Cpts 515. 10/7/2020

Today: Symbolic graph search.

Consider: I want to know whether a piece of data A sat. a property $\mathcal{P}$.

Two approaches:

1. Directly check A wrt. $\mathcal{P}$
2. Indirectly,
   Encode $A \longrightarrow A'$
   Encode $\mathcal{P} \longrightarrow \mathcal{P}'$
   Check A' wrt. $\mathcal{P}'$.

Exapes of "symbolic approach":

(1). A set of integers:

$$1, 2, 3, 4, \ldots, 100$$

$$\equiv \text{"} \forall x \in \mathbb{Z}.\ 1 \le x \le 100. \text{"}$$

(2). From (1), we can use a fomula to describe a set (without explicitly stating all the elements)

(3). "Symbolic" execution of C:

```
if x ≠ 1
    x = y + z;
```

What's the value of x now?

It's "$y+z$".

A universal approach:

A finite set can <u>always</u> be represented as a Boolean formula.

// today's challenge: How about infinite set?

Consider a finite set: {  ,  ,  } . . .

Can be symbolically rep. by

{00, 01, 10}.

// We use two Bool vars x_1, x_2
// to denote an elemt in
// the set

$\overline{x_1} \vee \overline{x_2}$.

// this fomla's sat. assygnets form exactly // the above set.

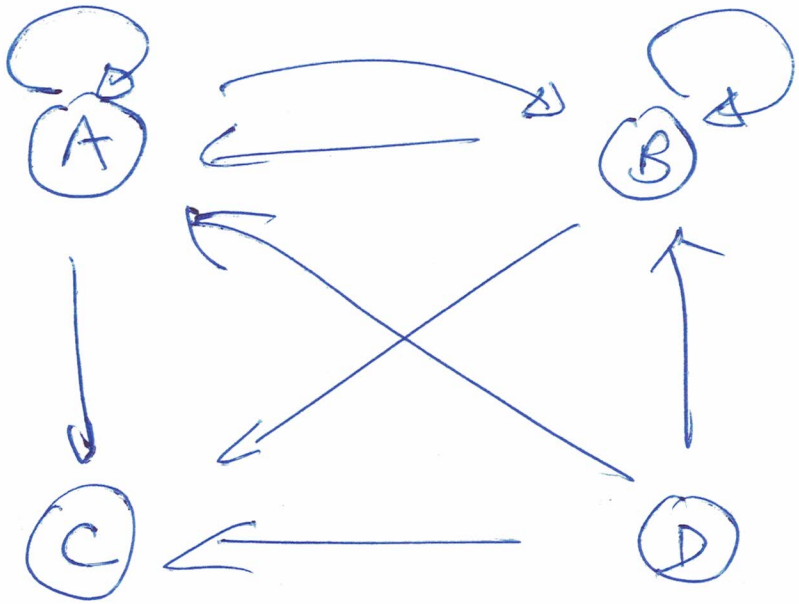
Remark: the representation is not unique. But at least, every finite set can be rep. symbolically.

Graphs are a universal data structure in CS.

This lecture: graph ⟶ Boolean formula.

(Ed. Clarke, McMillan, Bryant).

Look at a small graph:



Encoding:

A ⟶ 00

B ⟶ 01

C ⟶ 10

D ⟶ 11.

} values of two Boolvars x_1, x_2.

Encoding (of edges):

edge 

$x_1=0,\ x_2=1,\ y_1=0,\ y_2=0.$

$\bar{x}_1 \wedge x_2 \wedge \bar{y}_1 \wedge \bar{y}_2$, defined as R_{BA}.

So, each edge is encoded as a Boolean formula over 4 vars: x_1, x_2, y_1, y_2.

I have 9 edges. So, I obtain 9 Boolean formulas; all of them are on vars x_1, x_2, y_1, y_2:

R_{AA}, R_{AB}, R_{BA}, R_{BB}, R_{AC}, R_{BC},

R_{DC}, R_{DB}, R_{DA}.

The Boolean formula for the graph is:

$R_{AA} \vee R_{AB} \vee R_{BA} \vee B_{BB} \vee R_{AC} \vee$

$R_{BC} \vee R_{DC} \vee R_{DB} \vee R_{DA}$;

which is defined by $R(x_1, x_2; y_1, y_2)$.

Infact, $R(x_1, x_2; y_1, y_2) \equiv$

$$(\bar{x}_1 \vee x_2) \wedge (\bar{y}_1 \vee \bar{y}_2).$$

Why this rep. works ?

① We hope that if G is a huge graph, and G has some inherent regularity, then the resulting Boolean formula R can be simplified into a succinct form.

② if G is a random graph, the approach can't work.

"huge graph":

Let G be a graph with 2^{10000} nodes.

how big? if each atom in the universe is a node, the whole Universe fm a graph smaller than the G.

⇒ No algorithm can run on this kind of "huge" graphs.

$G \longrightarrow R$

we hope R is small.

graph search.

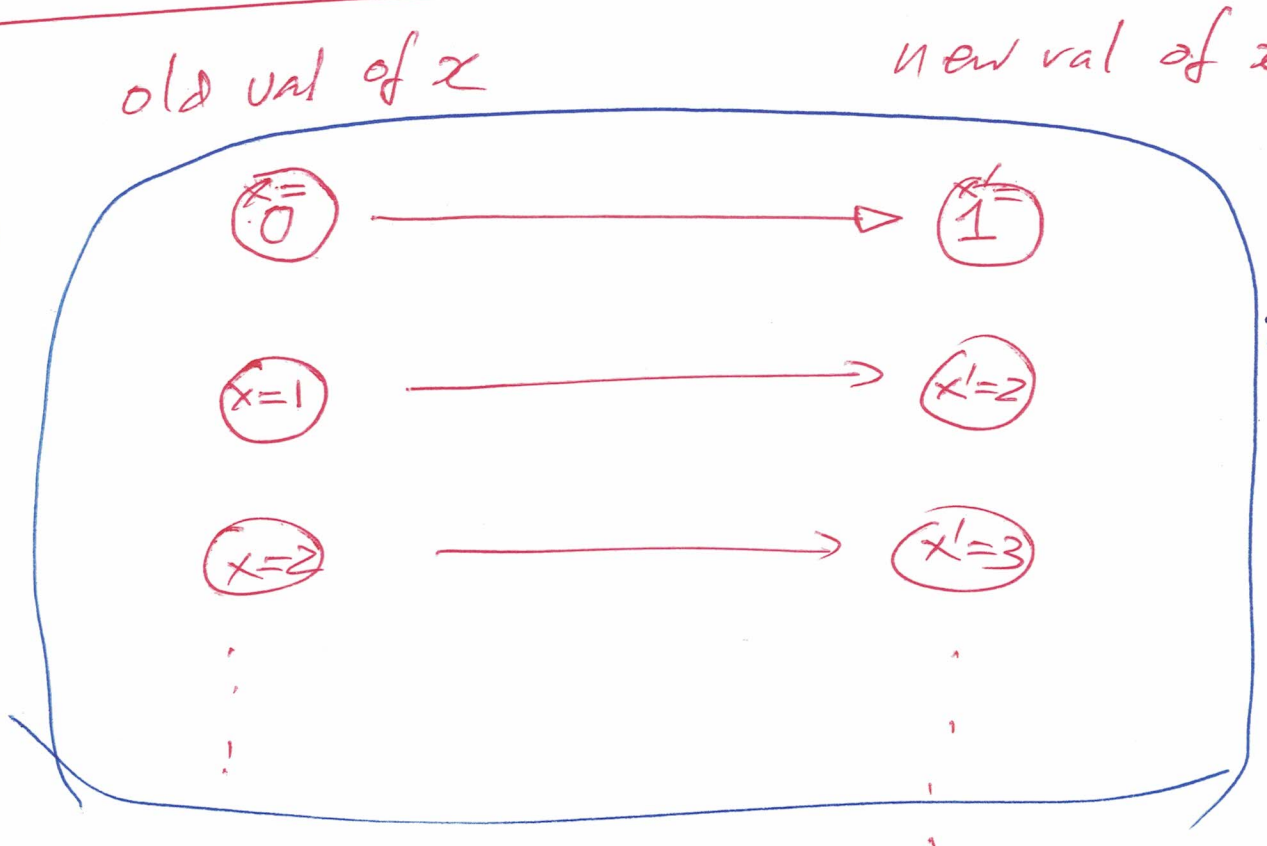
Then we do "search" on R

Two questus to answer:

1. In CS, do we have these huge graphs?

Exaple.

```
x := x+1;   // x is of 32-bits unsigned.
```



# of nodes
$= 2 \cdot 2^{32}$
$\approx$ 10 bln.

= is one graph for rep. the instruction x := x+1,

Consider a C-program running over an array of 100 integers. // each int is of 32-bits.

This C-program can be rep. as a graph with at least $2^{100 \cdot 32}$ nodes.

In other words, a C-program running over k bits memory can be rep. as a graph with at least 2^k nodes.

2. How to operate the R instead of searching on G ?