Datalog

- Becoming comfortable with the language

  Writing programs

- The problem of negation

  Stratified negation (Most Datalogs) / Negation-as-failure (Prolog)

  ⇓

  Aggregation

  Stable model semantics (Answer set programming)

- Demystify the magic / Evaluation algorithms

Example (Rule / Clause)

Cousin(x, y) :- Parent(z x) Parent(w y)

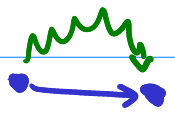
Parent(u z) Parent(u w)

Head | Body

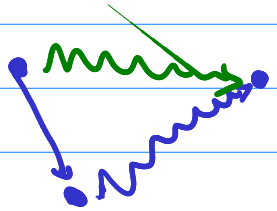
head :- if everything in body ∀xyzwu

Example

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



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



Path(x, x) :- _____

Path(x, x) :- . "true"

true and a and b and c and d

⇓

true and a and b and c

⇓

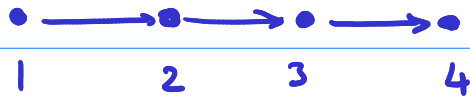
true and a and b

⇓

true and a

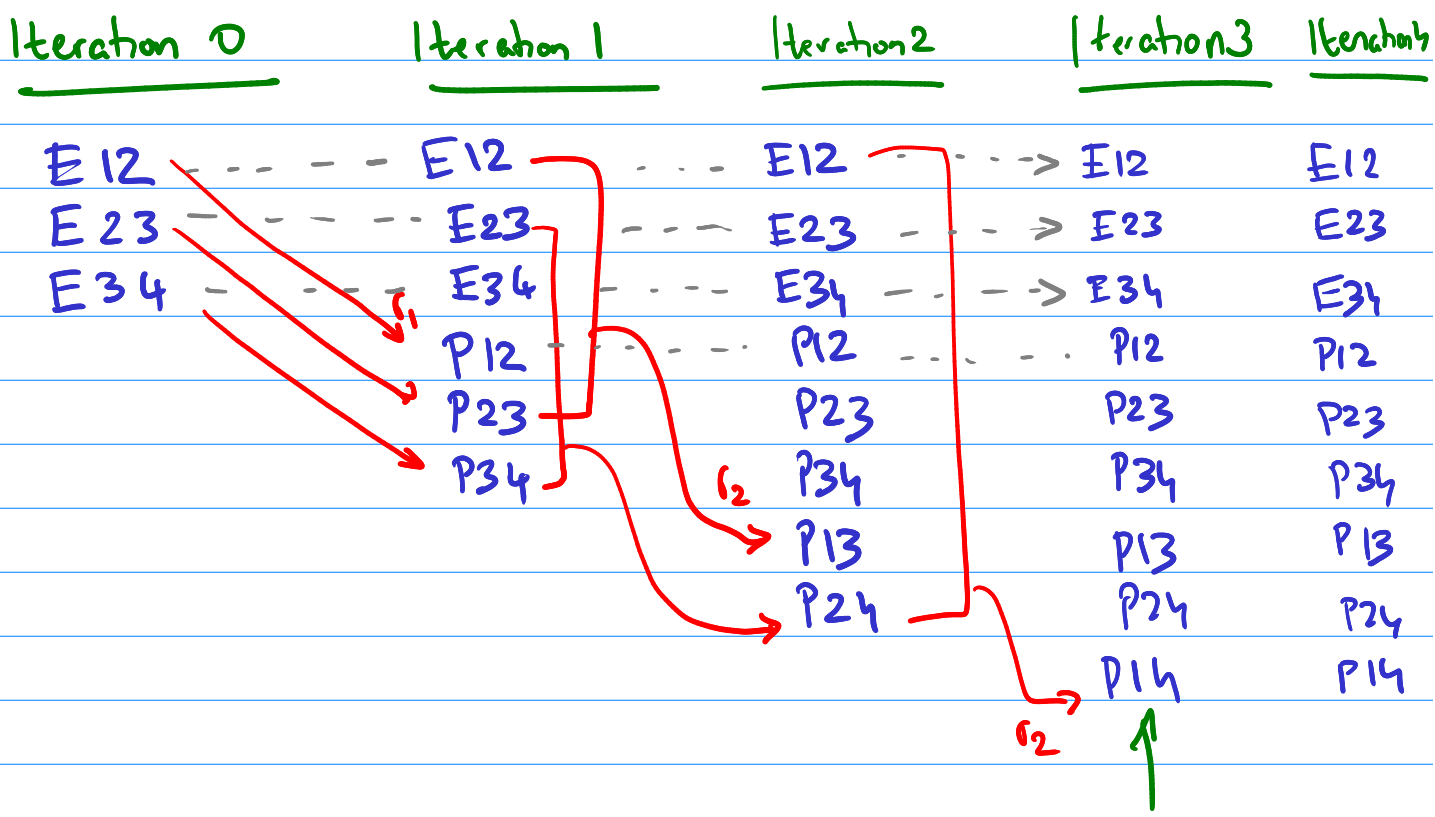
⇓

true and true



Path(x,y) :- Edge(x, y) (r_1)

Path(x, z :- Edge(x,y) Path(y z) (r_2)



From this point, the rules can't derive anything new.

"Fixpoint" / "Least fixpoint"

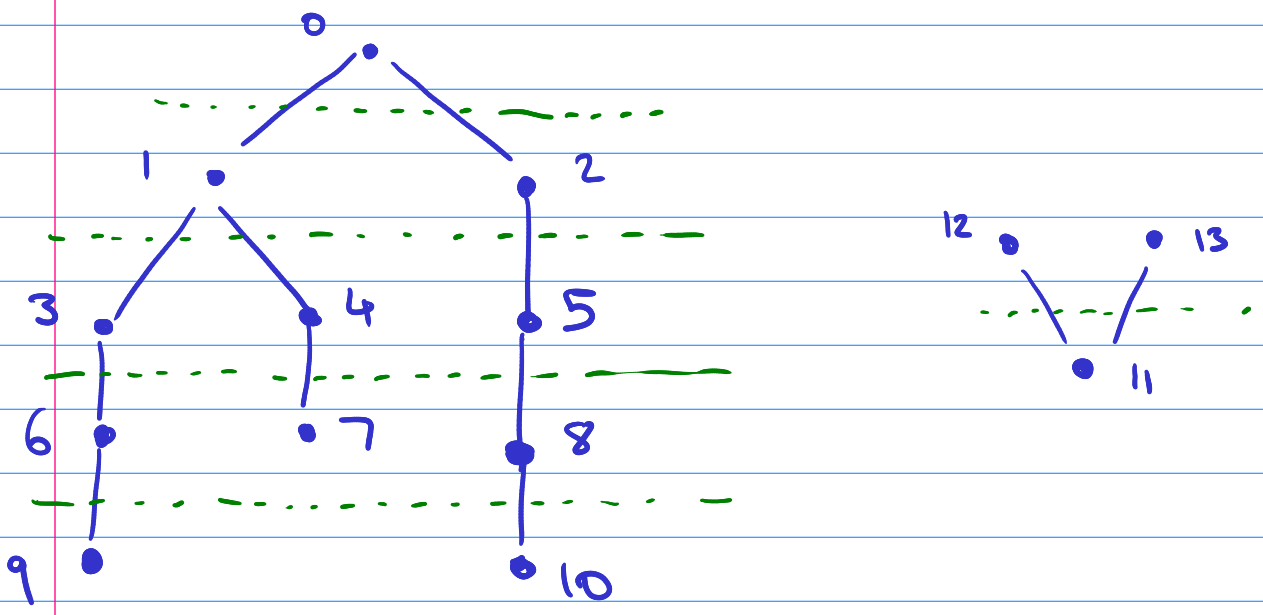
There are no unnecessary tuples in the output.

Programs will terminate (under some assumptions)

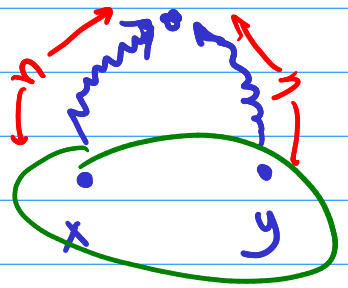
Example Find all people in the same generation of a family tree.

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

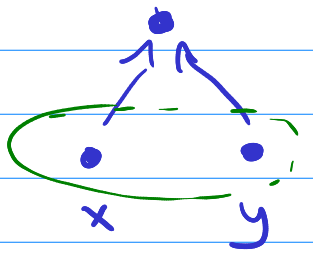
Samegen (x y) — "x & y are in the same generation"



R1: If x & y have a common ancestor,
& same # of levels to reach this ancestor,
then Samegen (x y).

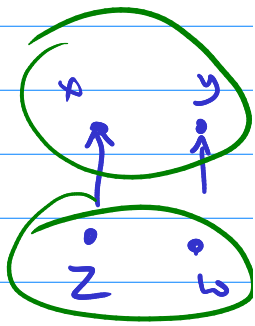


R2: If x & y have the same parent,
then Samegen (x, y)



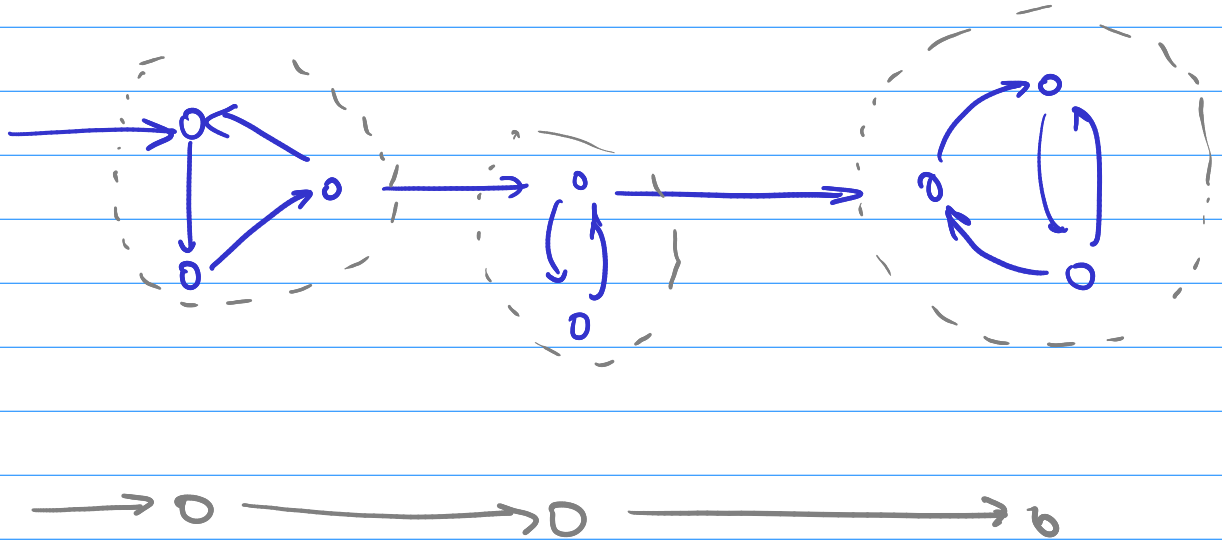
Samegen (x, y) :- Parent (z x), Parent (z, y)

R3: If x & y are in the same generation
& if Parent (x z) & Parent (y w)
then z & w are in the same gen.



Samegen (z w) :- Samegen (x y)
Parent (x z) Parent (y w).

Example Strongly Connected Components



Defn: x & y are in the same scc if there is a path from x to y & back.