Datalog and Answer Set Programming (part 2)

Riccardo Rosati

Knowledge Representation and Semantic Technologies
Maser of Science in Engineering in Computer Science
Sapienza Università di Roma

a.a. 2020/2021

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

Positive Datalog: reasoning

Reasoning in positive Datalog

Basic reasoning task: construction of $\mathrm{MM}(P)$

Derived reasoning task: ground atom entailment

- Given a Datalog program P and a ground atom α, we say that P **entails** α if $\alpha \in \mathrm{MM}(P)$

Immediate consequence operator

Given a ground positive Datalog program P, the **immediate consequence operator for $\boldsymbol{P}$**, denoted as T_P, is the function over the domain of interpretations for P defined as follows:

$$T_P(I) = \{\, \alpha \mid \text{there exists a rule } \alpha \text{ :– } \beta_1, \ldots, \beta_n \text{ in } P \text{ such that } \{\beta_1, \ldots, \beta_n\} \subseteq I \,\}$$

The **least fixed point** of the function T_P is the minimal interpretation I such that $T_P(I) = I$.

Immediate consequence operator

Example: let P be the following ground program:

$r(a) \coloneq p(a).$
$r(b) \coloneq q(b).$
$p(a).$

Then:

$T_P(\{\ \}) = \{\, p(a)\, \}$

$T_P(\{\, p(a)\, \}) = \{\, p(a), r(a)\, \}$

$T_P(\{\, p(a), r(a)\, \}) = \{\, p(a), r(a)\, \}$ (least fixed point)

$T_P(\{\, q(b)\}) = \{\, p(a), r(b)\, \}$

$T_P(\{\, p(a), r(b)\}) = \{\, p(a), r(a)\, \}$

$T_P(\{\, p(a), q(b), r(b)\}) = \{\, p(a), r(a), r(b)\, \}$

$T_P(\{\, p(a), r(a), r(b)\}) = \{\, p(a), r(a), r(b)\, \}$ (fixed point)

$T_P(\{\, p(a), q(b), r(a), r(b)\}) = \{\, p(a), r(a), r(b)\, \}$

Operational semantics

Property: For every ground positive Datalog program P, the immediate consequence operator T_P has a unique least fixed point that coincides with MM(P).

The above property provides an **operational semantics** for ground positive Datalog program.

In fact, from the previous property we can immediately derive the following algorithm for the computation of the miminal model of a ground positive program P.

Naive evaluation

```
Algorithm naive-evaluation
Input: ground positive Datalog program P
Output: MM(P)
begin
  let I'={};
  repeat
    let I=I';
    compute I'=T_P(I)
  until I'==I;
  return I
end
```

This algorithm executes at most k+1 iterations of the repeat-until loop, where k is the number of rules of P

Reasoning over non-ground programs

The naive evaluation algorithm could be used also for non-ground programs:

1. First, compute the ground program ground(P);
2. Then, execute the naive evaluation algorithm on ground(P).

The interpretation returned by the algorithm is the minimal model of ground(P) and therefore the minimal model of P.

We now present the **semi-naive evaluation** algorithm, which optimizes the naive evaluation.

Delta predicates

Idea of the optimization: reformulate the program P in a way such that the immediate consequence operator T_P can derive a ground atom only if its derivation depends on at least one ground atom derived in the previous application of T_P in the algorithm.

This minimizes redundant derivations, i.e., repeated derivations of the same ground atoms in different applications of T_P.

This idea is realized by introducing **delta** versions of the IDB predicates of the program, and rewriting the program through the usage of such delta predicates.

Delta predicates

In the algorithm, the extension of a Δ-predicate Δp represents the ground atoms relative to predicate p derived by the **previous** application of the immediate consequence operator.

A second set of Δ'-predicates is used in rule heads: in this way, each $\Delta' p$ represents the ground atoms relative to predicate p derived by the **current** application of the immediate consequence operator.

Δ-transformation of a rule

In the following, we denote a rule r of a positive program P as follows:

$$\alpha :- \beta_1, \ldots, \beta_k, \gamma_1, \ldots, \gamma_h$$

where every β_i is an IDB-atom (i.e. an atom in which an IDB predicate of P occurs), and every γ_i is an EDB-atom

Given a rule r of the above form, Δr is the following set of k rules:

$$\Delta'\alpha :- \Delta\beta_1, \ \beta_2, \ldots, \beta_k, \gamma_1, \ldots, \gamma_h$$

$$\Delta'\alpha :- \beta_1, \ \Delta\beta_2, \ldots, \beta_k, \gamma_1, \ldots, \gamma_h$$

…

$$\Delta'\alpha :- \beta_1, \ \beta_2, \ldots, \Delta\beta_k, \gamma_1, \ldots, \gamma_h$$

where:

- if $\beta_i = r(t_1, \ldots, t_n)$, then $\Delta\beta_i$ denotes the atom $\Delta r(t_1, \ldots, t_n)$
- if $\alpha = r(t_1, \ldots, t_n)$, then $\Delta'\alpha$ denotes the atom $\Delta' r(t_1, \ldots, t_n)$

Δ-transformation of a rule

Therefore, if the body of r contains k atoms with IDB predicates, Δr contains k rules (for every such atom there is a rule where the delta predicate is used in such an atom instead of the standard predicate)

Δ-transformation of a rule

Example: let IDB = $\{r/1, s/2\}$, EDB = $\{p/2\}$, and let P be as follows:

$r(X)$:– $s(X,Y), p(X,Y)$. [r1]

$s(X,Y)$:– $p(X,Y), p(Y,Z)$. [r2]

$s(X,Y)$:– $s(X,Z), r(Y), p(Y,X)$. [r3]

$p(a)$. [r4]

Then:

Δr1 = { $\Delta' r(X)$:– $\Delta s(X,Y), p(X,Y)$. }

Δr2 = { }

Δr3 = { $\Delta' s(X,Y)$:– $\Delta s(X,Z), r(Y), p(Y,X)$.

$\Delta' s(X,Y)$:– $s(X,Z), \Delta r(Y), p(Y,X)$. }

Δr4 = { }

Δ-transformation of a program

Given a positive Datalog program P, the Δ-transformation of P, denoted as ΔP, is the program obtained as the union of all the sets Δr of every rule r in P.

Example (contd.):

$$\Delta P = \Delta r1 \cup \Delta r2 \cup \Delta r3 \cup \Delta r4 =$$

$$= \{\ \Delta' r(X) :\!-\ \Delta s(X,Y), p(X,Y).$$

$$\Delta' s(X,Y) :\!-\ \Delta s(X,Z), r(Y), p(Y,X).$$

$$\Delta' s(X,Y) :\!-\ s(X,Z), \Delta r(Y), p(Y,X).\ \}$$

The algorithm then computes the immediate consequence operator $T_{\Delta P}$ of ΔP, treating both Δ-predicates and Δ'-predicates like all other standard predicates.

Semi-naive evaluation

```
Algorithm semi-naive-evaluation
Input: positive Datalog program P
Output: MM(P)
begin
  let I=EDB(P); //(EDB(P) is the set of facts in P)
  compute I'=T_P(I);
  if I==I' then return I;
  ΔI = { Δ'α | α ∈ I' – I };
  repeat
    let I = I ∪ { α | Δ'α ∈ Δ'I };
    let ΔI = { Δα | Δ'α ∈ Δ'I };
    let Δ'I = T_ΔP(I ∪ ΔI)
  until Δ'I=={};
  return I
end
```

Example of execution of the algorithm

Let P be the following positive Datalog program:

$r(X,Y) :- p(X,Y).$ [r1]
$r(X,Z) :- p(X,Y), r(Y,Z).$ [r2]
$s(X,Y) :- r(Y,X).$ [r3]
$p(a,b).$
$p(b,c).$
$p(c,d).$

- The initial value of I is the set of facts of P, i.e. $\{ p(a,b), p(b,c), p(c,d) \}$
- Then, $I' = T_P(I) = I \cup \{ r(a,b), r(b,c), r(c,d) \}$ (applying r1)
- So, $\Delta I = \{\Delta r(a,b),\ \Delta r(b,c), \Delta r(c,d) \}$

Example (contd.)

- Then, the algorithm executes the repeat-until loop for the first time, obtaining:

$$\mathtt{I} = \mathtt{I} \cup \{ r(a,b), r(b,c), r(c,d) \}$$

$$\Delta\mathtt{I} = \{ \Delta r(a,b), \Delta r(b,c), \Delta r(c,d) \}$$

$$\Delta'\mathtt{I} = \mathrm{T}_{\Delta P}(\mathtt{I} \cup \Delta\mathtt{I}) =$$

$$= \{ \Delta' r(a,c), \Delta' r(b,d), \Delta' s(b,a), \Delta' s(c,b), \Delta' s(d,c) \}$$

(applying r2 and r3)

- Then, the algorithm executes the repeat-until loop for the second time, obtaining:

$$\mathtt{I} = \mathtt{I} \cup \{ r(a,c), r(b,d), s(b,a), s(c,b), s(d,c) \}$$

$$\Delta\mathtt{I} = \{ \Delta r(a,c), \Delta r(b,d), \Delta s(b,a), \Delta s(c,b), \Delta s(d,c) \}$$

$$\Delta'\mathtt{I} = \mathrm{T}_{\Delta P}(\mathtt{I} \cup \Delta\mathtt{I}) =$$

$$= \{ \Delta' r(a,d),\ \Delta' s(c,a), \Delta' s(d,b) \}$$ (applying r2 and r3)

Example (contd.)

- Then, the algorithm executes the repeat-until loop for the third time, obtaining:

$$\mathtt{I} = \mathtt{I} \cup \{ r(a,d), s(c,a), s(d,b) \}$$
$$\Delta\mathtt{I} = \{ \Delta r(a,d), \Delta s(c,a), \Delta s(d,b) \}$$
$$\Delta'\mathtt{I} = \mathrm{T}_{\Delta\mathrm{P}}(\mathtt{I} \cup \Delta\mathtt{I}) = \{ \Delta' s(d,a) \} \text{ (applying r3)}$$

- Then, the algorithm executes the repeat-until loop for the fourth time, obtaining:

$$\mathtt{I} = \mathtt{I} \cup \{ s(d,a) \}$$
$$\Delta\mathtt{I} = \{ \Delta s(d,a) \}$$
$$\Delta'\mathtt{I} = \mathrm{T}_{\Delta\mathrm{P}}(\mathtt{I} \cup \Delta\mathtt{I}) = \{ \}$$

Example (contd.)

- Since Δ'`I` is empty, the algorithm exits the repeat-until loop and terminates returning the interpretation I, that is, the set

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

Such an interpretation I is the minimal model of P.