Datalog and Answer Set Programming

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/

Outline

- Positive Datalog
- Positive Datalog with constraints
- Datalog with stratified negation
- Datalog with (non-stratified) negation
- Answer Set Programs (Datalog with negation and disjunction)

Positive Datalog: syntax

Datalog: alphabets

We start from the following alphabets (sets of symbols):

- Alphabet of predicate symbols **Pred**
  - every predicate symbol is associated with an **arity** (non-negative integer) (e.g. $p/2, r/1, s/0, \ldots$)
- Alphabet of constant symbols **Const**
  - Syntactic convention: we will use symbols starting with lowercase letters (e.g. $a, b, c, \ldots$)
- Alphabet of variable symbols **Var**
  - Syntactic convention: we will use symbols starting with uppercase letters (e.g. $X, Y, Z, \ldots$)

The three alphabets are pairwise disjoint (every symbol belongs at most to one alphabet)

We also define the (derived) set of terms **Term** = **Const** ∪ **Var**

Datalog: atoms

atom = expression of the form

$$p(t_1, \dots, t_n)$$

where:

- n is a non-negative integer
- p is a predicate symbol of arity n
- every t_i is a term

Datalog: positive rules

Positive rule = expression of the form

$$\alpha \mathrel{:-} \beta_1, \ldots, \beta_n$$

where:

- n is a non-negative integer
- α is an atom
- every β_i is an atom
- (**safeness** condition) every variable symbol occurring in α must appear in $\beta_1, \ldots, \beta_n$

α is called the rule **head**

$\beta_1, \ldots, \beta_n$ is called the rule **body**

Examples of positive rules

Pred = $\{p/1, q/0, r/2, s/2, t/1, u/3\}$

Const = $\{a, b, c\}$

Var = $\{X, Y, Z, W\}$

Correct positive rules:

- $p(X)$:– $r(X, Y), s(Y, Z)$.
- q :– $r(X, Y), s(Y, Z)$.
- $r(X, Z)$:– $r(X, Y), s(Y, Z)$.
- $u(X, Y, Z)$:– $r(X, Y), s(Y, Z)$.
- $u(X, a, b)$:– $r(X, Y)$.
- $u(c, a, b)$:– .

Examples of positive rules

Pred = $\{p/1, q/0, r/2, s/2, t/1, u/3\}$

Const = $\{a, b, c\}$

Var = $\{X, Y, Z, W\}$

Incorrect positive rules:

- $p(W)$:– $r(X,Y), s(Y,Z)$.
  - variable W is not safe
- q :– $r(X,Y), s(Y,Z)$
  - missing "." at end of rule
- $t(X,Y,r)$:– $r(X,Y), s(Y,Z)$.
  - uses a predicate in argument position
- $Y(X,a,b)$:– $r(X,Y)$.
  - uses a variable in predicate position

Facts

A **fact** is a positve rule with an empty body

E.g.:

- $u(c, a, b)$:– .
- $p(a)$:– .
- q :– .

When representing facts we omit the "if" symbol, i.e.:

- $u(c, a, b)$.
- $p(a)$.
- q.

Notice that variable symbols cannot appear in a fact (due to the safeness condition)

Positive Datalog

Positive Datalog program = set of Positive datalog rules

Example:

$$r(X,Z) \coloneq p(X,Y), p(Y,Z).$$
$$s(X,Y) \coloneq r(Y,X).$$
$$t(X,Y) \coloneq q(X), r(X,Y), s(Y,Z).$$
$$p(a,b).$$
$$p(b,c).$$
$$q(a).$$

EDB and IDB predicates

Predicates in a Datalog program are actually partitioned into **EDB** (Extensional Database) predicates and **IDB** (Intensional Database) predicates:

- EDB predicates can only be used in facts and in the body of rules
- IDB predicates cannot be used in facts

E.g.: IDB= $\{p/2, q/1\}$, EDB= $\{r/2, s/2\}$

$p(a, b)$.
$p(b, c)$.
$q(b)$.
$r(X, Y) :- p(X, Y)$.
$r(X, Z) :- p(X, Y), r(Y, Z)$.
$s(X, Z) :- r(X, Y), q(Y)$.

Recursive rules

A Datalog rule is **recursive** if the predicate occurring in its head also occurs in its body.

- EDB predicates can only be used in facts and in the body of rules
- IDB predicates cannot be used in facts

E.g., the following are recursive rules:

$$r(X,Z) :- p(X,Y), r(Y,Z).$$
$$s(X,Y) :- s(Y,X).$$

Ground rules and programs

A Datalog rule is **ground** if no variable occurs in it

Examples of ground rules:

$r(a,b) :\!- p(b,c).$

$s(c,d) :\!- t, r(a,a).$

$t :\!- r(a,c), s(b,c), q(a).$

$p(a,b).$

$q(b).$

(of course, every fact is a ground rule)

A Datalog program is **ground** if all its rules are ground

Positive Datalog: semantics

Herbrand Base

Given a Datalog program P, the **Herbrand Universe** of P (denoted by HU(P)) is the set of constant symbols occurring in P

Given a ground Datalog program P, the **Herbrand Base** of P (denoted by HB(P)) is the set of ground atoms occurring in P

E.g., let P_1 be the following ground program:

$r(a, b)$:– $p(b, a)$.
$s(c, d)$:– $t, r(a, a)$.
t :– $r(a, c), s(b, c), q(a)$.
$p(b, a)$.
$q(b)$.

Then, $HB(P_1) = \{r(a,b), p(b,a), s(c,d), t, r(a,a), r(a,c), s(b,c), q(a), q(b)\}$

Interpretations of ground programs

Given a ground Datalog program P, an **interpretation** for P is a subset of HB(P).

E.g., the following are possible interpretations for the ground program P_1 of the previous example:

$I_1 = \{r(a,a), p(b,a)\}$

$I_2 = \{q(b)\}$

$I_3 = \{r(a,b), p(b,a), s(c,d), t, r(a,a), r(a,c), s(b,c), q(a), q(b)\}$

$I_4 = \{\ \}$

$I_5 = \{p(b,a), q(b), r(a,b)\}$.

Models for ground programs

A ground positive rule r is **satisfied** in an interpretation I

if either some atom in the body of r does not belong to I or the atom in the head of r belongs to I

Examples:

- the rule $r(a,b)$:– $p(b,a)$. is satisfied in the interpretations I_2 , I_3 , I_4 , I_5 , but not in I_1.
- the rule (fact) $q(b)$. is satisfied in the interpretations I_2 , I_3 , I_5 , but not in I_1, I_4

An interpretation I is a **model** for a ground Datalog program P, if all the rules in P are satisfied in I

Models for ground programs

Example: let P be the program

$r(a) \text{ :- } p(a).$

$r(b) \text{ :- } q(b).$

$p(a).$

then, the following interpretations are models for P:

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

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

while the following are NOT models for P:

$\{\ \}$

$\{p(a)\}$

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

Herbrand Base of non-ground programs

Herbrand Base of a **non-ground** program P = HB(P) = set of all the ground atoms that can be built with the predicates and the constants occurring in P

E.g., if P is the program

$r(X,Y) :- p(X,Y).$
$s(X,Y) :- r(X,Z), s(Z,Y).$
$p(b,a).$
$q(b).$

Then HB(P) =

$$\left\{ \begin{array}{c} p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b), r(a,a), r(a,b), r(b,a), \\ r(b,b), s(a,a), s(a,b), s(b,a), s(b,b) \end{array} \right\}$$

Interpretations and models

Given a non-ground Datalog program P, an **interpretation** for P is a subset of HB(P).

The **grounding** of a Datalog **rule** r with respect to a set of constants C, denoted as $ground(r, C)$, is the set of all ground rules that can be obtained from r by replacing, for every variable x occurring in r, every occurrence of x with a constant from C.

The **grounding** of a non-ground Datalog **program** P, $ground(P)$, is the ground Datalog program obtained by the union of all the sets $ground\big(r, HU(P)\big)$ such that $r \in P$.

An interpretation I of a non-ground Datalog program P is a **model** for P if I is a model for $ground(P)$

Minimal models

An interpretation I is a **minimal model** for a (non-ground) Datalog program P, if I is a model for P and there exists no model I’ for P such that I’ is a strict subset of I.

Property: every positive Datalog program P has exactly one minimal model (we denote such a model by MM(P))

Minimal model: example

Example: let P be the program

$r(a)$:– $p(a)$.

$r(b)$:– $q(b)$.

$p(a)$.

the following interpretations are the models for P:

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

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

So, the minimal model for P is:

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

Reasoning in positive Datalog

Basic reasoning task: construction of 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 $\alpha \in$ MM(P)