite Automata do not have memory.

NFA to DFA time complexity(2^no. Of states)

- There are patterns of strings that Finite Automata can't recognize(predicting a heatwave might require counting an... ...umber of hot days from

past data)



Applications of Finite Automata

- Finite automata can be used to recognize patterns in the data in an efficient manner when there are billions of data points!
- Since Finite Automata are basically First Order Markov Chains that ignore what's on the stack includes all applications of first order markov chains
- Programming Language Design
- Computer Performance Evaluation
- Shannon Information Theory

References

- Michael Sipser. 2006. Introduction to the Theory of Computation (2nd. ed.). International Thomson Publishing
- Alon, N., Krivelevich, M., Newman, I. and Szegedy, M., 2001. Regular languages are testable with a constant number of queries. SIAM Journal on Computing, 30(6), pp.1842-1862.

Appendix

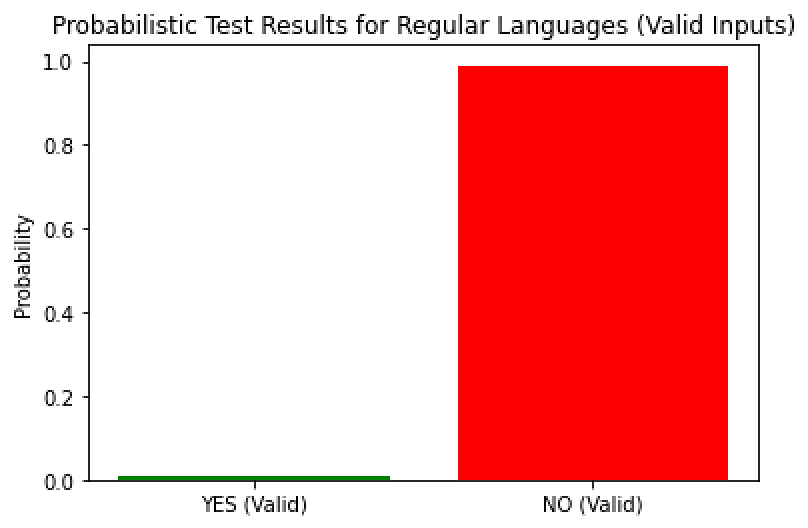
- Probabilistic Algorithm for testing if a Language is regular
- Designed an algorithm to concatenate strings from a regular language to form a 1st order Dycke Language proved and tested if its a context free language.

Probabilistic Algorithm for testing if a Language is Regular

Definition 2.2 *A language is regular iff there exists a finite automaton that accepts it.*

Question: Is there a single machine that can read the entire Language?
Test Input: Language L : strings with equal number of 1's followed by a 0 and 0's followed by a 1

Sample Test Output:



ALGORITHM

Input: a word w of length $|w| = n$;

1. For each $1 \leq i \leq \log(8km/\epsilon)$ choose r_i random runs in w of length 2^{i+1} each;
2. For each admissible triplet (A, P, Π) with $A = (C_{i_1}, \ldots, C_{i_t})$, $P = (p_j^1, p_j^2)_{j=1}^t$, $\Pi = (n_j)_{j=1}^{t+1}$ such that for all $2 \leq j \leq t$ one has $n_j \in T_s$ for some $1 \leq s \leq S$, do the following:
   - Form the automata M_j, $1 \leq j \leq t$, as described above.
   - Discard those chosen runs which end or begin at place p for which $|p - n_j| \leq \epsilon n/(128km\log(1/\epsilon))$. Namely, those runs which have one of their ends closer than $\epsilon n/(128km\log(1/\epsilon))$ from some $n_j \in \Pi$.
   - For each remaining run R, if R falls between n_j and n_{j+1}, check whether it is feasible for the automaton M_j starting at $b - n_j + 1$ where b is the first coordinate of R in w. Namely, $b - n_j + 1$ is the place where R starts relative to n_j, which is the the place w "enters" M_j.
3. If for some admissible triplet all checked runs turned out to be feasible, output "YES". Otherwise (i.e, in the case where for all admissible triplets at least one infeasible run has been found) output "NO".

Lemma 2.8 *If $dist(w, L) \geq \epsilon n$, then the above algorithm outputs "NO" with probability at least 3/4. If $w \in L$, then the algorithm always outputs "YES".*

Designed an algorithm to concatenate strings from a regular language to form a 1st order Dycke Language proved and tested probabilistically that its a context free language.

For an integer $n \geq 1$, the *Dyck language of order n*, denoted by D_n, is the language over the alphabet of $2n$ symbols $\{a_1, b_1, \ldots, a_n, b_n\}$, grouped into n ordered pairs $(a_1, b_1), \ldots, (a_n, b_n)$. The language D_n is defined by the following productions:

1. $S \longrightarrow a_i S b_i$ for $i = 1, \ldots, n$;
2. $S \longrightarrow SS$;
3. $S \longrightarrow \gamma$,

ALGORITHM

Input: a word w of length $|w| = n$;

1. Choose a sample S of bits in the following way: For each bit of w, independently and with probability $p = d/n$ choose it to be in S. Then, if S contains more then $d + \Delta/4$ bits, answer 'YES' without querying any bit. Else,
2. If $dist(S, D_1 \cap \{0,1\}^{d'}) < \Delta$, where $d' = |S|$, output "YES", otherwise output "NO".