Datalog and Answer Set Programming (part 5)

Riccardo Rosati

Knowledge Representation and Semantic Technologies
Master of Science in Engineering in Computer Science
Sapienza Università di Roma
a.a. 2020/2021

http://www.diag.uniroma1.it/rosati/krst/

Datalog with stratified negation

Non-recursive use of negation

As previously illustrated, the addition of negation to Datalog creates both semantic and computational issues.

The problematic examples presented above suggest that the use of **negation** creates problems when it is used in combination with **recursion**.

To overcome such problems, we define a subclass of programs with a non-recursive form of negation, called programs with **stratified negation**.

Precedence graph

Let P be a Datalog program with negation. The precedence graph of P is a graph G=(V,E,L) where V is the set of vertices, E is the set of edges, and L is a labeling of the edges in E, defined as follows:

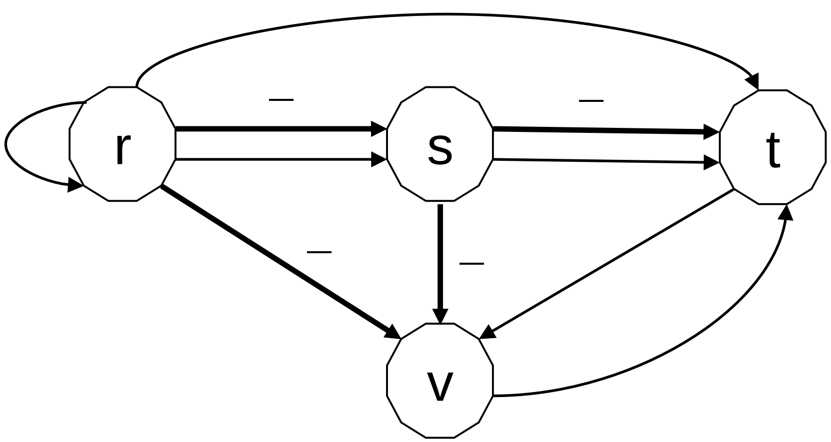
- There is one vertex v in V for every IDB predicate in P;
- There is an edge (s,t) in E (without label) if s,t are IDB predicates and P contains a rule R such that t appears in the head of R and s appears in a positive atom in the body of R;
- There is a **negated** edge, i.e. an edge with label "-", (s,t) in E if s,t are predicates and P contains a rule R such that t appears in the head of R and s appears in a negated atom in the body of R.

Precedence graph: Example

Example: let P be the following Datalog program with negation:

$r(X,Y) \text{ :– } p(X,Y),\ not\ p(Y,X).$
$r(X,Y) \text{ :– } p(X,Y),\ \ r(Y,Z).$
$s(X,Y) \text{ :– } r(Z,X),\ r(Z,Y),\ not\ r(X,Y).$
$t(X) \text{ :– } r(Y,X),\ s(X,X),\ not\ s(Y,Y).$
$t(X) \text{ :– } v(X,Y).$
$v(X,Y) \text{ :– } t(X),\ t(Y),\ not\ r(X,Y),\ not\ s(X,Y).$
$p(a,b).\quad p(b,c).$

Precedence graph G of P:



Programs with stratified negation

We say that a Datalog program with negation P is **stratified** if no cycle of the precedence graph of P contains a negated edge.

Example (contd.):

The precedence graph G has no cycle that contains a negated edge: the only cycles are: 1) the single edge (r,r) which is a positive edge; 2) the path (t,v), (v,t) which is only made of positive edges.

Consequently, P is a stratified program (or, equivalently, a program with stratified negation).

Operational semantics

For programs with stratified negation, the problems with the operational semantics (i.e. the iterated application of the immediate consequence operator) previously described do not arise.

Property: every stratified program P has a **unique** answer set, which coincides with the **unique** minimal model of P, and is denoted by MM(P).

It is possible to identify an algorithm that **deterministically** converges to the unique minimal model of the program with stratified negation.

Stratification of a program

A vertex v of G has a **negative dependence** if there exists a vertex v' in G such that there is a path from v' to v passing through a negated edge

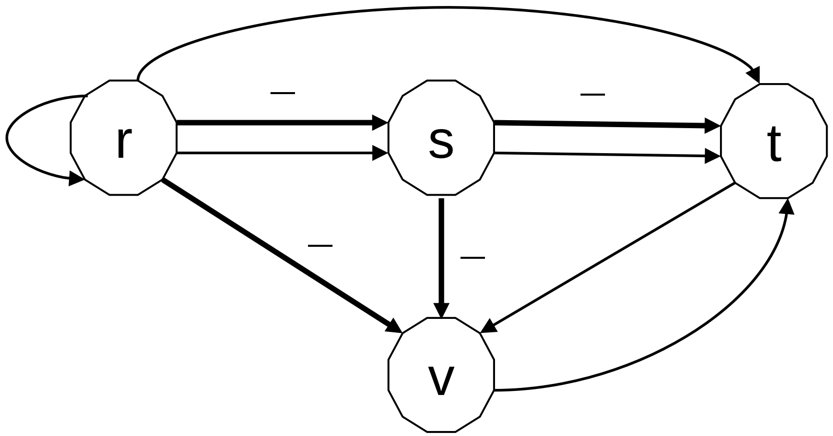
The **stratification** of a stratified program P is a sequence $S_1,\ldots, S_k$ of sets of IDB predicates of P (called **strata**) obtained as follows:

G = predecence graph of P;
i=0;
while G is not empty do begin
  i=i+1;
  S_i = the set of vertices of G that do not have negative dependencies;
  G = the precedence graph of P obtained considering (in addition to the initial EDB predicates) the predicates in $S_1,\ldots, S_i$ as EDB
end;
return $S_1,\ldots, S_i$;

Stratification: Example

Example (contd.):

The algorithm first considers the initial precedence graph G of P:

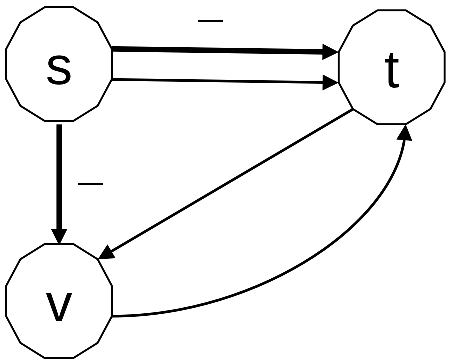


The only vertex that does not have negative dependencies is r, therefore

$S_1 = \{ r \}$

Stratification: Example

We now analyze the precedence graph obtained considering r an EDB predicate, i.e. we eliminate the vertex r and all its outcoming edges:

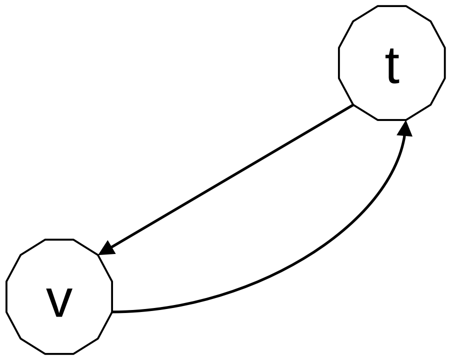


The only vertex that does not have negative dependencies is s, therefore

$S_2 = \{ s \}$

Stratification: Example

We now analyze the precedence graph obtained considering both r and s as EDB predicates, i.e. we eliminate from the previous graph the vertex s and all its outcoming edges:



Since there are no more negated edges, both t and v have no negative dependencies, therefore

$S_3 = \{ t, v \}$

And since, after deleting t and v, the graph is empty, the algorithm ends returning the stratification S_1, S_2, S_3.

Minimal model of a stratified program

Algorithm for computing the minimal model of a stratified program P:

Let $S_1, \ldots, S_k$ be the stratification of P;

MM_0 = EDB(P);

For i=1 to k do begin

  $P(S_i)$ = the program obtained from P by considering only the rules having a predicate from S_i in their head;

  MM_i = minimal model of $P(S_i) \cup MM_{i-1}$

end;

return MM_i;

Example

Example (contd.):

If we execute the previous algorithm on program P, we obtain:

MM_0 = EDB(P) = $\{\, p(a,b), p(b,c) \,\}$

$P(S_1)$ =

$r(X,Y)$:– $p(X,Y),\ \ not\ p(Y,X)$.

$r(X,Y)$:– $p(X,Y),\ \ r(Y,Z)$.

$MM_1 = MM_0 \cup \{\, r(a,b), r(b,c), r(a,c) \,\}$

$P(S_2)$ =

$s(X,Y)$:– $r(Z,X),\ \ r(Z,Y),\ \ not\ r(X,Y)$.

$MM_2 = MM_1 \cup \{\, s(b,b), s(c,c) \,\}$

Example

P(S_3) =

$t(X)$:– $r(Y,X),\ s(X,X),\ not\ s(Y,Y)$.

$t(X)$:– $v(X,Y)$.

$v(X,Y)$:– $t(X),\ t(Y),\ not\ r(X,Y),\ not\ s(X,Y)$.

$MM_3 = MM_2 \cup \{\ t(b), t(c), v(c,b)\ \}$

Therefore, the algorithm returns the model MM_3, i.e.:

$\{\ p(a,b), p(b,c), r(a,b), r(b,c), r(a,c), s(b,b), s(c,c), t(b), t(c), v(c,b)\ \}$