Datalog with Integer Periodicity Constraints

David Toman
Jan Chomicki
David S. Rogers

Kansas State University

Requirements

- Datalog (deductive rules)
- constraints over integers (infinite sets)

Example:

```
connection(paris, london, X, Y):-       % every day flight
        X = Mon-10:00(mod 1day),
        Y = Mon-11:30(mod 1day).
connection(paris, london, X, Y):-       % and weekdays
        X = Mon-14:30(mod 1day),
        Y = X+1:30(mod 1day),
        Z = Mon-00:00(mod 7days), W = Fri-23:59(mod 7days),
        Z < X, X < W.
connection(london, toronto, X, Y):-     % twice a week
        X = Tue-14:30(mod 7days),       % on Tuesday
        Y = Tue-16:50(mod 7days).
connection(london, toronto, X, Y):-     % and
        X = Fri-14:30(mod 7days),       % on Friday
        Y = Fri-16:50(mod 7days).

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

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

Related work

- Chomicki J., Imielinski T. Temporal Deductive Databases and Infinite Objects. PODS 1988.

  Chomicki J., Imielinski T. Finite Representation of Infinite Query Answers. TODS (18) 2, 1993.

- Revesz P. A Closed Form for Datalog Queries with Integer Order. ICDT 1990.

- Wolper P. Handling Infinite Temporal Data. PODS 1990.

Plan of the talk

1. bottom-up evaluation
2. integer constraints
3. periodicity constraints, closed form evaluation
4. order constraints (overview)
5. combination of constraint classes over integers
6. combination of constraints over disjoint domains

Bottom-up evaluation of Datalog

- interpretation

$$\text{subset of } \{R(c_1, \ldots, c_k) | R, c_i \in P\}$$

- consequence operator

$$TP(I) = \{A\theta | A : -B_1, \ldots, B_n \in P \wedge B_i\theta \in I\}$$

- bottom-up evaluation

$I := \emptyset$
repeat
  $J, I := I, \mathrm{TP}(I)$
while $J \neq I$

Integer constraints

- equality constraints:

$$x = y \qquad\qquad x = c$$

  where x and y are variables and c is integer constant.

- (gap-)order constraints:

$$x <_c y \qquad\qquad x < u \qquad\qquad l < x$$

  where x and y are variables and c, u, and l are integer constants.

- periodicity constraints:

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

  where x and y are variables, c is integer, and k is natural number chosen from a fixed set.

<u>Example:</u>

$$\begin{array}{ll} R(1,2) & \{(x,y) | x = 1 \wedge y = 2\} \\ R(x,y) :- x <_5 y & \{(x,y) | x + 5 < y\} \\ R(x,y) :- x \equiv_3 (y+1) & \{(x,y) | 3 \text{ divides } x - (y+1)\} \end{array}$$

Periodicity constraints and Periodicity graphs

- periodicity graph:

  A complete graph $G = (N_G, E_G)$ over $n+1$ nodes such that $N_G = \{0, x_1, \ldots, x_n\}$ and all edges $e \in E_G$ are labeled with positive integers $\alpha(e)$ and $\beta(e)$ where $0 \leq \beta(e) < \alpha(e)$.

- normal form:

  We say that G is in normal form if for all $x_i, x_j, x_k \in N_G$

$$x_i \equiv_{\alpha(x_i,x_j)} (x_j + \beta(x_i, x_j))$$

implies

$$x_i \equiv_{\gcd(\alpha(x_i,x_k),\alpha(x_k,x_j))} (x_j + (\beta(x_i, x_k) + \beta(x_k, x_j)))$$

- coding of elementary constraints:

  1. $x_i \equiv_k (x_j + c)$: complete graph on $\{0, x_i, x_j\}$ where

	$(0, x_i)$	$(0, x_j)$	(x_i, x_j)
α	1	1	k
β	0	0	c

  2. $x_i \equiv_k c$: complete graph on $\{0, x_i\}$ where

	$(0, x_i)$
α	k
β	c

Operations on Periodicity Graphs

Proposition (Chinese Remainder Theorem)
Let $x \equiv_{k_1} (y + c_1)$ and $x \equiv_{k_2} (y + c_2)$. If $\gcd(k_1, k_2)$ divides $|c_1 - c_2|$ then there exists a unique c such that

$$x \equiv_{k_1} (y + c_1) \wedge x \equiv_{k_2} (y + c_2) \iff x \equiv_{\mathrm{lcm}(k_1,k_2)} (y + c)$$

where $0 \leq c < \mathrm{lcm}(k_1, k_2)$.
Otherwise $x \equiv_{k_1} (y + c_1)$ and $x \equiv_{k_2} (y + c_2)$ are inconsistent.

- conjunction

  combines two periodicity graphs into one by merging variables with the same name using the Chinese remainder theorem.

- projection

  actual quantifier elimination procedure: trivial elimination of a node from a normalized periodicity graph.

- consistency checking

  all normalized periodicity graphs are consistent.

- subsumption

  a periodicity graph subsumes other periodicity graph iff this relation holds for all pairs of corresponding edges in the respective graphs.

Bottom-up evaluation

- $\equiv$-interpretation

  Set of tuples (B, G) where B is predicate symbol and G is periodicity graph.

- consequence operator

$$
\begin{array}{rl}
\mathrm{TP}_{\equiv}(I) = \{ & (A, \pi_{FV(A)}(G)) : A \leftarrow B_1, \ldots, B_m, C \in P \\
& (B_i, G_i) \in I \text{ for all } 0 < i \leq m \\
& G \text{ is normal form of } G_1 \wedge \ldots \wedge G_m \wedge G_C \\
& (A, \pi_{FV(A)}(G)) \text{ is not subsumed by } I
\end{array}
$$

- bottom-up evaluation

  Standard bottom-up evaluation algorithm with $\mathrm{TP}_{\equiv}$ substituted for TP.

Complexity:

1. number of different labels on a single edge: $\alpha(P)^2$
2. maximal number of edges in a graph: $a(a+1)/2$
3. number of predicate symbols in the program: p
4. maximal number of goals in a single clause: m
5. number of clauses in the program: n

Together:

$$
n\left[\left(p\alpha(P)^{a(a+1)}\right)^m \left(mO(a^2) + O(a^4)\right)\right] p\alpha(P)^{a(a+1)})
$$

Order constraints and Gap-graphs

- gap-graphs

  directed graphs where nodes represent variables and lower and upper bounds of the constraints, and edges represent the gaps, minimal integer distances between nodes.

- operations on gap-graphs
  - conjunction (merge)
  - projection (shortcut)
  - consistency checking
  - subsumption

- bottom-up evaluation

  procedures EVAL and TEST

Combination of order and periodicity constraints

- Datalog$^{\equiv Z, <Z}$ program

  a finite set of clauses of the form

  $$A \leftarrow B_1, \ldots, B_k, C_1, C_2$$

  where A and $B_1, \ldots, B_k$ are predicate symbols, C_1 is a satisfiable periodicity constraint and C_2 is a satisfiable gap-order constraint.

- bottom-up evaluation:

  $$\begin{aligned} \mathrm{TP}_{\equiv,<}(I) = \{ \; & (A, G, H) : A \leftarrow B_1, \ldots, B_m, C_1, C_2 \in P \\ & (B_i, G_i, H_i) \in I \text{ for all } 0 < i \leq m \\ & G' \text{ is normal form of } G_1 \wedge \ldots \wedge G_m \wedge G_{C_1} \\ & H' \text{ is normal form of } H_1 \wedge \ldots \wedge H_m \wedge H_{C_2} \\ & (G, H) \in \pi_{FV(A)}(G', H')) \\ & \qquad \text{is consistent and not subsumed by } I \end{aligned}$$

  with the standard bottom-up evaluation procedure.

Complexity of consistency checking

- consistency checking of a conjunction of periodicity graph with a gap graph is NP-complete.
  Proof: By reduction of 3SAT to satisfiability of conjunctions

- consistency checking of a conjunction of periodicity graph constructed only from simple constraints with a gap graph can be computed in PTIME.
  Proof: By generalization of the consistency checking procedure for gap-graphs.

  (simple constraints are of the form $x \equiv_k c$)

- all (atomic) periodicity constraints can be expressed using simple constraints:

  <u>Example:</u>

  $x \equiv_k (y + c)$ can be defined as

$$
\begin{array}{lcll}
x \equiv_k (y + c) & \leftarrow & x \equiv_k 0 & \wedge \;\; y \equiv_k c \\
x \equiv_k (y + c) & \leftarrow & x \equiv_k 1 & \wedge \;\; y \equiv_k (c+1) \\
 & \vdots & & \\
x \equiv_k (y + c) & \leftarrow & x \equiv_k (k-1) & \wedge \;\; y \equiv_k (c-1)
\end{array}
$$

Quantifier elimination

Example:

Quantifier elimination from:

$$\exists y . x + c_1 < y \wedge y + c_2 < z \wedge y \equiv_k d$$

Result of the elimination:

$$\begin{array}{lcll}
(x + c_1 + 1) \equiv_k d & \wedge & x + c_1 + c_2 + 1 < z & \vee \\
(x + c_1 + 2) \equiv_k d & \wedge & x + c_1 + c_2 + 2 < z & \vee \\
 & \vdots & & \\
(x + c_1 + k) \equiv_k d & \wedge & x + c_1 + c_2 + k < z &
\end{array}$$

This procedure is applied on all incoming and outgoing edges in a gap graph:

Let $x + c < y$, $y + c_j < z_j$, and $y \equiv_k d$.

1. delete the edge (x, y)
2. insert edges (x, z_j) with the label $c + c_j + i$
3. add the constraint $x \equiv_k (d - c - i)$
4. check the resulting pair (G_i, H_i) for consistency and remove inconsistent pairs.

Termination of bottom-up evaluation

- Let I be a $(\equiv, <)$-interpretation. Then
  $\pi_{\equiv}(I) = \{(A, G) : (A, G, H) \in I\}$
  $\pi_{<}(I) = \{(A, H) : (A, G, H) \in I\}$.

- we can show

$$\pi_{\equiv}(\mathrm{TP}^i_{\equiv,<}(\emptyset)) \subseteq \mathrm{TP}^i_{\equiv}(\emptyset)$$

$$\pi_{<}(\mathrm{TP}^i_{\equiv,<}(\emptyset)) \subseteq \mathrm{TP}^i_{<}(\emptyset)$$

  (w.r.t. the appropriate restriction of clauses in P).

- complexity of the bottom-up evaluation is bounded by product of cardinalities of $\mathrm{TP}^i_{<}(\emptyset)$ and $\mathrm{TP}^i_{\equiv}(\emptyset)$

Equality constraints

- constraint of the form $x = y$:

$$\varphi \wedge (x = y) \iff \varphi \wedge \varphi[x/y, y/x]$$

  1) The constraint classes are closed under conjunction
  2) x and y are "equivalent" after this transformation

- constraint of the form $x = c$:

  if the constraint is consistent with the other constraints we simply eliminate x.

Constraints over disjoint domains

- Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be disjoint.

$$\begin{aligned} \mathrm{TP}_{\mathcal{S}_1,\ldots,\mathcal{S}_n}(I) = \{ \ & (A, G_1, \ldots, G_n) : \\ & A \leftarrow B_1, \ldots, B_m, C_1, \ldots, C_n \in P \\ & G_i = \mathrm{TP}_{\mathcal{S}_i}^{A \leftarrow B_1,\ldots,B_m,C_i}(\pi_{\mathcal{S}_j}(I)) \text{ exists } \} \end{aligned}$$

<u>Example:</u>

Datalog over standard "uninterpreted constants" in D (Datalog$^{=D}$)

1. a =-interpretation is a set of tuples (A, R_A) where A is a predicate symbol and R_A is a A-tuple over D.
2. a conjunction of two atoms (A, R_A) and (B, R_B) is a tuple where fields of R_A are matched against fields of R_B with the same name,
3. a conjunction of atoms is consistent if the matching exists,
4. a projection operation is removing of a field in the tuple, and
5. (A, R_A) subsumes (A, S_A) iff $S_A = R_A$

We can combine Datalog$^{=D}$ with Datalog$^{<Z,\equiv Z,=Z}$ using the standard bottom-up procedure with $\mathrm{TP}_{=D,<Z,\equiv Z,=Z}$.

Conclusion

- Closed bottom-up evaluation and tuple recognition for Datalog$^{\equiv Z}$ in PTIME
- Closed bottom-up evaluation for Datalog$^{\equiv Z,<Z}$
- Tuple recognition for Datalog$^{\equiv Z,<Z}$ in PTIME.
- Combination of arbitrary classes of constraints (over disjoint domains)

Open problems

- other "well behaved" classes of constraints over integers
- addition of limited negation
- expressiveness of the language
- efficient implementation (optimization techniques)