Data Integration: Negation

Jan Chomicki

University at Buffalo

January 22, 2008

Open vs. Closed World Assumption

Closed World Assumption (CWA)

What is not implied by a logic program is false.

Open World Assumption (OWA)

What is not implied by a logic program is unknown.

Scope

- traditional database applications: CWA
- information integration: OWA or CWA

Can negation be allowed inside Datalog rules?

Datalog^{not}

Syntax

Rules with negated goals in the body:

$$A_0 :- A_1, \ldots, A_k, not\ B_1, \ldots, not\ B_m.$$

Example

```
forebear(X,Y):-anc(X,Y), not parent(X,Y).
```

Generalizing T_P

$$T_P(I) = \{A \mid \exists r \in ground(P).\ r = A :- A_1, \ldots, A_n, not\ B_1, \ldots, not\ B_m \\ \wedge A_1 \in I \wedge \cdots A_n \in I \wedge B_1 \notin I \wedge \cdots B_m \notin I\}$$

Datalog^{not}: semantics

Semantics

- minimal (Herbrand) models:
  - one or more
  - the right one?
- minimal fixpoints of T_P:
  - none, one, or more than one
  - the right one?
- bottom-up evaluation

Solutions

- restrict programs syntactically: *stratified,...*
- consider multiple logical meanings: *stable models,...*

Stratification [ABW88]

Dependency graph $pdg(P)$

- vertices: predicates of a Datalog^{not} program P
- edges:
  - a positive edge (p, q) if there is a clause in P in which q appears in a positive goal in the body and p appears in the head
  - a negative edge (p, q) if there is a clause in P in which q appears in a negative goal in the body and p appears in the head

Stratified P

No cycle in $pdg(P)$ contains a negative edge.

Stratification

Mapping s from the set of predicates in P to nonnegative integers such that:

1. if a positive edge (p, q) is in $pdg(P)$, then $s(p) \geq s(q)$
2. if a negative edge (p, q) is in $pdg(P)$, then $s(p) > s(q)$

There is a polynomial-time algorithm to determine whether a program is stratified, and if it is, to find a stratification for it.

Stratified Datalog^{not}: query evaluation

Bottom-up evaluation

1. compute a stratification of a program P
2. partition P into $P_1, \ldots, P_n$ such that
   - each P_i consisting of all and only rules whose head belongs to a single stratum
   - P_1 is the lowest stratum
3. evaluate bottom-up $P_1, \ldots, P_n$ (in that order).

Result

- does not depend on the stratification
- can be semantically characterized in various ways: minimal, perfect...
- is used to compute query results (like M_P)

Expressiveness

Query equivalence

Two queries are equivalent if their semantics defines the same mapping from input databases to output results.

Query language containment

$L_1 \subseteq L_2$ if for every query $Q_1 \in L_1$, there is an equivalent query Q_2 in L_2.

Expressiveness

- Relational Algebra $\subseteq$ Stratified Datalog^{not}
- Datalog $\not\subseteq$ Relational Algebra
  - transitive closure
- Relational Algebra $\not\subseteq$ Datalog
  - set difference

Computational complexity

Decision problem

Is a tuple t in the result $Q(D)$ of a query Q applied to a database D?

Data complexity [Var82]

Complexity as a function of the cardinality of the database D:

- fixed: database schema, query Q
- input: database D

Combined complexity

Nothing considered fixed.

Theorem

Data complexity of Stratified Datalog^{not} queries is in PTIME.

Stable model semantics [GL88]

M a subset of the Herbrand base of a Datalog^{not} program P.

Reduct P_g^M

Obtained from $ground(P)$ by the Gelfond-Lifschitz transform:

- for every $A \in M$: remove every clause that contains *not* A in the body
- for every $A \notin M$: remove *not* A from the body of every clause in which it appears.

Stable model

M is a stable model of P if M is the least (Herbrand) model of the reduct P_g^M.

Properties of stable models

- a program can have zero, one, or more stable models
- a stratified program has a single stable model computed by bottom-up evaluation.

Encoding propositional satisfiability [MT99]

Given a CNF formula ϕ with the set of clauses C and the set of propositional variables V.

Set of facts E_ϕ

- `var`(a) for every $a \in V$
- `clause`(c) for every $c \in C$
- `pos`(c, v) if v occurs positively in c
- `neg`(c, v) if v occurs negatively in c

Generating all possible truth assignments

```
(SAT1)  true(X):- var(X), not false(X).
(SAT2)  false(X):- var(X), not true(X).
```

Clause satisfaction

```
(SAT3)  sat(C):- var(X), clause(C), true(X), pos(C,X).
(SAT4)  sat(C):- var(X), clause(C), false(X), neg(C,X).
(SAT5)  f:- clause(C), not sat(C), not f.
```

Fact

M is a stable model of the program consisting of E_ϕ and `(SAT1)-(SAT5)` *iff M contains exactly the following facts for some $U \subseteq V$:*

- E_ϕ
- $\texttt{sat}(c)$ *for every* $c \in C$
- $\texttt{true}(v)$ *for every* $v \in U$
- $\texttt{false}(v)$ *for every* $v \in V - U$.

Corollary

Data complexity of checking the existence of a stable model of a Datalog^{not} program is NP-complete.

Querying using stable models

Query answer

A tuple t is a cautious query answer if $\texttt{query}(t)$ belongs to every stable model of P.

Theorem

Data complexity of computing cautious answers to Datalog^{not} queries is co-NP-complete.

K. R. Apt, H. A. Blair, and A. Walker.
Towards a Theory of Declarative Knowledge.
In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann Publishers, 1988.

M. Gelfond and V. Lifschitz.
The Stable Model Semantics for Logic Programming.
In *ICLP/SLP*, 1988.

V. W. Marek and M. Truszczynski.
Stable logic programming – an alternative logic programming paradigm.
In K. R. Apt, V. W. Marek, M. Truszczynski, and D. S. Warren, editors, *The Logic Programming Paradigm: A 25-Year Perspective*. Springer-Verlag, 1999.
Also: CoRR cs.LO/9809032.

M. Y. Vardi.
The Complexity of Relational Query Languages.
In *ACM Symposium on Theory of Computing (STOC)*, pages 137–146, 1982.