Cpts 515. 10/14/2020.

Today: Midterm Exam: take home,
3 days.
Open book/note/internet.
Monday, early morning — Wed. midnight

this coming weekend: I post a video on midterm.

For FA M, and a regular property P, the approach is translated into

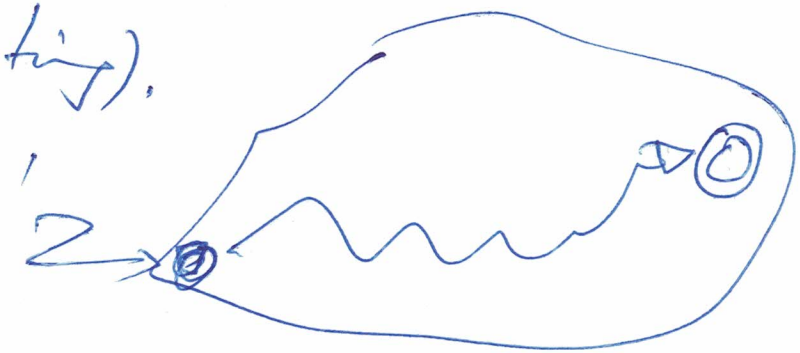
$$\mathcal{L}(M) \cap \neg P$$

Complement of a reg. language is regular.

$$= \mathcal{L}(M') = \emptyset \; ?$$

where: M' is a FA constructed as the Cartesian product of M and an automaton accepting $\neg P$.

And checking "$\mathcal{L}(M') = \emptyset$?" is equivalent to using DFS on the graph of M' from initial to accepting and see whether M' has such a path (from initial to accepting).



In reality,

a program ⟶ a finite automaton M (a graph).

the property L ⟶ halting, a specific statement is executed, etc.

This is hard; you shall look at CFG, Design of L, etc to obtain the graph.

② Astraction from program to automaton

In a program, we have two things to think of:

① PC. (line #)

② local vars. (memory).

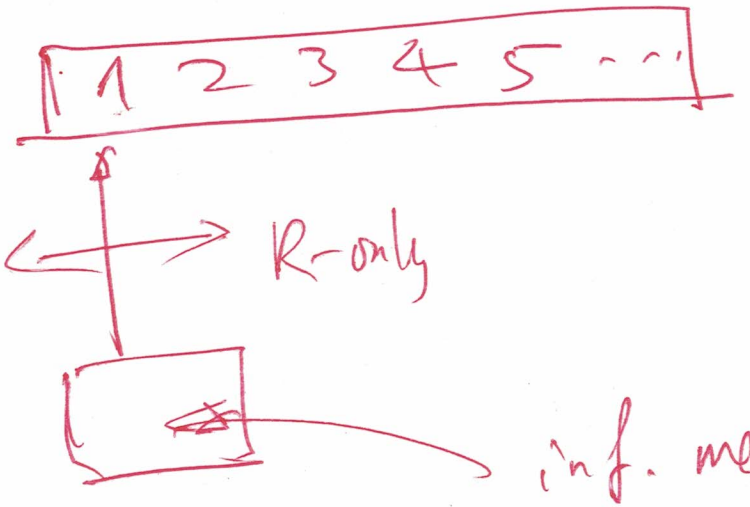
Example.

```
GCD (x, y) {
1   while (x≠y) {
2       if x > y then
3           x := x-y
4       else y := y-x;
5   }
6   ret ....;
7 }
```

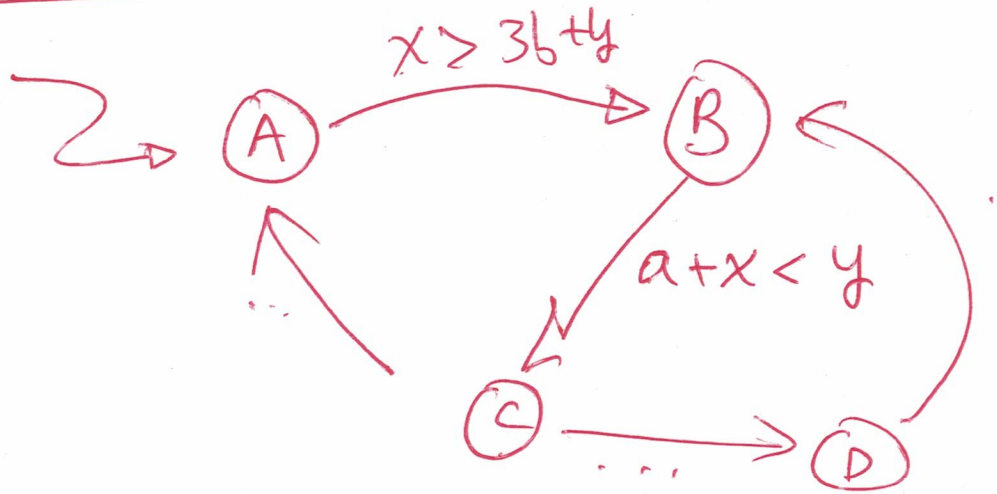
A. local vars : x, y —— unbounded memy.

The program is NOT a FA;

B. The line #'s:



UML:



A:
B: Do sth
C: on x, y.
D:

It's also infite states.

1.2. In some cases, an infinite automaton (program) can be abstracted as a finite automaton.

Example. Timed Automata (Rajeev Alur) — this is to model finite state real time systems.

M is a timed automaton:

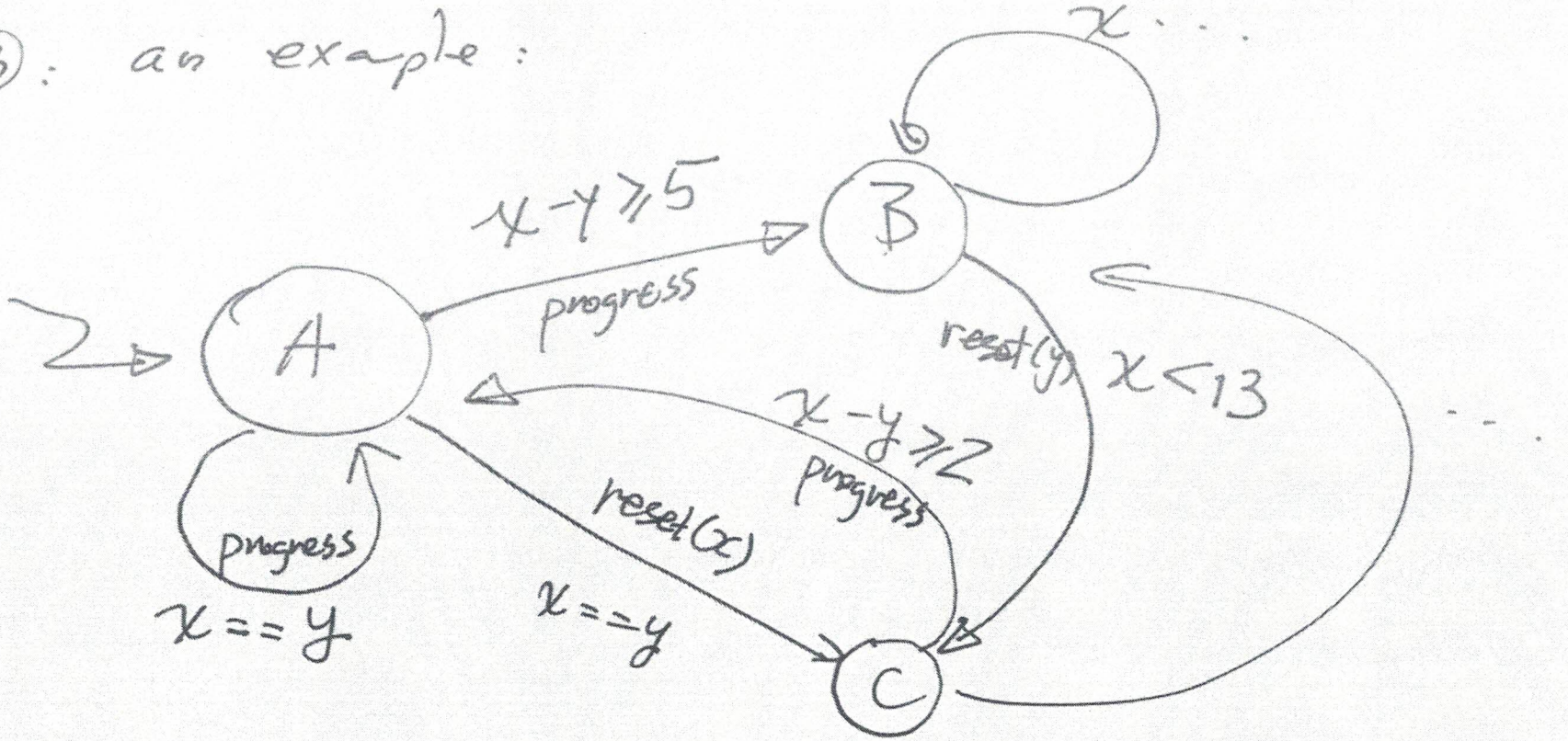
① M has a number of clocks: x, y, z, ...
Each takes nonnegative integer vals.
(discrete time model).

② When M makes a move from one state to another,

(2.1). if the move is "progress": all clocks are inc. by 1;

(2.2). if the move is "reset": a designated clock is reset to val 0, all others no change.

③: an example:

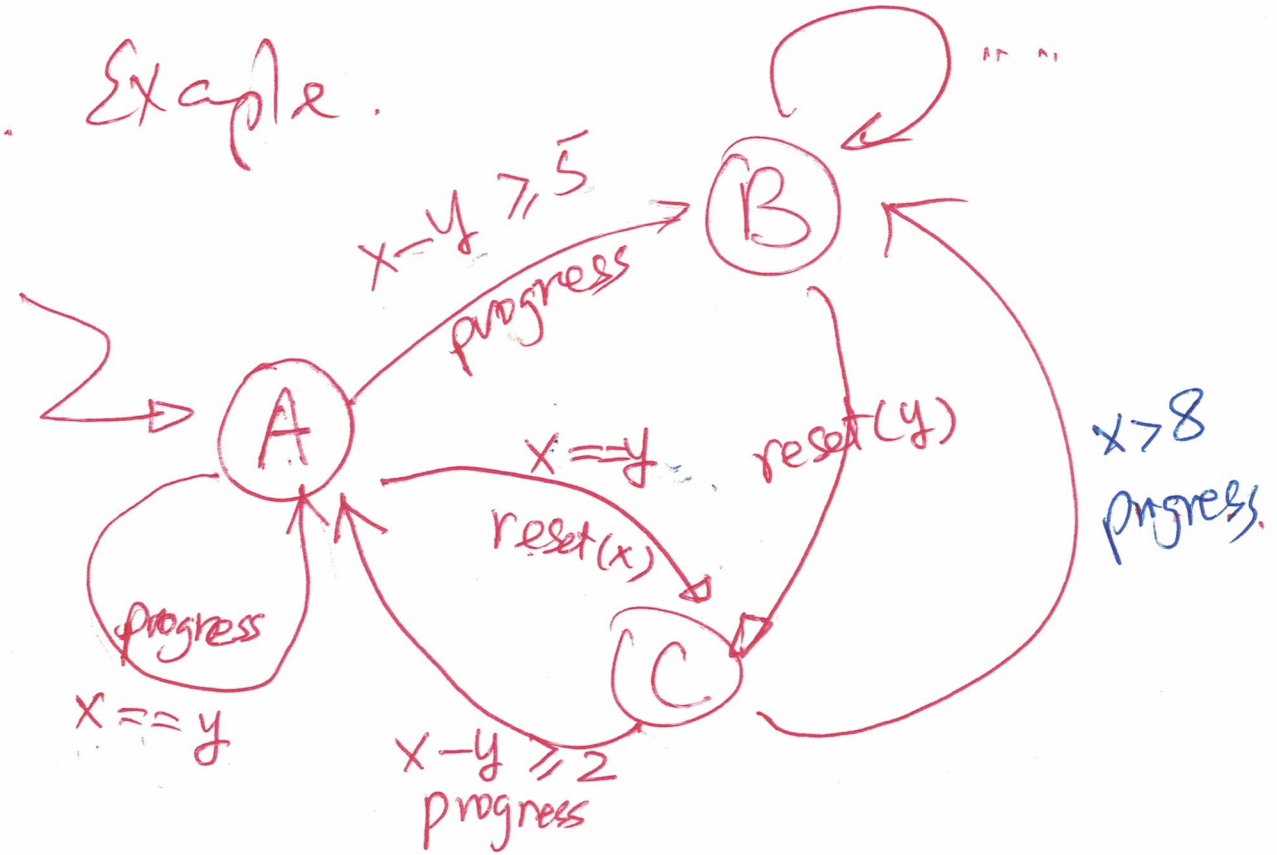


Each transition or move is labeled by a guard or condition in the form of Boolean combination of

$$x - y \# c \quad , \quad x \# c$$

(ex: $(x - y > 5 \vee y - z < 18 \vee z \geq 3) \wedge (y - z < 16)$)

③. Exaple.



all the triggering (enabling) conditions are of the following forms:

$x - y \; \# \; c$, // Comparing two clocks' difference to a constant

$x \; \# \; c$ // Comparing one clocks val to a constant

where $\# \in \{>, <, \geq, \leq, ==, =/=\}$.

M is essentially an inf. state system (because of the clocks which can go unbounded).

Translate M to a FA !

① Region Technique (AlurR). — does not maintain only clock values.

② by me. (Finite table construction).

M = FA + ridiculous simple "clock" structure.

↑ keep transition structure.

↑ keep clock only values.

(Dang (2003): Theoretical Computer Science 302 (2003). 93–121.)

Maintain a table with finite values: // they are variables

$$a_{ij}, \quad b_i : \quad 1 \le i, j \le k,$$

where M has clocks $x_1, \ldots, x_k$.

Let m be the max constant in all clock comparisons in M.

in triggering condition.

Translation:

progress mode:

$$a_{ij} := a_{ij} \quad // a_{ij} \text{ is to encode } x_i - x_j$$

$$b_i := b_i \oplus 1 \quad // b_i \text{ is to encode } x_i.$$

↑ add 1 when b_i is not bigger than m.

reset move: (assume that λ is the set of clocks reset on the move)

$a_{ij} := 0$ if $x_i, x_j \in \lambda$

$a_{ij} := -b_j$ if $x_i \in \lambda$, $x_j \notin \lambda$,

$a_{ij} := b_i$ if $x_i \notin \lambda$, $x_j \in \lambda$,

$a_{ij} := a_{ij}$ o.w.

---

after the reset move:

$b_i := b_i$ if $x_i \notin \lambda$

$b_i := 0$ o.w.

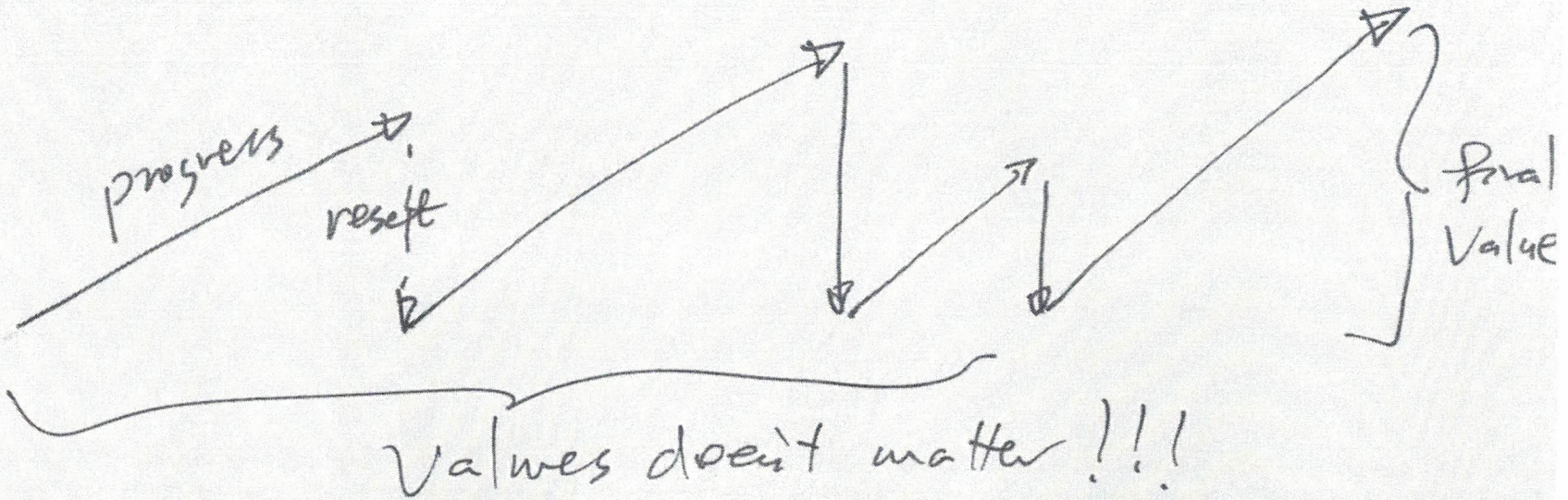
So far:

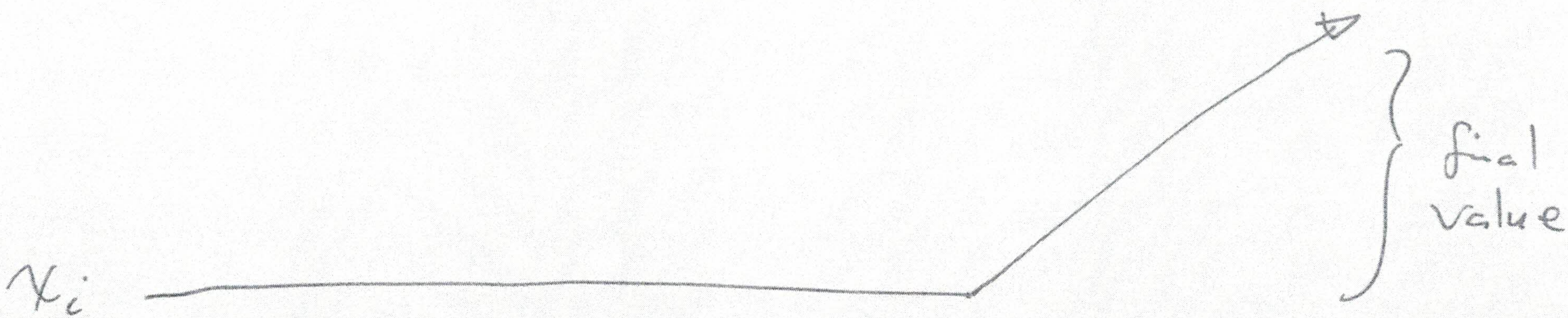
Clock-constants ⟶ finite state vars a_{ij}, b_i's.

Clock-updates ⟶ No clocks. progress, reset.

Key observation: clock doesnot participate clock constants!

X_i's behavior:





x_i is now monotonic!

Time automata = finite automata + monotonic counters!

Well-studied. (Oscar Ibarra, Ron Book) 1970's

Dense time is more complex but do-able!