Top-Down beats Bottom-Up for Constraint Extensions of Datalog

David Toman

Department of Computing and Information Sciences
Kansas State University

Constraint Database

- Constraint tuples

$$\begin{array}{ll} R(1,2) & \{(x,y) : x = 1 \wedge y = 2\} \\ R(x,y), x+5<y & \{(x,y) : x+5<y\} \\ R(x,y), x \equiv_3 y & \{(x,y) : 3|x-y\} \end{array}$$

- Constraint Relation: finite set of tuples.

- Datalog program:

  finite set of clauses of the form

$$A \leftarrow D, B_1, \ldots, B_k$$

  where A and B_i are atoms and D is a constraint.

- Query (R, C):

  where R is an atom and C is a constraint.

Example

```
connection(paris, london, X, Y):-   % daily flight
        X = Mon10:00(mod 1day),
        Y = Mon11:30(mod 1day).
connection(paris, london, X, Y):-   % and weekdays
        X = Mon14:30(mod 1day),
        Y = X+1:30(mod 1day),
        Z = Mon00:00(mod 7days),
        W = Fri23:59(mod 7days), Z < X, X < W.
connection(london, toronto, X, Y):- % twice a week
        X = Tue14:30(mod 7days),    % on Tuesday
        Y = Tue16:50(mod 7days).
connection(london, toronto, X, Y):- % and
        X = Fri14:30(mod 7days),    % on Friday
        Y = Fri16:50(mod 7days).

changetime(london, X, Y) :- X+30<Y. % change planes

trip(From, To, Start, End) :-
        connection(From, To, Dtime, Atime),
        Start+15<Dtime,             % check in
        Atime+20<End.               % baggage claim
trip(From, To, Start, End) :-
        trip(From, Change, Start, Atime),
        trip(Change, To, Dtime, End),
        changetime(Change,Atime,Dtime).
```

Outline of the talk

- Abstract definition of constraint classes
- Bottom-up evaluation and termination
- Top down evaluation

  algorithm
  correctness
  complexity

- Performance analysis

Constraint Classes

Let $\mathcal{C}_0$ be a set of satisfiable atomic constraints.

1. true $\in \mathcal{C}$
2. $\mathcal{C}_0 \subseteq \mathcal{C}$
3. $C_1, C_2 \in \mathcal{C}$ and $C_1 \wedge C_2$ is satisfiable then $C_1 \wedge C_2 \in \mathcal{C}$
4. $C \in \mathcal{C}$ and x a variable then there exists a quantifier free formula $C_1 \vee \ldots \vee C_k$ (in DNF) equivalent to $\exists x.C$ such that $C_i \in \mathcal{C}$ for every satisfiable C_i
5. $C \in \mathcal{C}$ and θ is a renaming of variables then $C\theta \in \mathcal{C}$.

$\|C\|$ is the set of all valuations θ such that $\theta \models C$.

Examples of Constraint Classes

1. Equality constraints

$$x = c$$

2. Dense Order Constraints [KKR90]

$$x < y, \quad x = y, \quad x > y$$

3. Gap-order Integer Constraints [Revesz93]

$$x < u, \quad x + c < y, \quad u < x$$

4. Periodicity constraints [BaNiWo91, TChR94]

$$x \equiv_k (y + c)$$

5. Linear arithmetic constraints

6. Polynomial constraints

Constraint Operations

- Constraint Conjunction $\wedge^{\mathcal{C}} : \mathcal{C} \times \mathcal{C} \to \mathcal{C} \cup \{\perp\}$

$$\begin{array}{ccc} C_1, C_2 & \xrightarrow{\qquad\qquad \wedge^{\mathcal{C}} \qquad\qquad} & C_1 \wedge C_2 \\ \downarrow & & \downarrow \\ \|C_1\|, \|C_2\| & \xrightarrow{\bowtie} & \|C_1\| \bowtie \|C_2\| = \|C_1 \wedge C_2\| \end{array}$$

- Constraint Projection $\exists^{\mathcal{C}} : \mathcal{C} \times V \to \mathrm{Fin}(2^{\mathcal{C}})$

$$\begin{array}{ccc} C & \xrightarrow{\qquad\qquad \exists^{\mathcal{C}}_V \qquad\qquad} & \{C_i\}_{i=1}^k \\ \downarrow & & \downarrow \\ \|C\| & \xrightarrow{\pi_V} & \pi_V \|C\| \ = \ \cup_{i=1}^k \|C_i\| \end{array}$$

- Constraint Subsumption $\leq^{\mathcal{C}}: \mathcal{C} \times \mathcal{C} \to$ bool that satisfies the following condition:

$$C_1 \leq^{\mathcal{C}} C_2 \text{ implies } C_1 \supset C_2$$

Bottom-up Evaluation

- Interpretation:

  A (finite) set of constraint tuples.

- Consequence Operator:

$$\begin{aligned}\mathrm{TP}_{\mathcal{C}}(I) = I \cup \{(A, C') :\ & A \leftarrow D, B_1, \ldots, B_k \in P,\\ & (B_i, C_i) \in I \text{ for all } 0 < i \leq k,\\ & C = D \wedge C_1 \wedge \ldots \wedge C_k \text{ exists},\\ & C' \in \exists_A . C,\\ & \text{and } C' \not\leq C'' \text{ for any } (A, C'') \in I\ \}\end{aligned}$$

- Bottom-up evaluation algorithm:

$$\begin{aligned}& I := \emptyset\\ & \text{repeat}\\ & \quad J, I := I, \mathrm{TP}_{\mathcal{C}}(I)\\ & \text{while } J \neq I\end{aligned}$$

Termination Criterion

Definition:

$\mathcal{C}$ is called *constraint-compact* if

for every finite set of variables V and
for every set $\mathcal{D} \subseteq \mathcal{C}$ such that $\forall C \in \mathcal{D}. FV(C) \subseteq V$

there is a finite subset $\mathcal{D}_{fin} \subseteq \mathcal{D}$ such that

$$\forall C \in \mathcal{D}. \exists C' \in \mathcal{D}_{fin}. C \leq^{\mathcal{C}} C'$$

(i.e., $\mathcal{D}_{fin}$ covers $\mathcal{D}$ with respect to $\leq^{\mathcal{C}}$)

Theorem:

Bottom-up evaluation terminates for every constraint-compact class of constraints.

Goal-oriented Evaluation

- Bottom-up evaluation is not goal-oriented
- Two approaches to goal-oriented evaluation:
    1. Magic Set Transformation
    2. Top-down (resolution-based) evaluation

Top-down Evaluation

Starts with a forest $\{\mathsf{root}(G_0, C_0)\}$

Action/Node	Children
$\boxed{\text{Clause resolution}}$	
$\mathsf{root}(G; C)$	$\begin{cases} \mathsf{body}(G; B_1^1, \ldots, B_{k_1}^1; C \wedge D^1) \\ \vdots \\ \mathsf{body}(G; B_1^l, \ldots, B_{k_1}^l; C \wedge D^l) \end{cases}$
$\boxed{\text{Query projection}}$	
$\mathsf{body}(G; B_1, \ldots, B_k; C)$	$\begin{cases} \mathsf{goal}(G; B_1, C_1; B_2, \ldots, B_k; C) \\ \vdots \\ \mathsf{goal}(G; B_1, C_l; B_2, \ldots, B_k; C) \end{cases}$
$\boxed{\text{Answer propagation}}$	
$\mathsf{goal}(G; B, C'; B_2, \ldots, B_k; C)$	$\begin{cases} \mathsf{body}(G; B_2, \ldots, B_k; C \wedge A_1) \\ \vdots \\ \mathsf{body}(G; B_2, \ldots, B_k; C \wedge A_l) \end{cases}$
$\boxed{\text{Answer projection}}$	
$\mathsf{body}(G;\ ; C)$	$\begin{cases} \mathsf{ans}(G; A_1) \\ \vdots \\ \mathsf{ans}(G; A_l) \end{cases}$

Top-down—example

SLG-tree for goal (R, C_0)	Other SLG-trees

$$\begin{tikzcd}
\mathsf{root}(R; C_0) \arrow[d] & & \\
\mathsf{body}(R; B_1, \ldots, B_k; C_1) \arrow[d] & & \\
\mathsf{goal}(R; B_1, C_1^i; B_2, \ldots, B_k; C_1) \arrow[d] \arrow[rr, "C_1^i \leq_{\mathcal{C}} C_1'"] & & \mathsf{root}(B_1; C_1') \arrow[d, phantom, "\text{SLG-tree}"] \\
\mathsf{body}(R; B_2, \ldots, B_k; C_2) \arrow[d, phantom, "\vdots"] & & \mathsf{ans}(B_1; A_1) \arrow[ll, "\text{new } A_1"'] \\
\vdots \arrow[d, phantom, "\vdots"] & & \\
\mathsf{goal}(R; B_k, C_k';\ ; C_k) \arrow[d] \arrow[rr, "C_k^i \leq_{\mathcal{C}} C_k'"] & & \mathsf{root}(B_k; C_k') \arrow[d, phantom, "\text{SLG-tree}"] \\
\mathsf{body}(R;\ ; C_{k+1}) \arrow[d] & & \mathsf{ans}(B_k; A_k) \arrow[ll, "\text{new } A_k"'] \\
\mathsf{ans}(R; A_i) & &
\end{tikzcd}$$

Soundness, Completeness, and Termination

Theorem:

For any query (G, C, P) and any valuation ϑ

$$G\vartheta \in \mathrm{TP}^{\omega}(\emptyset) \wedge \vartheta \models C \iff \vartheta \in \| \operatorname{ans}(G, C) \|$$

Theorem:

If the class $\mathcal{C}$ is constraint-compact then the top-down evaluation terminates for all queries (G, C, P).

Complexity

Theorem:

Let $f(n)$ be a function such that $|\,\mathrm{TP}_{\mathcal{C}}^{\omega}(\emptyset)| \in O(f(|P|))$. Then

1. the forest $\mathrm{SLG}(G, C)$ has at most $O(f^2)$ nodes.
2. there are at most $O(f^2)$ applications of the rules in the evaluation of the query.

Experimental results

- Implementation in Prolog:

  naive bottom-up

  seminaive bottom-up

  seminaive bottom-up with MST

  top-down

- Comparable performance on uninstantiated queries

Test Programs

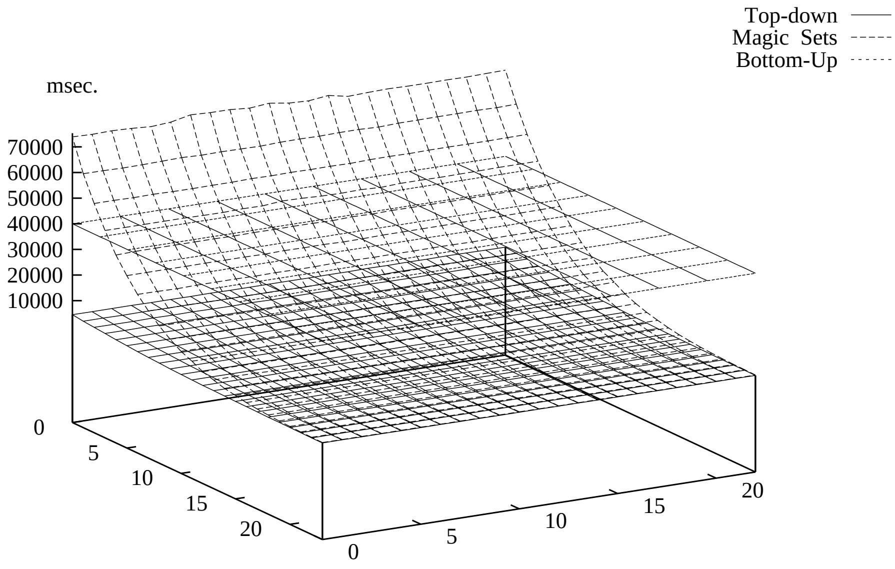
Standard Datalog:

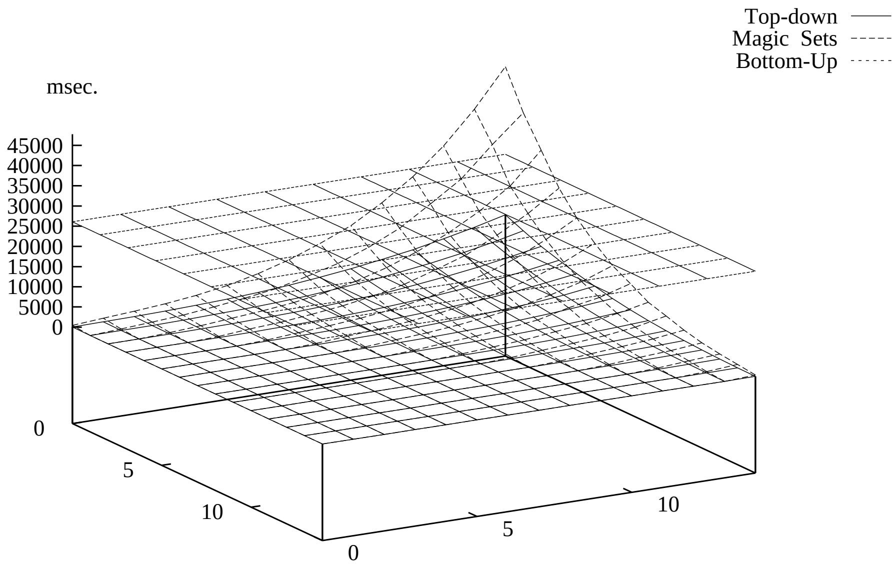
```
tc(X,Z):- tc(X,Y),tc(Y,Z).
tc(X,Y):- edge(X,Y).

edge(1,2).
   ...
edge(N-1,N).
```

Constraint Datalog$^{<Z}$:

```
path(A,C,X,Z):- X<Y<Z,
           path(A,B,X,Y), path(B,C,Y,Z).
path(A,B,X,Y):- edge(A,B,X,Y).

edge(1,2,X,Y):- 0<X<2, 1<Y<3.
      ...
edge(N-1,N,X,Y):- N-2<X<N, N-1<Y<N+1.
```

Test Results (running time)





Future Work

- Compilation of Constraints
- Static Analysis
- Interface to RDBMS
- Negation
- Storage and Access methods