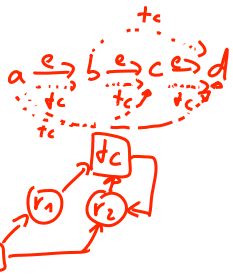
Announcements

- **Individual Assignment #1:**
  - Due Friday.
- **This week's schedule:**
  - **Lectures**
    - Mon, 3:10pm
    - **Wed 9am (discussion slot)**
    - Wed 3:10pm
  - **Discussion??**
    - **Fri 3:10pm (lecture slot)**

More on Datalog Evaluation




```
e(a,b). e(b,c). e(c,d).          % FACTS (EDB-relation: e/2)

tc(X,Y) :- e(X,Y).               % RULES (IDB-relation: tc/2)
tc(X,Y) :- e(X,Z), tc(Z,Y).
```

$2 = $1

Plan:
1. exec r_1 (completely)
2. Repeat exec r_2 until no new tc atoms

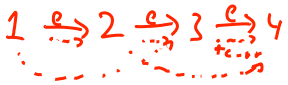
n	I_n
0	$\emptyset$
1	$\{\mathbf{e(a,b)}, \mathbf{e(b,c)}, \mathbf{e(c,d)}\}$ ← facts (EDB) – rule 0
2	$\{e(a,b), e(b,c), e(c,d)\} \cup \{\mathbf{tc(a,b)}, \mathbf{tc(b,c)}, \mathbf{tc(c,d)}\}$ ← r_1 copy e/2 into tc/2
3	$\{e(a,b), e(b,c), e(c,d)\} \cup \{tc(a,b), tc(b,c), tc(c,d), \mathbf{tc(a,c)}, \mathbf{tc(b,d)}\}$ r_2
4	$\{e(a,b), e(b,c), e(c,d)\} \cup \{tc(a,b), tc(b,c), tc(c,d), tc(a,c), tc(b,d), \mathbf{tc(a,d)}\}$
5	$\{\ldots\}$ = ↑ no change!

⇒ if the longest path in e/2 has length n, then $O(n)$ rounds are needed!

(Exercise: how about the following rule? `tc(X,Y)` ← `tc(X,Z), tc(Z,Y).`)

Three transitive closure variants

```
% EDB
e(1,2).
e(2,3).
e(3,4).

% IDB

% left-recursive (linear)
tc_lr(X,Y) :- e(X,Y).
tc_lr(X,Y) :- tc_lr(X,Z), e(Z,Y).

% right-recursive (linear)
tc_rr(X,Y) :- e(X,Y).
tc_rr(X,Y) :- e(X,Z), tc_rr(Z,Y).

% double-recursive (non-linear)
tc_dr(X,Y) :- e(X,Y).
tc_dr(X,Y) :- tc_dr(X,Z), tc_dr(Z,Y).
```

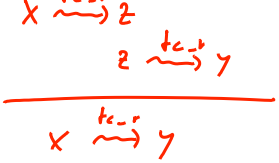


Round	tc_rr
1	{ (1,2), (2,3), (3,4) }
2	-11- ∪ { (1,3), (2,4) }
3	-11- ∪ { (1,4) }
4	no change! ⇒ fixpoint

r_1 : tc_dr(X,Y) ∈ e(X,Y).

r_2 : tc_dr(X,Y) ∈ tc_dr(X,Z), tc_dr(Z,Y).

Round	tc_dr	
1	{(1,2), (2,3), (3,4)}	// r_1
2	{(1,3), (2,4)} ∪ {…}	// r_2
3	{(1,4)} ∪ {…}	// r_2
4	no change	



Recursion vs Aggregation, Negation

- Negation before or after recursion: OK

(1) E(X,Y) :- A(X,Y), not B(X,Y).
(2) TC(X,Y) :- E(X,Y).
(3) TC(X,Y) ⇐ E(X,Z), TC(Z,Y).
(4) nTC(X,Y) :- node(X), node(Y), not TC(X,Y).

// $E := A \setminus B$
$TC := E^+$
node(x) ∈ E(x,_).
node(x) ∈ E(_,x).
(0)
↑ complement of TC (= E^+) ↷ one cannot reach y from X via $E^{(*)}$

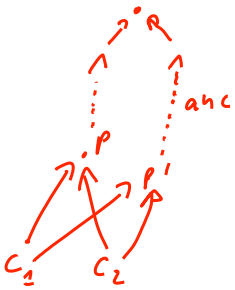
"Wrong schedule":
(1)
(4)
{(2),(3)} until no change

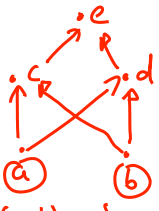
"Correct schedule":
(0)
(1)
(2)
Repeat (3) until no change
(4)

Lowest Common Ancestor

```
% anc(Node, Ancestor)
anc(C,P) :- parent(C,P).
anc(X,Y) :- anc(X,Z), anc(Z,Y).

% ca(NodeX,NodeY, CommonAncestor)
ca(X,Y,A) :-
        anc(X,A),
        anc(Y,A),
        X≠Y.

not_lca(X,Y,P) :-
        pc(P,C),
        ca(X,Y,P),
        ca(X,Y,C).

lca(X,Y,L) :-
        ca(X,Y,L),
        not not_lca(X,Y,L).
```

anc := parent+

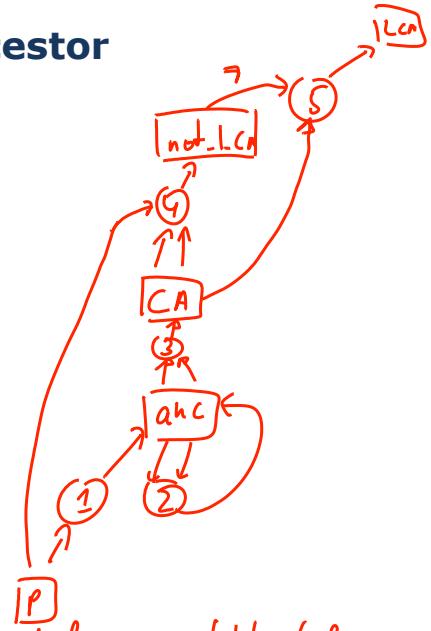




CA(a,b) = {c, d, e}
LCA(a,b) = {c, d}

Lowest Common Ancestor

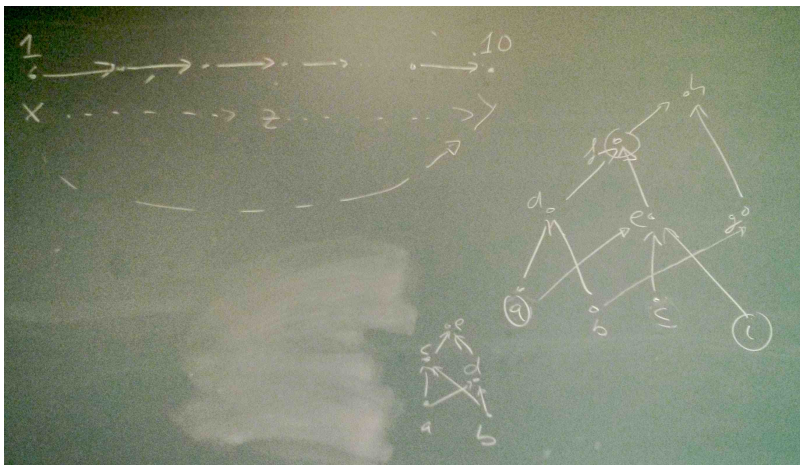
```
% anc(Node, Ancestor)
anc(C,P) :- parent(C,P).
anc(X,Y) :- anc(X,Z), anc(Z,Y).

% ca(NodeX,NodeY, CommonAncestor)
ca(X,Y,A) :-
        anc(X,A),
        anc(Y,A).

not_lca(X,Y,P) :-
        parent(P,C),
        ca(X,Y,P),
        ca(X,Y,C).

lca(X,Y,L) :-
        ca(X,Y,L),
        not not_lca(X,Y,L).
```



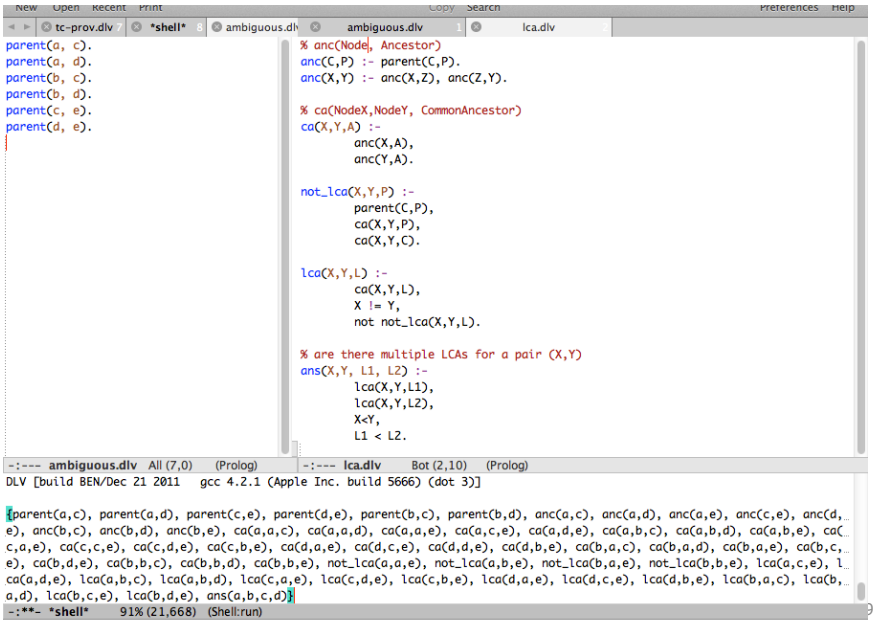
need to compute not_lca completely before applying this rule ⇒ stratification

Recursion Examples

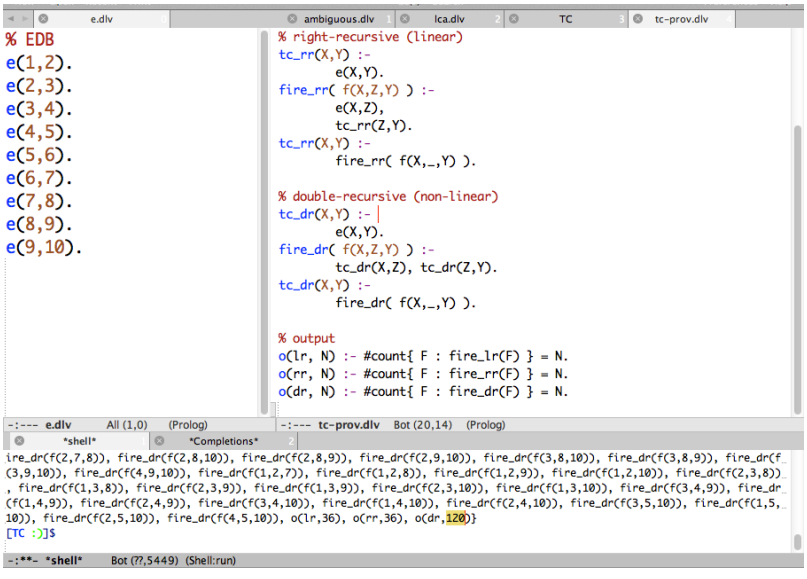
- TC variants
- LCA: on trees vs on DAGs



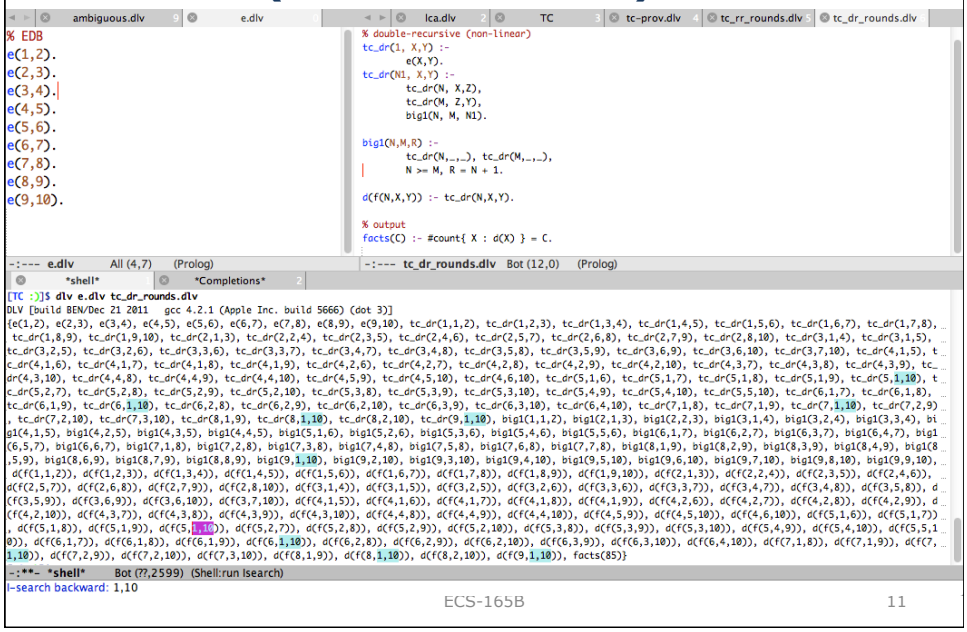
LCA (ambiguous case on DAGs)



Counting rule firings (right-recursive vs double-recursive)



Counting derived facts and "rounds" (double recursive)



Counting derived facts and "rounds" (right recursive)