Lecture 25

Tuesday, November 17, 2020 4:01 PM

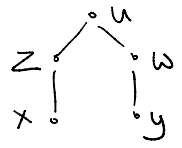
Datalog

Cousins (x y) :- Parent (z x) Parent (w y)

Parent (u z) Parent (u w)

Head

Body



Path (x z) :- Path (x y) Edge (y z)

Path (x y) :- Edge (x y).

"All-pairs reachability"

"Transitive"

$$\frac{a \leq b \quad b \leq c}{a \leq c}$$

Transitive closure: Begin with a seed relation.

"Make it transitive".

- Look for violations of the transitivity rule.
- Whenever $\exists\, x\ y\ z$ s.t

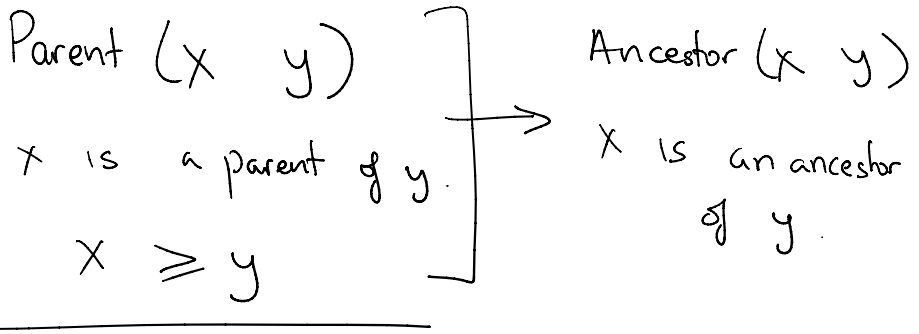
$$x \leq y \ \& \ y \leq z$$

If it is not yet the case

that $x \leq z$,

then just "throw it in".

Add $x \leq z$.



The Parent relation is not transitive

"My grandparent is not my parent."

The Ancestor relation is transitive.

— My grandparent is an ancestor of mine.

— His grandparent is also an ancestor of mine.

Equivalently, the transitive closure of a relation R is the smallest relation R^* s.t. ① $R \subseteq R^*$

② R^* is transitive.

Plan

Plan

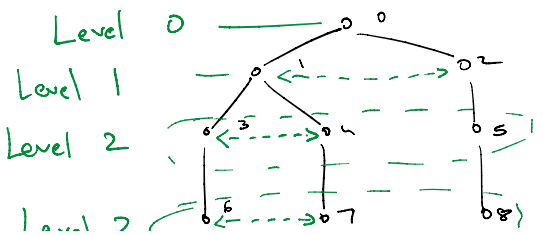
1. Become comfortable with Datalog "Writing queries".
2. Demystify the language Evaluation algorithms
3. "The Problem of Negation".
   - Stratified programs
   - Aggregation

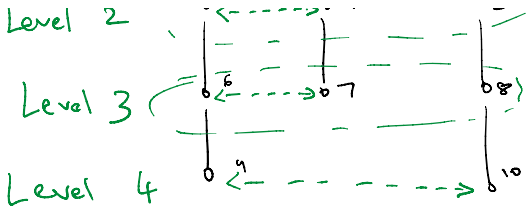
Write a program to find all pairs of people who are in the same generation of the family

---

Parent (x y) = "x is a parent of y"

Samegen (x y) = "x & y are in the same generation"





Rule 1: "We are in the same generation if our parents are in the same gen."

samegen(x y) :- samegen(u v), parent(u x), parent(v y).

Rule 2: Base case.

samegen(x x) :- person(x).

We just invented this relation.

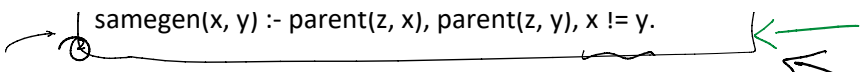
person(x) :- parent(x, _)

person(x) :- parent(_, x).

① samegen(x, y) :- samegen(u, v), parent(u, x), parent(v, y).

② samegen(x, y) :- parent(z, x), parent(z, y), x != y.

Base case of



Base case of the induction.

Alternative program which avoids samegen (x x)

Assume not. Say we have a tuple samegen (x x).

Ask the prosecutor: "Show your evidence!"

- Could the last step in the stack trace be Rule 2?

  No, because $x \neq y$.

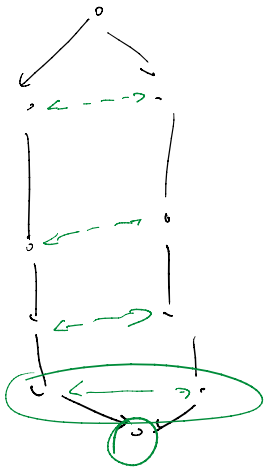
- Could the last step in the stack trace be Rule 1?

  samegen (x y) :- samegen(u v)

  parent (u x) parent (v y), $x \neq y$.

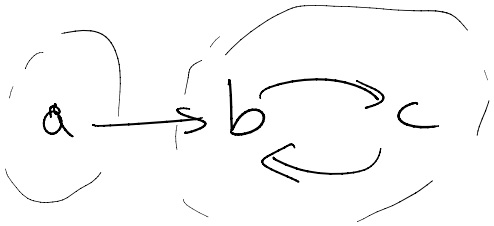
Samegen (x x) $\Leftarrow$ samegen (u v)

Parent (u x) parent (v x)

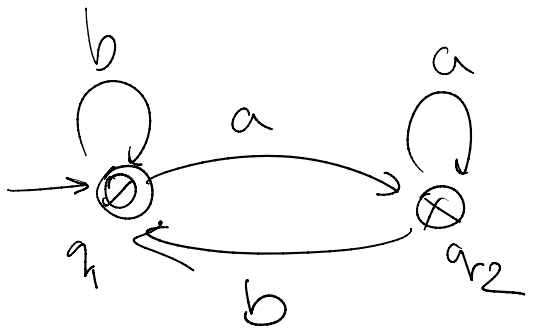


Q2: Find if two nodes in a graph are in the same scc.

Edge	
a	b
b	c
c	b



scc(u v) if there is a path from u to v & a path from v to u.



Write a Datalog program which executes this DFA.

Index	0	1	2	3	4	5	6
Char	a	a	b	b	a	a	b

Negation assumes "closed world".

The closed world assumption states:

"All that is true is known to be true."

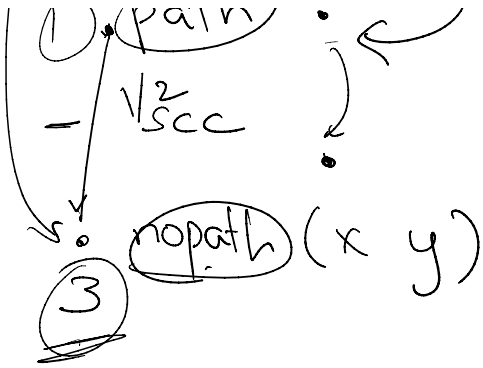
Souffle employs "stratified negation"

.decl vertex
.decl edge
.decl path

path(x y) :- edge(x y)

path(x z) :- path(x y) edge(y z)

1/2



path(x z) :- path(x y) edge(y z)

scc(x y) :- path(x y), path(y, x)

nopath(x y) :- vertex(x) vertex(y)
!path(x y).

Paradox because ! operator in a cycle.

Souffle tries to check if application of ! forms a DAG.