Foundations of Data and Knowledge Bases

Datalog Queries

Joachim Niehren

Links: Linking Dynamic Data
Inria Lille

November 10, 2020

What and Why

- Logic programs without function symbols.
- Expressive query language for relational structures with recursion.
- Basis of efficient algorithms.
- Datalog for words related to finite automata.

Outline

Datalog

Datalog Programs

- logic programs without function symbols.
- can define relations recursively (without negation).
- can express recursive queries like "accessibility" in a graph.

Example for Words

$q_{even}(x) :- x{=}\text{start}.$	add start to q_{even}
$q_{even}(x) :- \text{next}(y, x), q_{odd}(y).$	add successor of even positions to q_{odd}
$q_{odd}(x) :- \text{next}(y, x), q_{even}(y).$	add successor of odd positions to q_{even}
$q_{ok} :- q_{even}(\text{last}).$	accept if last position is even.

Least Fixed points Semantics

- add elements sets, only if this is required by a rule: sets remain as small as possible (least).
- continue adding elements, as long rules add something new: compute fixed point.

Datalog for Relational Structures

Syntax

terms	$t ::= a \mid x$ where $a \in \mathrm{Consts}$ and $x \in \mathrm{Vars}$
literals	$l ::= q(t_1, \ldots, t_{ar(q)}) \mid r(t_1, \ldots, t_{ar(q)})$ where $q \in \mathrm{RelVars}$ and $r \in \mathrm{Rels}$
Horn clauses	$q(t_1, \ldots, t_{ar(q)}) :- l_1, \ldots, l_n.$
Datalog Programs	P finite set of Horn clauses

Horn Clauses as Formulas

example	$q_{even}(x) :- \mathrm{next}(y, x), q_{odd}(y)$
formula	$\forall x \forall y.\ (q_{even} \leftarrow \mathrm{next}(y, x) \wedge q_{odd}(y))$

Translation of Clauses

- comma operator "," is conjunction $\wedge$
- operator $:-$ is implication $\leftarrow$
- all variables of a clause are universally quantified by the clause

Outline

Least Solutions

Solutions of Datalog programs

Fix a Σ structure S and a Datalog program P over Σ.

SO assignment: a function β that maps SO-variables $q \in \mathrm{RelVars}$ to relations $\beta(q) \subseteq \mathrm{Dom}(S)^{ar(q)}$.

FO-extension of β: a variable assignment α mapping variables $x \in \mathrm{Vars}$ to $\alpha(x) \subseteq \mathrm{Dom}(S)$ and variables $q \in \mathrm{RelVars}$ to $\alpha(q) = \beta(q)$

solution of P: a SO assignment β such that for all clauses $clause \in P$ and FO-extensions α of β: $eval_{S,\alpha}(clause) = 1$.

Order on SO variable assignments

$\beta \leq \beta'$ iff $\forall q \in \mathrm{RelVars}.\beta(q) \subseteq \beta(q')$

Theorem

Every Datalog program P over Σ has a least solution for every Σ-structure S.

Examples

Program $q(x) :- q(x).$

has least solution lfp with $lfp(q) = \emptyset$. All other SO-assignments are solutions of this program.

Least solution of program words of even length

Consider word $aabbaa$. Least solution lfp of this program for relational structure of this word satisfies:
$lfp(q_{even}) = \{0, 2, 4, 6\}$
$lfp(q_{odd}) = \{1, 3, 5\}$
$lfp(q_{ok}) = \{()\}$

When do Least Solutions Exist?

SO-formulas without least solutions

$q(x) \leftarrow \neg q(x)$: has no solution at all.

$q(a) \vee q(b)$: has two minimal solutions $\beta_1(q) = \{a\}$ and $\beta_2(q) = \{b\}$ but no least solution.

Why do Datalog programs have least solutions

Idea: lfp always contains all tuples we can collect by the rules. This objects grows continuously. It exists in the limit.

Negation spoils monotonic increase.

Disjunction spoils uniqueness of what to add.

Proof of Theorem quite easy, since fixed points are reached in finitely many steps for finite structures (avoids fixed point theory).

Compute Least Solutions

Grounding Datalog program P for finite structure S

Every FO-variable x can be instantiated with all elements of $\mathrm{Dom}(S)$. This yields ground (variable free) Datalog program of size $O(|\mathrm{Dom}(S)|^n \cdot |P|)$ where n is the number of FO-Vars of P and $|P|$ the number of letters in P.

Lemma Least solutions are preserved by grounding, relation symbols and constants of the signature are removed.

Example

program:	$q(x) :- \mathrm{next}(y, x), q(y).$
	$q(\mathrm{start}).$
structure of word :	ab
grounded program:	$q(2) :- q(1).$
	$q(1) :- q(0).$
	$q(0).$

Valid literals such as $\mathrm{next}(1, 2)$ can be safely removed.

Clauses with failed literals such as $\mathrm{next}(2, 1)$ can be safely deleted.

Efficient Algorithm for Ground Datalog

Theorem

The least fixed point of a ground Datalog programs P (over empty signature) can be computed in time $O(|P|)$.

Outline

Exercises: Finite Automata and Datalog [Homework]

The following exercises can be be done for general nondeterministic finite automata (nFAs) A and words w over the alphabet of A. However, if seems simpler for you, it will be sufficent to do these exercise for the concrete nFA A_0 below word $w_0 = 010$. The nFA A_0 has the alphabet $\Sigma = \{0, 1\}$ and two states p and q such that p is initial and q is final. The transitions of A_0 are $p \xrightarrow{0} q$ and $q \xrightarrow{1} p$.

2. Define a ground Datalog program that tests whether w is recognized by A. This yields an algorithm to decide whether w is recognized by A. How much time does this algorithm need?
3. Define a ground Datalog program that decides whether the language recognized by A is nonempty.