Cpts 515. 10/9/2020.

A graph $\longrightarrow$ A Boolean formula:

Let G be a graph with 2^k nodes.

$\Updownarrow$

Boolean formula R with $2k$ variables.

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

Meaning of the R:

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

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

two nodes.

Symbolic Graph $\equiv$ the R.

Boolean formula,

How to "Search" on symbolic graph.

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

$B = (t_1, \dots, t_k)$

a graph G rep. by a Boolean formula

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

Question: Can A reach B in G?

Reachability

Real-world: $G \longrightarrow$ Software System

$A \longrightarrow$ starting state

$B \longrightarrow$ "bad" state ("crash")

Remark:

Compose(R_1, R_2), written $R_1 \circ R_2$, is defined as:

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

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

---

1. $R_1 \circ R_2$ is a Boolean formula over $2k$ vars. (Boolean formulae is closed under quantifiers).
2. How about $R \circ R$?

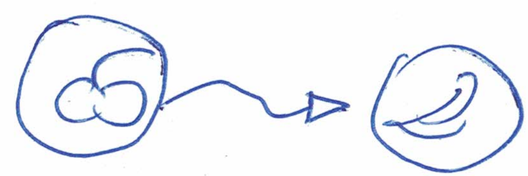
$(R \circ R)(x_1, \ldots, x_k; y_1, \ldots, y_k) \equiv$

$\exists z_1, \ldots, z_k :$

$R(\underbrace{x_1, \ldots, x_k}_{\text{a node 🍎}}; \underbrace{z_1, \ldots, z_k}_{\text{a node 🍒}}) \wedge R(\underbrace{z_1, \ldots, z_k}_{\text{a node 🍒}}; \underbrace{y_1, \ldots, y_k}_{\text{a node 🍌}}).$

$\equiv \exists$ 🍒 : $R($🍎, 🍒$) \wedge R($🍒, 🍌$)$

$\equiv$ in G, $\exists$ 🍒 s.t.

$\equiv$ in G,  in two steps.

step 1. We need compute The Transitive closure

$R^* \equiv$

$$\left.\begin{array}{l} R \ \vee \\ R \circ R \ \vee \\ R \circ R \circ R \ \vee \\ \vdots \end{array}\right\} \begin{array}{l} \longrightarrow \text{reachability in 1 step.} \\ \longrightarrow \text{reachability in two steps} \\ \longrightarrow \text{- - - three steps} \\ \vdots \end{array}$$

$\equiv$ reachability in any steps.

Alg for transitive closure R^*:

$H := R$;

Repeat:

$H' := H$;

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

Simplify(H);

Until Equivalent(H, H');

return H as R^*;

$H = R$

$H = R \cup R \circ R$

$H = R \cup R \circ R \cup R \circ R \circ R$

$\vdots$

$\exists m$ s.t. the H obtained in m-th iteration is equivalent to the H obtained in $(m-1)$-th iteration.

The m always exists and hence the alg will stop. Why? Given the graph G, we have a number m s.t. for all nodes u, v in G,

$u \rightsquigarrow v$ iff u can reach v in m steps.

(m is upper bounded by the # of nodes in G.)

step 3. Return $R^*(\underbrace{s_1, \ldots, s_k}_{\text{a given node}}; \underbrace{t_1, \ldots, t_k}_{\text{a given node}})$.

So far,

$$G \longrightarrow R \longrightarrow R^*$$

G with 2^k nodes; R Boolean formula with $2k$ vars; R^* Boolean formula with $2k$ vars.

---

checking $A \rightsquigarrow B$ in G?

$\equiv$ checking the truth value of $R^*(s_1, \ldots, s_k, t_1, \ldots t_k)$

with
$A = (s_1, \ldots s_k)$
$B = (t_1, \ldots t_k)$
in the symbolic encoding of G into R.

Ex. How to check $A \rightsquigarrow B$ in even # of steps?

$A = (s_1, \ldots, s_k)$, $B = (t_1, \ldots, t_k)$

We compute $\mathcal{R} := R \circ R$; // two-step // reachability

We compute $\mathcal{R}^*$; // even-step // reachability

return $\mathcal{R}^*(\underbrace{s_1, \ldots, s_k}_{A}, \underbrace{t_1, \ldots, t_k}_{B})$;

In real world, further reading:

Ed Clarke's Book. .. Model checking;

top conference: CAV.

Computer Aided Verification.

top conferences in Software Engg: ICSE,

FSE.

Other places that you can read the topic:

(1). Design Automation (Computer Engg.)

(2). testing & verification combined, seen in software testing conferences,

ISSTA, ...

In real world, ① R^*, a Boolean formula, is encoded as BDD. (Randal Bryant 1986).

Boolean Decision Diagram, which is a graph!

② we have "free" BDD libraries available in python. (PYEDA package).

③ "Symbolic Model-checking" ≡ How to Search on a huge graph. Most applications are in Software Engg & Circuit Verification. New applications: other areas?

Today's Challenge:

(Finite) graph $G \Longrightarrow R^*$ in Boolean formula.

How about

infinite graph $G \Longrightarrow$?

(Possible answer: approx. the inf. graph by a finite graph.)

is not a reliable approach since a query in inf. graph $\not\equiv$ a query on finite graph.

you don't read-off internet randomly!

why? the entry-barrier to CAV conference is very-high.

Comunity is of 500 people.

Example in PyEDA package

eight-Queen Puzzle

python code.