VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE304L
Theory of Computation

M7L1 – Recursive and Recursive Enumerable Languages

Dr. P. Saravanan

Turing Machine

- 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

Church-Turing Thesis

- **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.

Recursively Enumerable

- A language is **recursively enumerable** if some Turing machine accepts it
- Let L be a recursively enumerable language and M be the Turing Machine that accepts it

For string w :

- If W Є L then M halts in a final state
- If W $\not\in$ L then M halts in a non-final states or loops forever

Recursive Language

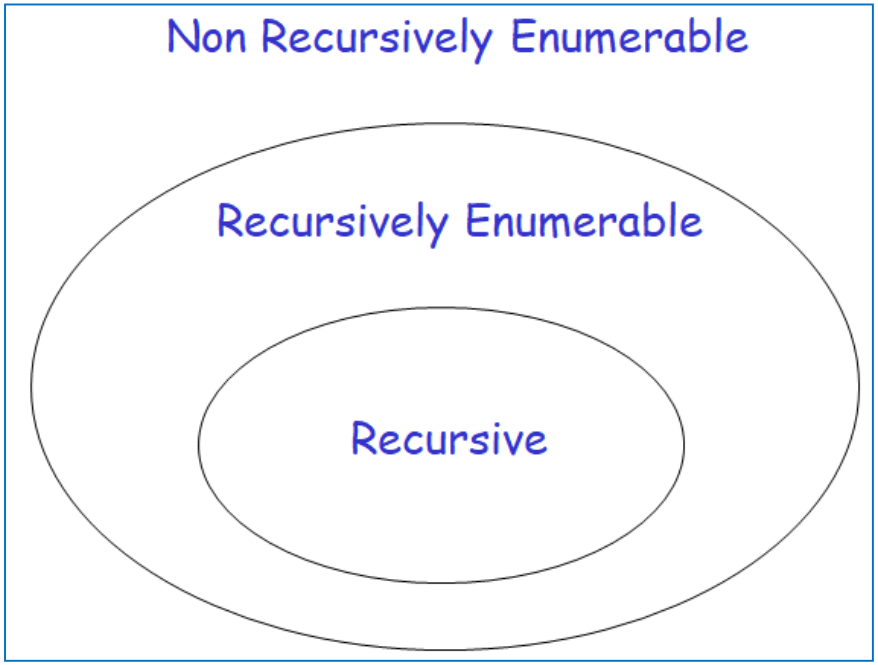
- A language is **recursive** if some Turing machine accepts it and halts on any input string
- Let L be a recursive language and M be the Turing Machine that accepts it

For string L :

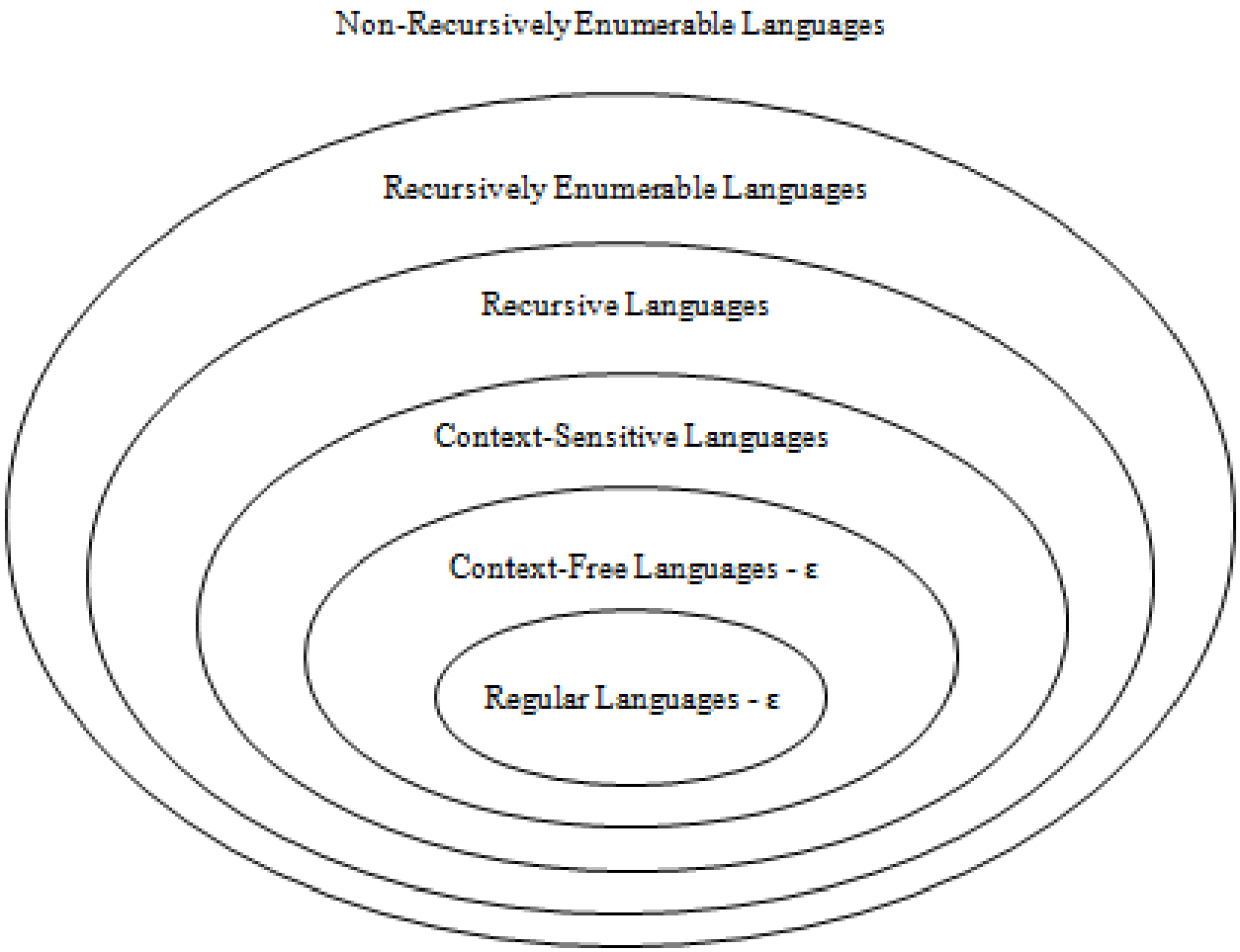
- If W $\in$ L then M halts in a final state
- If W $\notin$ L then M halts in a non-final state

RE and REC

- There is a specific language which is not recursively enumerable.
- There is a specific language which is recursively enumerable but not recursive



Chomsky hierarchy



Some questions..

- **Let *M* be a TM.**
  - Question: Is L(M) r.e.?
  - Answer: Yes! By definition it is!

  - Question: Is L(M) recursive?
  - Answer: Don't know, we don't have enough information.

Some questions..

- **Let *M* be a TM that halts on all inputs:**
  - Question: Is L(M) recursively enumerable?
  - Answer: Yes! By definition it is!

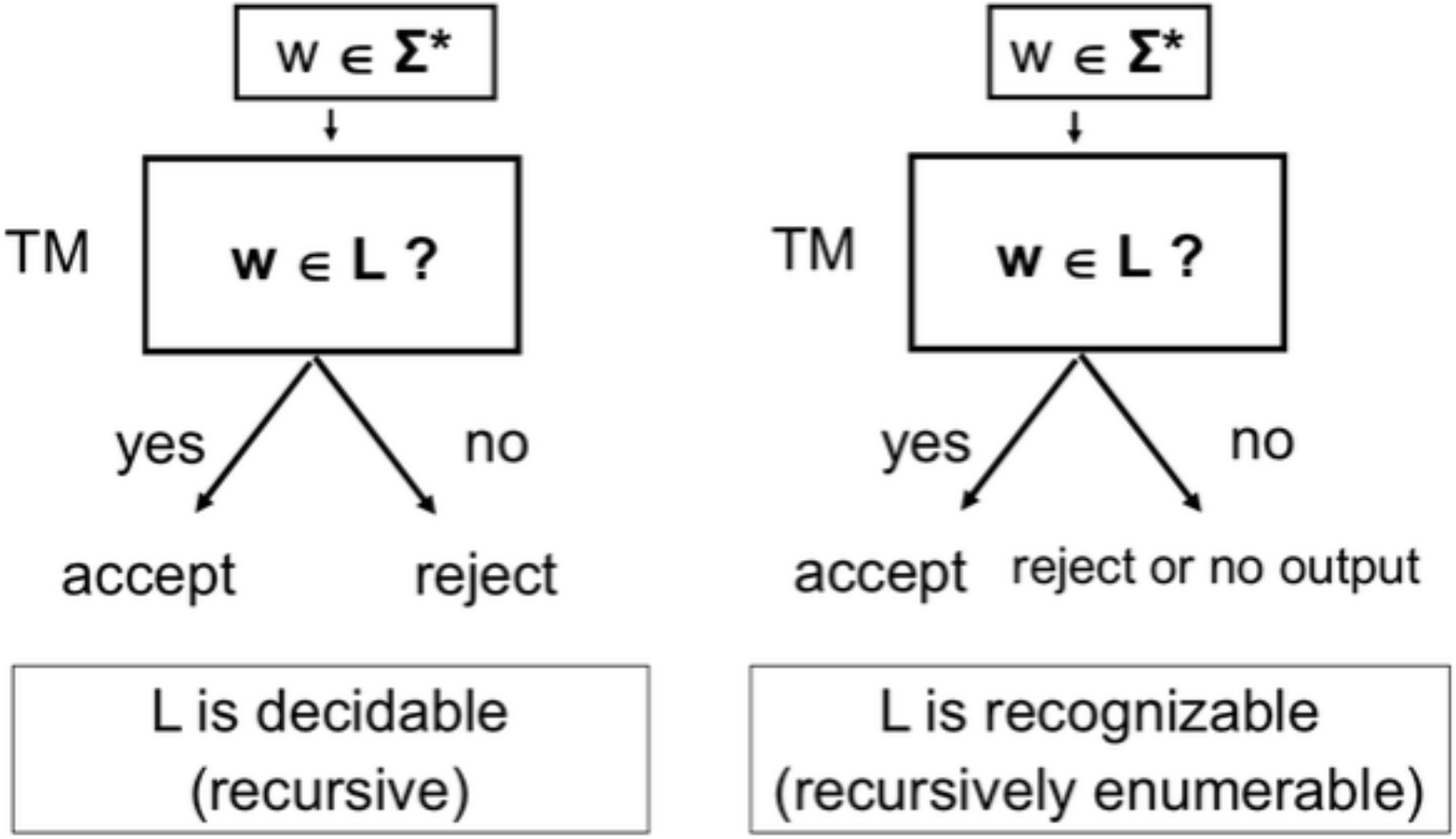
  - Question: Is L(M) recursive?
  - Answer: Yes! By definition it is!

  - Question: Is L(M) in r.e – r?
  - Answer: No! It can't be. Since *M* always halts, L(M) is recursive.

Recognition vs. decision

- A TM **recognizes** a language if it accepts all and only those strings in the language.
- A language is called recognizable or recursively enumerable, (or r.e.) if some TM recognizes it.
- A TM **decides** a language if it accepts all strings in the language and rejects all strings, not in the language.
- A language is called decidable (or recursive) if some TM decides it.

Recognition vs. decision



L is decidable (recursive)

L is recognizable (recursively enumerable)

Recognition vs. decision

- L is Turing-recognizable if some Turing machine recognizes it
- L is Turing-decidable if some Turing machine that is a decider recognizes it. M is a decider TM if it halts on all inputs

Closure Properties of Recursive and RE Languages

- Both closed under union, concatenation, star, reversal, intersection, inverse homomorphism.
- Recursive closed under difference, complementation.
- RE closed under homomorphism.

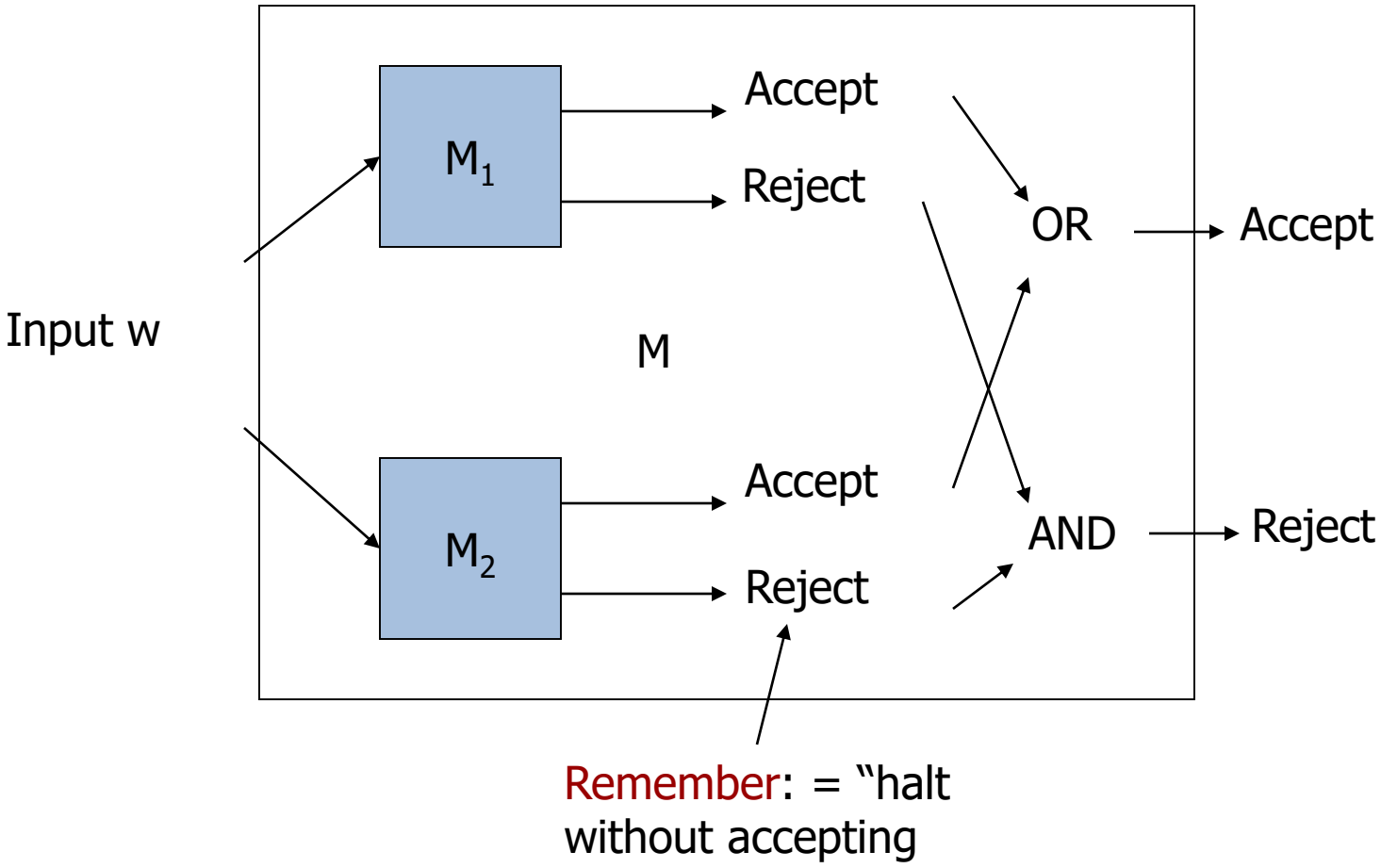
Union

- Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$.
- Assume M_1 and M_2 are single-semi-infinite-tape TM's.
- Construct 2-tape TM M to copy its input onto the second tape and simulate the two TM's M_1 and M_2 each on one of the two tapes, "in parallel."

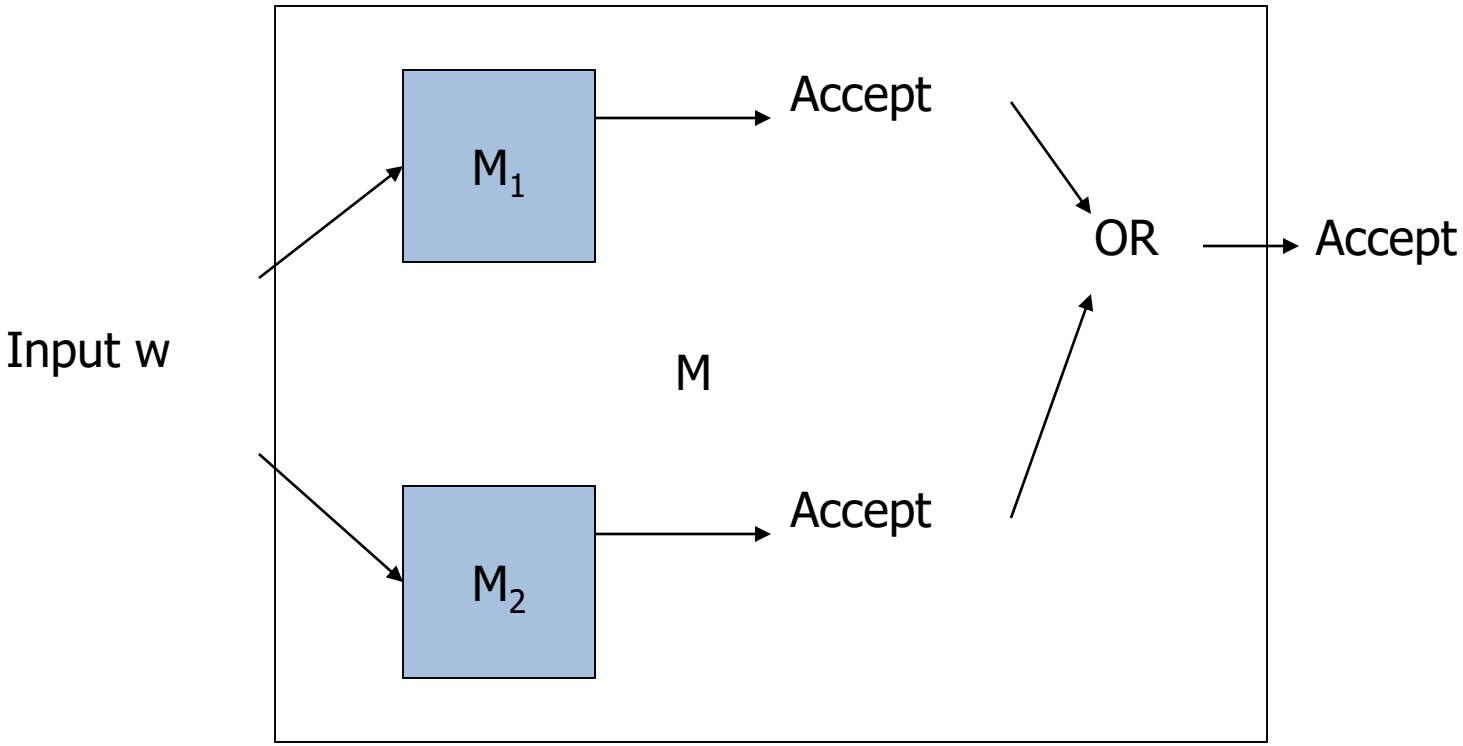
Union – (2)

- Recursive languages: If M_1 and M_2 are both algorithms, then M will always halt in both simulations.
- Accept if either accepts.
- RE languages: accept if either accepts, but you may find both TM's run forever without halting or accepting.

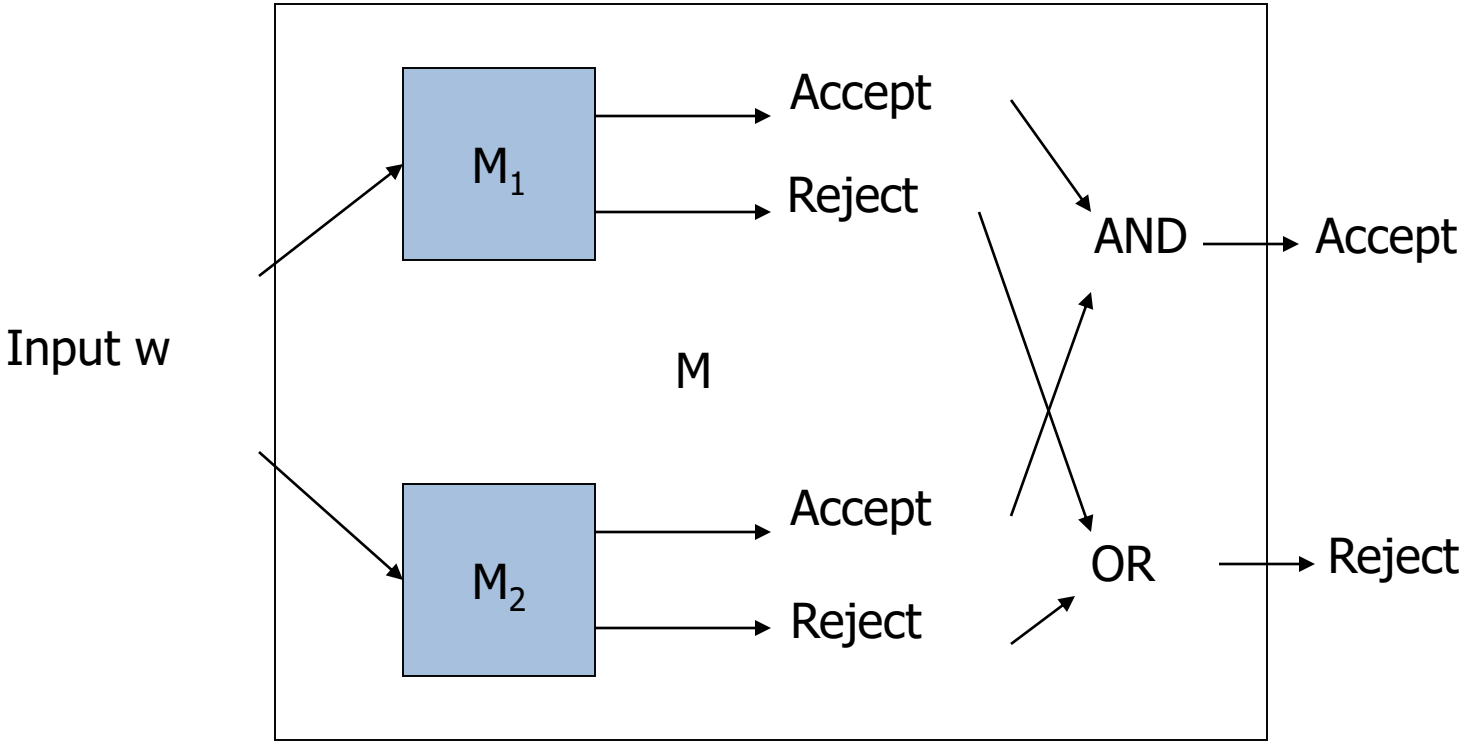
Picture of Union/Recursive



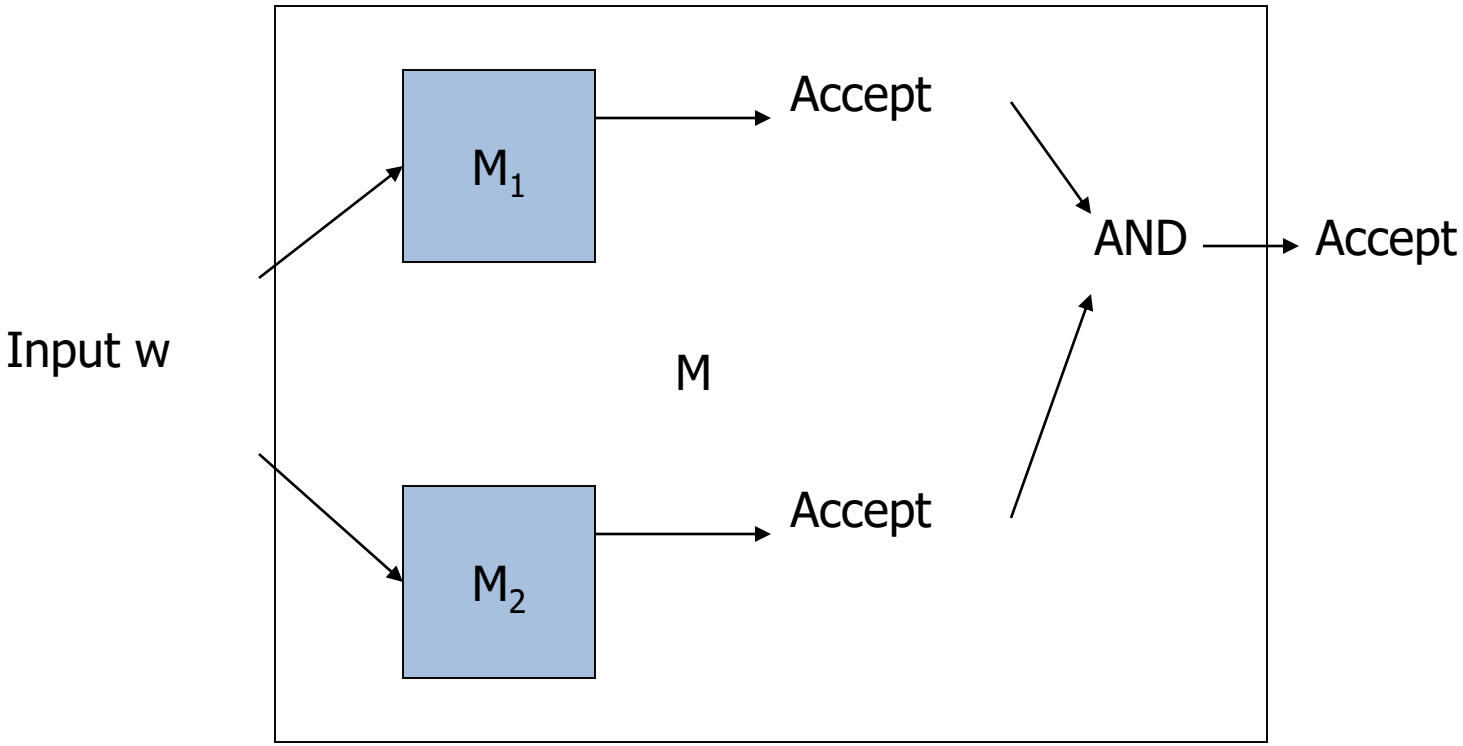
Picture of Union/RE



Intersection/Recursive – Same Idea



Intersection/RE



References

- John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman, *Introduction to Automata Theory*, Languages, and Computation, Pearson, 3rd Edition, 2011.
- Peter Linz, An Introduction to Formal Languages and Automata, Jones and Bartle Learning International, United Kingdom, 6th Edition, 2016.