Cpts 515. 10/12/2020.

Today: Automata Approach

Background: Undergraduate Cpts 317.
(Automata & Formal Languages).

1.0. Automata: Basic Concept.

Automaton: anything that's moving.
(and not moving).
← dead automaton.

So, it's <u>universal</u>.

History: 1st Automaton = Turing machine.

However, TMs are too powerful.
← Halting is undecidable.

①, In reality, need weak model so that many good ~~perties~~ properties can be decided using Algorithms;

②, We need study a hierarchy of such models.

- Finite Automata ⟸ most useful.

Pushdown Automata

⋮

③. Finite Automata can be represented in many ways:

Any C-program that fixed and finite Amount of memory is a Finite automaton.

Example: int x, y;

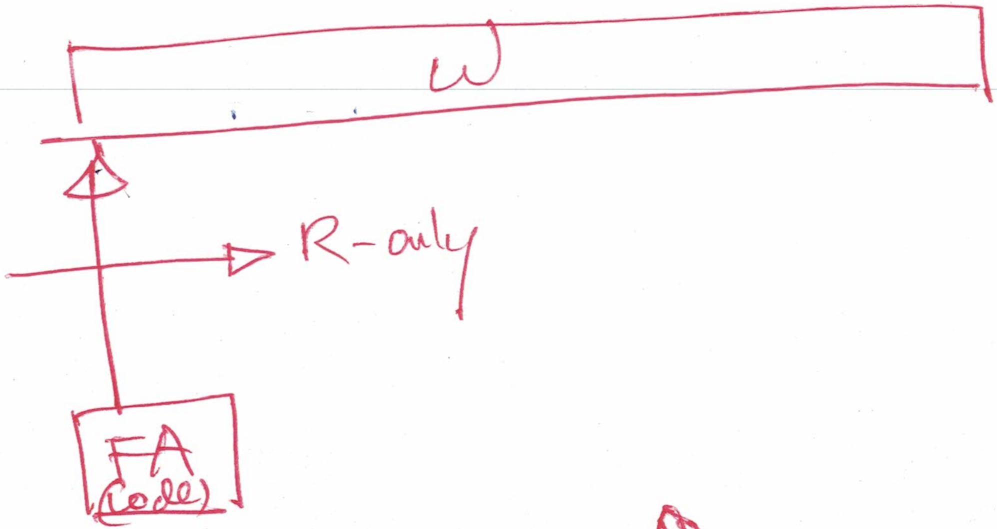
Unbounded memory.

⋮

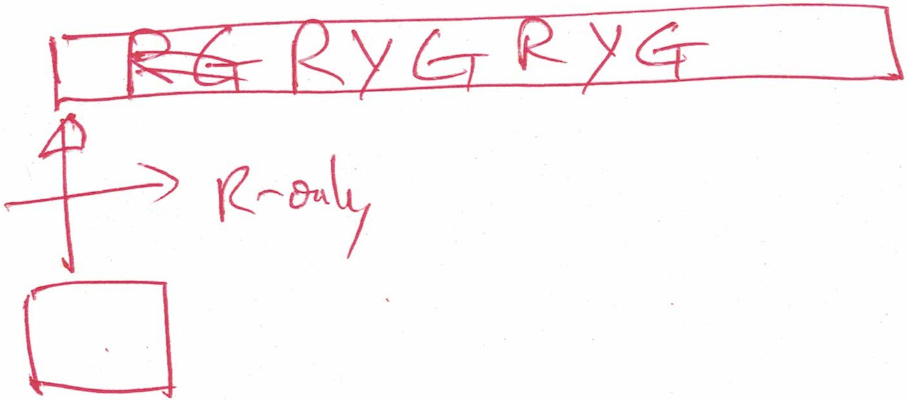
bool x, y;

finite OR bounded memory

One way to draw a FA:



Exaple: describe a FA to accept $(RYG)^*$

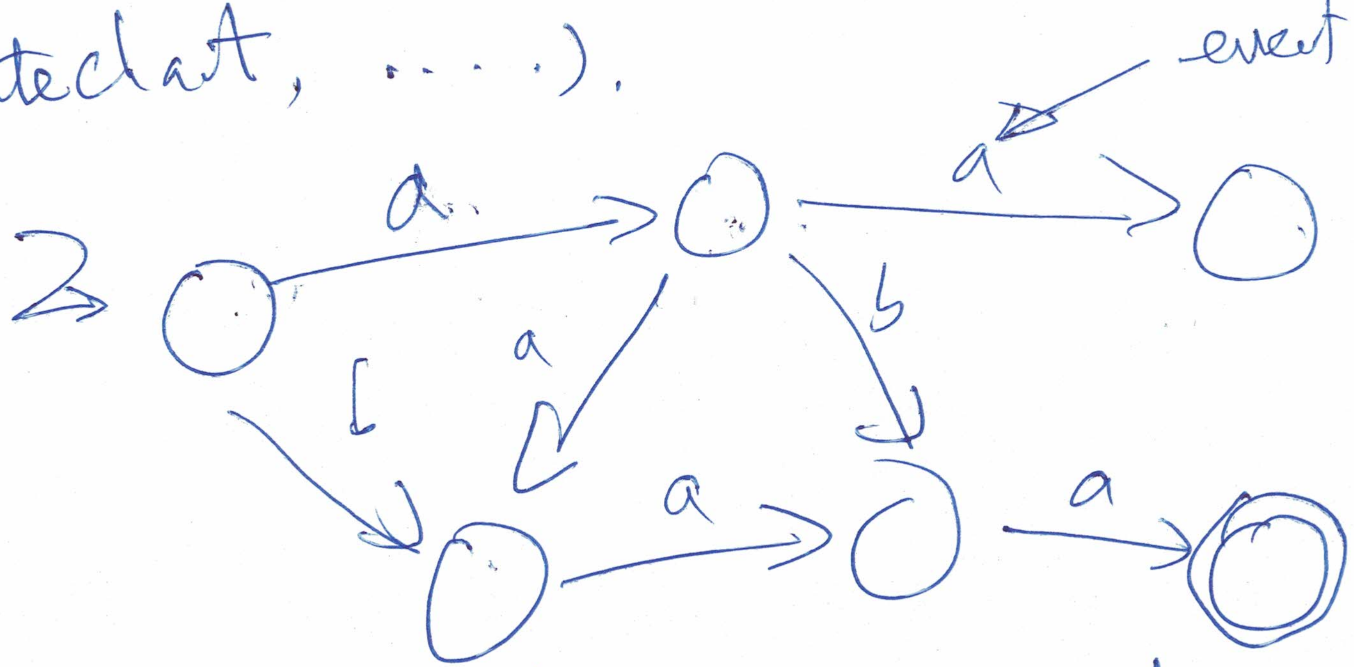


First I read R,
then I read Y,
then I read G,
Repeat ....

how much memory?

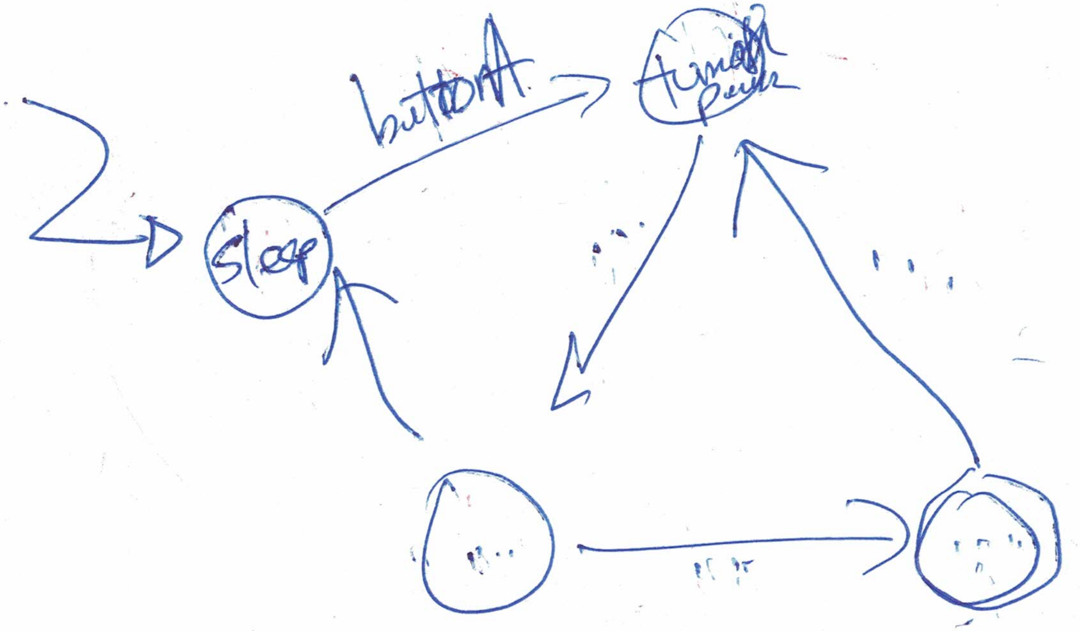
The memory used here is to tell one of three colors. $\log_2 3$. bits.

Another way to describe FA: a graph representation.
(state - transition diagram; finite state transition system)
(statechart, ...).



two versions; det. vs. nondet.

(A good name shall be: labeled transition system, LTS).



Coffee maker.

— how much memory used?

$\log_2 4$ bits. why? you only need to remember one of 4 states, at any time.

which? current state.

Events don't take mmry!

Go back to SCC:

↑ loop-analysis of a graph

---

logic thinking:

C-programs (with finite memory) → Finite automata →

labeled transition → a graph.

From this chain:

if you want an alg to solve problems on C-programs, we have a new approach now:

(if it runs on finite-memory) translate the program into a graph and use known graph alg

(Vardi & LTL): uses SCC to solve a program's liveness query:

□, ◇, ○, ...

Nexttime: ideas in using negation in automata.