

# Semantics of Datalog: Circumscription, Stable Models, Aggregates

Vladimir Lifschitz  
University of Texas at Austin

We are interested in answer set programming systems, such as SMODELS and DLV, viewed as mechanisms for characterizing “output” predicates in terms of “input” predicates.

Example 1: Transitive closure.

$$q(X,Y) \text{ :- } p(X,Y) .$$
$$q(X,Z) \text{ :- } q(X,Y), q(Y,Z) .$$

Input:

$$p(a,b) . p(b,c) .$$

Output:

$$\{q(a,b), q(a,c), q(b,c)\}$$

Example 2: Partitioning a set.

$q(X) ; r(X) :- p(X).$

Input:

$p(a). p(b). p(c).$

Output:

$\{r(a), r(b), r(c)\}$

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

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

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

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

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

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

$\{q(a), q(b), q(c)\}$

Example 3: Partitioning subject to a constraint.

```
q(X) ; r(X) :- p(X).  
:- q(a).
```

Input:

```
p(a). p(b). p(c).
```

Output:

```
{r(a), r(b), r(c)}  
{r(a), q(b), r(c)}  
{r(a), r(b), q(c)}  
{r(a), q(b), q(c)}
```

Example 4: Partitioning subject to a numeric constraint.

```
q(X) ; r(X) :- p(X).  
:- #sum{X:q(X)}>9.
```

Input:

```
p(4). p(5). p(6).
```

Output:

```
{r(4), r(5), q(6)}  
{r(4), q(5), r(6)}  
{q(4), q(5), r(6)}  
{q(4), r(5), r(6)}  
{r(4), r(5), r(6)}
```

Example 5: Terminal vertices.

$r(X) \text{ :- } q(X, Y).$

$s(X) \text{ :- } p(X), \text{ not } r(X).$

Input:

$p(a). p(b). q(a, b).$

Output:

$\{r(a), s(b)\}$

Example 6: Vertex degrees.

$r_0(X) \text{ :- } p(X), \#count\{Y:q(X,Y)\}=0.$

$r_1(X) \text{ :- } p(X), \#count\{Y:q(X,Y)\}=1.$

$r_2(X) \text{ :- } p(X), \#count\{Y:q(X,Y)\}=2.$

Input:

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

Output:

$\{r_0(b), r_0(c), r_2(a)\}$

Example 7: Choice.

$$\{q(X)\} \text{ :- } p(X).$$

Input:

$$p(a). \quad p(b). \quad p(c).$$

Output:

$$\{ \}$$
$$\{q(c)\}$$
$$\{q(b)\}$$
$$\{q(b), q(c)\}$$
$$\{q(a)\}$$
$$\{q(a), q(c)\}$$
$$\{q(a), q(b)\}$$
$$\{q(a), q(b), q(c)\}$$

## PDD: Positive Disjunctive Datalog

A *PDD program* is a conjunction of formulas of the form  $\tilde{\forall}(Body \rightarrow Head)$ , where

- *Body* is a first-order formula built using conjunction, disjunction, and quantifiers,
- *Head* is a disjunction of atoms.

$$\begin{array}{ll}
 q(X,Y) \text{ :- } p(X,Y) . & \forall xy(p(x,y) \rightarrow q(x,y)) \wedge \\
 q(X,Z) \text{ :- } q(X,Y), q(Y,Z) . & \forall xyz(q(x,y) \wedge q(y,z) \rightarrow q(x,z)) \\
 \\ \\
 q(X) \text{ ; } r(X) \text{ :- } p(X) . & \forall x(p(x) \rightarrow q(x) \vee r(x)) \wedge \\
 \text{ :- } q(a) . & (q(a) \rightarrow \perp)
 \end{array}$$

How are the intended models of a PDD program different from other models?

In a model of

$$\forall xy(p(x, y) \rightarrow q(x, y)) \wedge \forall xyz(q(x, y) \wedge q(y, z) \rightarrow q(x, z)),$$

$q$  is a superset of  $p$ , and it is a transitive relation.

In the intended models of this formula,  $q$  is the transitive closure of  $p$ .

*Semantic Idea No. 1:*

In intended models, the intensional predicates are minimal w.r.t. set inclusion.

Circumscription:

$p \leq q$  stands for  $\forall \mathbf{x}(p(\mathbf{x}) \rightarrow q(\mathbf{x}))$ ;

$(p_1, \dots, p_n) \leq (q_1, \dots, q_n)$  stands for  $(p_1 \leq q_1) \wedge \dots \wedge (p_n \leq q_n)$ ;

$\mathbf{p} < \mathbf{q}$  stands for  $(\mathbf{p} \leq \mathbf{q}) \wedge \neg(\mathbf{q} \leq \mathbf{p})$ .

$$CIRC_{\mathbf{p}}[F] = F \wedge \neg \exists \mathbf{u} ((\mathbf{u} < \mathbf{p}) \wedge F_{\mathbf{u}}^{\mathbf{p}}).$$

Example: The result of applying  $CIRC_{qr}$  to

$$\forall x(p(x) \rightarrow q(x) \vee r(x))$$

is equivalent to

$$\forall x(p(x) \rightarrow q(x) \vee r(x)) \wedge \neg \exists x(q(x) \wedge r(x)).$$

## DD: Disjunctive Datalog

A *DD program* is a conjunction of formulas of the form  $\tilde{\forall}(Body \rightarrow Head)$ , where

- *Body* is a first-order formula built using **negation**, conjunction, disjunction, and quantifiers,
- *Head* is a disjunction of **literals**.

$$\begin{array}{ll}
 r(X) \text{ :- } q(X, Y) . & \forall xy(q(x, y) \rightarrow r(x)) \wedge \\
 s(X) \text{ :- } p(X), \text{ not } r(X) . & \forall xy(p(x) \wedge \neg r(x) \rightarrow s(x))
 \end{array}$$

More examples:

$$r_0(X) \text{ :- } p(X), \text{ \#count}\{Y:q(X,Y)\}=0.$$

$$\forall x(p(x) \wedge \neg \exists y q(x,y) \rightarrow r_0(x))$$

$$r_1(X) \text{ :- } p(X), \text{ \#count}\{Y:q(X,Y)\}=1.$$

$$\forall x(p(x) \wedge \exists y q(x,y) \wedge \neg \exists_2 y q(x,y) \rightarrow r_1(x))$$

$$r_2(X) \text{ :- } p(X), \text{ \#count}\{Y:q(X,Y)\}=2.$$

$$\forall x(p(x) \wedge \exists_2 y q(x,y) \wedge \neg \exists_3 y q(x,y) \rightarrow r_2(x))$$

The DD program

$$\forall xy(q(x, y) \rightarrow r(x)) \wedge \forall xy(p(x) \wedge \neg r(x) \rightarrow s(x))$$

has fewer intended models than the PDD program

$$\forall xy(q(x, y) \rightarrow r(x)) \wedge \forall xy(p(x) \rightarrow r(x) \vee s(x)).$$

How to adapt the circumscription operator to programs with negation?

*Semantic Idea No. 2:*

Do not substitute variables for intensional predicates in the scope of negation.

For any DD program  $F$ ,  $F^\diamond(\mathbf{u})$  is obtained from  $F$  by replacing each part  $p_i(\mathbf{t})$  that is not in the scope of negation with  $u_i(\mathbf{t})$ .

$$DD_{\mathbf{p}}[F] = F \wedge \neg \exists \mathbf{u} ((\mathbf{u} < \mathbf{p}) \wedge F^\diamond(\mathbf{u})).$$

Example: the result of applying  $DD_{rs}$  to

$$\forall xy(q(x, y) \rightarrow r(x)) \wedge \forall xy(p(x) \wedge \neg r(x) \rightarrow s(x))$$

is  $\forall xy(q(x, y) \rightarrow r(x)) \wedge \forall xy(p(x) \wedge \neg r(x) \rightarrow s(x))$

$$\wedge \neg \exists uv((u, v) < (r, s))$$

$$\wedge \forall xy(q(x, y) \rightarrow u(x)) \wedge \forall x(p(x) \wedge \neg r(x) \rightarrow v(x)).$$

It is equivalent to

$$\forall x(r(x) \leftrightarrow \exists y q(x, y)) \wedge \forall x(s(x) \leftrightarrow (p(x) \wedge \neg \exists y q(x, y))).$$

Theorem: The meaning of a DD program is preserved by intuitionistically equivalent transformations.

Another example: the result of applying  $DD_{r_2}$  to

$$\forall x(p(x) \wedge \exists_2 y q(x, y) \wedge \neg \exists_3 y q(x, y) \rightarrow r_2(x))$$

is equivalent to

$$\forall x(r_2(x) \leftrightarrow p(x) \wedge \exists_2 y q(x, y) \wedge \neg \exists_3 y q(x, y)).$$

How to define the semantics of choice?

*Semantic Idea No. 3:*

Treat choice as shorthand for the law of the excluded middle.

$$\{q(X)\} \text{ :- } p(X). \quad \forall x(p(x) \rightarrow q(x) \vee \neg q(x))$$

The result of applying  $DD_q$  to this formula is equivalent to

$$\forall x(q(x) \rightarrow p(x)).$$

The definition of  $DD_p$  generalizes the stable model semantics.

*Traditional* DD program:

- in each rule  $Body \rightarrow Head$ ,  $Body$  is a conjunction of literals, and  $Head$  is an atom;
- all predicate symbols are intensional.

Theorem: For any traditional DD program  $F$ , the Herbrand interpretations satisfying  $DD_p[F]$  are identical to the stable models of  $F$ .

We would like to extend the semantics of Disjunctive Datalog to aggregates other than #count, such as #sum.

### Digression: Stable Model Operator

$$SM_{\mathbf{p}}[F] = F \wedge \neg \exists \mathbf{u}((\mathbf{u} < \mathbf{p}) \wedge F^*(\mathbf{u}))$$

- $p_i(\mathbf{t})^* = u_i(\mathbf{t})$
- $F^* = F$  for any atomic  $F$  that does not contain members of  $\mathbf{p}$
- $(F \wedge G)^* = F^* \wedge G^*$ ;  $(F \vee G)^* = F^* \vee G^*$
- $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$
- $(\forall x F)^* = \forall x F^*$ ;  $(\exists x F)^* = \exists x F^*$

The circumscription operator describes minimal models.

The stable model operator describes equilibrium models.

*SM* vs. *DD*:

Since  $\neg F$  stands for  $F \rightarrow \perp$ ,

$$(\neg F)^* = \neg F^* \wedge \neg F.$$

$DD_{pq}[\neg p \rightarrow q]$  is

$$(\neg p \rightarrow q) \wedge \neg \exists uv((u, v) < (p, q) \wedge (\neg p \rightarrow v)).$$

$SM_{pq}[\neg p \rightarrow q]$  is

$$(\neg p \rightarrow q) \wedge \neg \exists uv((u, v) < (p, q) \wedge (\neg u \wedge \neg p \rightarrow v) \wedge (\neg p \rightarrow q)).$$

Theorem: For any DD program  $F$ ,  $SM_{\mathbf{p}}[F]$  is equivalent to  $DD_{\mathbf{p}}[F]$ .

Uniform treatment of propositional connectives in the definition of  $F^*$ :  
the clauses

$$(F \wedge G)^* = F^* \wedge G^*$$

$$(F \vee G)^* = F^* \vee G^*$$

$$(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$$

can be equivalently replaced with

$$(F \odot G)^* = (F^* \odot G^*) \wedge (F \odot G) \quad (\odot \in \{\wedge, \vee, \rightarrow\}).$$

How to define the semantics of aggregates?

*Semantic Idea No. 4:*

In the definition of  $F^*$ , treat aggregates in the same way as propositional connectives.

$$(\#\text{sum}(x : F) > t)^* = (\#\text{sum}(x : F^*) > t) \wedge (\#\text{sum}(x : F) > t).$$

## Conclusion

Semantics of many Datalog constructs can be defined by modifying the definition of circumscription.

## Brief History

Program completion: Clark, 1978.

Circumscription: McCarthy, 1980.

Relating circumscription to stable models: Lin, 1991.

Equilibrium logic, propositional case: Pearce, 1997.

Operator SM, propositional case: Pearce, Tompits, and Woltran, 2001.

Equilibrium logic, first-order case: Pearce and Valverde, 2004.

Choice as excluded middle: Ferraris and Lifschitz, 2005.

Operator SM, first-order case: Ferraris, Lee, and Lifschitz, 2007.

Operator SM, language with aggregates: Lee and Meng, 2009.