LESSON 7: Symbolic Graph Search

Consider: I want to know whether a piece of data A has property p.

Two approaches:

(1). Directly check A wrt. p

(2). Indirectly.

Encode $A \longrightarrow A'$

Encode $p \longrightarrow p'$

Check A' wrt p'.

Example:

(1). Rep. of a set of integers

$1, 2, \ldots, 100 \equiv \forall x^{\in \mathbb{Z}} . 1 \leq x \leq 100$

(2). Need a "general" approach in symbolically represent data

"Symbolic" Execution of C.

(3).

if $x \neq 1$

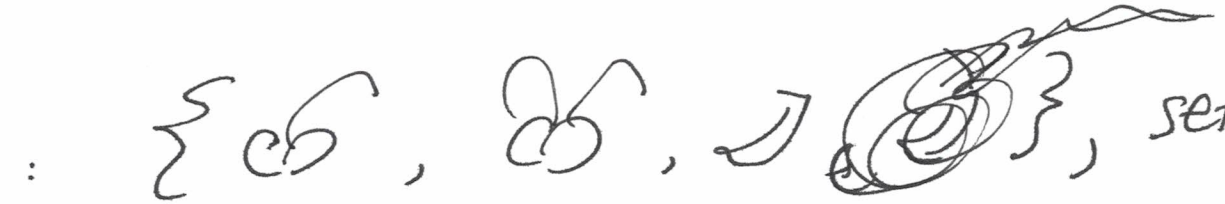
$x = y + z$;

what is the val of x now?

It's "$y + z$" symbolically.

A finite set can always be represented as a Boolean formula.

(Question: How about infinite sets?)

Consider:  , set of three.

⇓ encoding

$\{00, 01, 10\}$, vals of two Boolean vars x_1 and x_2

⇕ satisfying assignments of

$\bar{x}_1 \vee \bar{x}_2$

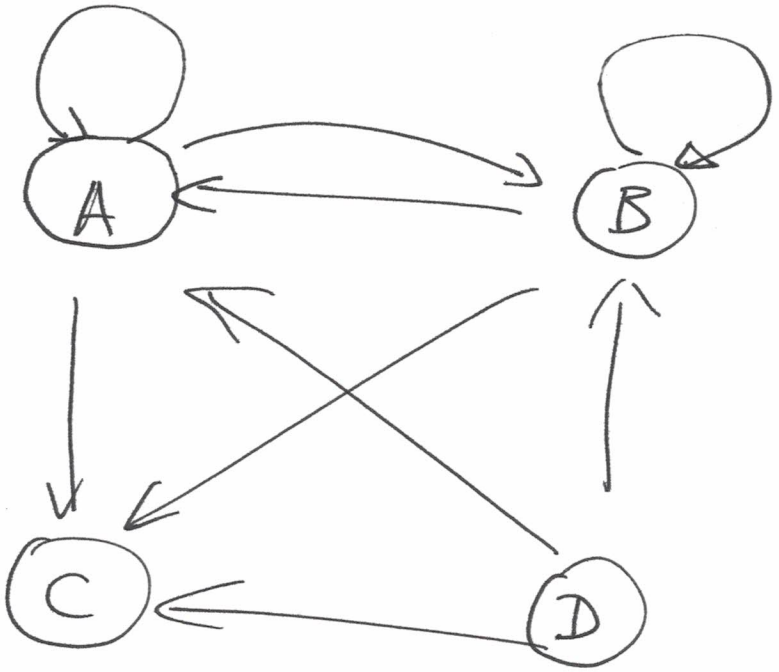
is symbolic representation of

Remark: The encoding is not unique. But at least gives some hint at how to encode a data set, symbolically.

This lecture: graph $\longrightarrow$ Boolean formula.
(due to McMillan, Clarke, Bryant at CMU)
in 1980 ~ 1990's.

A little story of McMillan.

Given a graph G



Encoding:

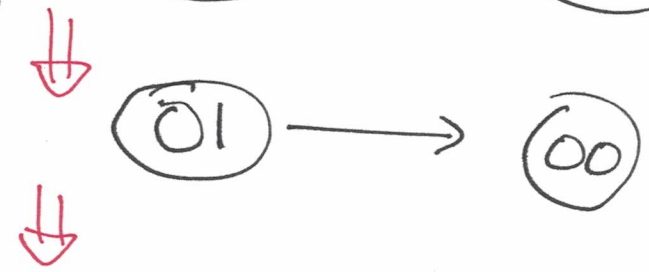
A ⟶ 00
B ⟶ 01
C ⟶ 10
D ⟶ 11

} vals of two Boolean vars x_1, x_2.

For each edge, say 

,

we have:



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

So, each edge is a Boolean formula

$R_{BA} \equiv$ $\bar{x}_1 \wedge x_2 \wedge \bar{y}_1 \wedge \bar{y}_2$

name of the Boolean formula for the edge BA.

Similarly, we can obtain 9 Boolean formula's for the 9 edges:

R_{AA}

R_{AB}

R_{BB}

R_{BA}

R_{AC}

R_{BC}

R_{DC}

R_{DB}

R_{DA}

Similarly, we can obtain 9 Boolean fomla's for the 9 edges:

R_{AA} V

R_{AB} V

R_{BB} V

R_{BA} V

R_{AC} V

R_{BC} V

R_{DC} V

R_{DB} V

R_{DA}

This is R, the boolean fomla of the graph.

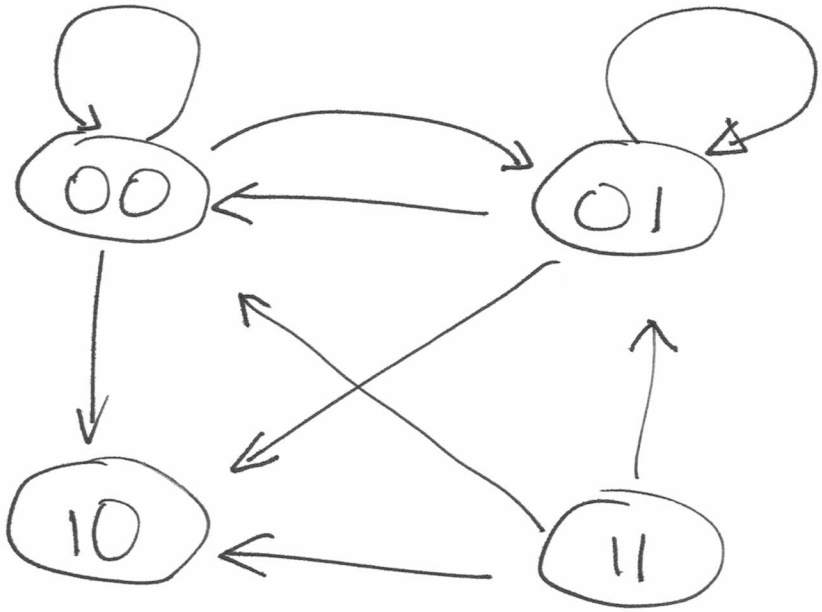
R is super long formula, BUT it can be simplified into

Boolean formula R on 4 Boolean var's

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

Truth table:

x_1	x_2	y_1	y_2
0	1	0	1
0	1	0	0
0	1	1	0
0	0	0	1
0	0	0	0
0	0	1	0
1	1	0	1
1	1	0	0
1	1	1	0



Graph $\Longleftrightarrow$ Truth Table

Summary:

Graph G with 2^k nodes

Boolean formula R with $2k$ var's

$R(x_1, \ldots, x_k ; y_1, \ldots, y_k)$

Meaning: $R(x_1, \ldots, x_k ; y_1 \ldots y_k)$ true

iff $(x_1, \ldots, x_k) \longrightarrow (y_1, \ldots, y_k)$ is an edge in G.

two nodes.

Symbolic Graph = Bool formula Rep. of the Graph.

Symbolic Graph Search.

Given: two nodes $A = (s_1, .., s_k)$

$B = (t_1, .., t_k)$

a graph G rep. by a Bool formula

$R(x_1, \ldots, x_k; y_1 \ldots y_k)$.

Question: Can A reach B in G ?

Called "Reachability".

In real world,

G ⟶ software system

A ⟶ starting state

B ⟶ "bad" state like crash, can Assert, ...

Step 1. Calculate transitive closure R^*

$H := R$;

Repeat

$H' := H$;

$H := H' \vee \text{Compose}(H', R)$;

Simplify (H);

Until Equivalent (H, H')

Return H as R^*.

Step 2. Check $R^*(s_1, \ldots, s_k; t_1, \ldots, t_k)$.

Remark

Compose (R_1, R_2) often written as $R_1 \circ R_2$

Its meaning is:

$$(R_1 \circ R_2)(x_1, \ldots, x_k ; y_1, \ldots, y_k) =_{def}$$

$$\exists z_1, \ldots, z_k : \quad R_1(x_1, \ldots, x_k ; z_1, \ldots, z_k) \wedge$$
$$R_2(z_1, \ldots, z_k ; y_1, \ldots, y_k).$$

① $R_1 \circ R_2$ is a Boolean formula on $2k$ var's.

Since Boolean formulas are closed under $\exists$-quantifier.

② R_1 and R_2 are Boolean ~~var's~~ formulas on $2k$ var's.

What is $R \circ R$?

What is $R \circ R \circ R$?

$\vdots$

$(R \circ R)(x_1, \dots, x_k ; y_1, \dots, y_k)$

(a node) (a node) (a node)

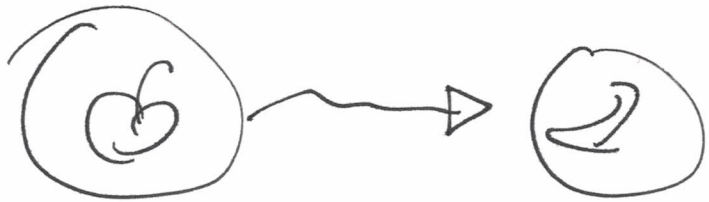
$\equiv \exists z_1, \dots, z_k :$

$R(x_1, \dots, x_k ; z_1 \dots z_k) \wedge R(z_1, \dots, z_k ; y_1 \dots y_k)$

$\exists$: $R(\,,\,) \wedge R(\,,\,)$

$\equiv$ in G,

$\exists$:

$\equiv$ in G,  in two steps!

The transitive closure alg computes

$$R^* \equiv \left\{ \begin{array}{l} R \cup \\ R \circ R \cup \\ R \circ R \circ R \cup \\ \vdots \end{array} \right\} \iff$$

in English, Bool fmla rep. for all state/node pairs that one node can reach the other in m steps, for some $m > 0$.

BUT: m is bounded essentially, since we have only finitely many nodes.

{ (Ex: in a graph of 8 nodes. node A can reach node B in 1000 steps $\Rightarrow$ $A \rightsquigarrow B$ in $i \leq 8$ steps.)

This is why the alg halts! ←

Remark: $G \longrightarrow R \longrightarrow R^*$

Graph with 2^k nodes

Bool funct with $2k$ var's

Bool funct with $2k$ var's.

Checking $A \rightsquigarrow B$ in G ?

$\equiv$

with $A = (s_1, \ldots, s_k)$

$B = (t_1, \ldots s_k)$

in the symbolic encoding of G into R.

Checking

$R^*(s_1, \ldots, s_k ; t_1, \ldots, t_k)$

Excercise: How to check $A \rightsquigarrow B$ in even number of steps? (I need a symbolic alg).

More remarks:

(1). In reality, Boolean formulas can be further encoded as a Data Structure called BDD (Randal Bryant, 1986)

↑ Boolean Decision Diagram.

BDD is a graph!

(2). BDD library is available in Python.

(3). Nowadays, we can use BDD-based symbolic graphs to encode and reason on Graphs with 2^{1000} nodes.

How many atoms in the universe?

(4). Reachability. is only a simple search.
More complex search using temporal logics to specify the search goals is studied, the whole area is called "symbolic" model checking. See the book of Ed. Clarke on Model-checking.

(5). HW on using the search for software verification.

Idea: Specification sat. Requirement

Specification → Graph G

Requirement → Property on Graph P

Then using model-checking algs to do "search" P on G and make sure that P is true on every node of G.

The search is done symbolically.

Side Note: some infinite states can be encoded using linear constraints. $\Rightarrow$ Integer Linear Programming.

Further Reading on papers: I gave the links.