Querying Incomplete Information in Semistructured Data

Problem

- Semistructured data – information lacks homogeneous structure and is incomplete.
- The incompleteness of information has not been reflected by query languages.
- Query Model should allow for answers that are *partial* – not all variables in the query might be bound by the answer.
Overview

- Semistructured database is modeled as a labeled directed graph.
- Search phase – a query graph containing variables is matched to maximal portions of database graph.
- Filter phase – the obtained matchings are subject to constraints to obtain solutions.
- Answers are derived from the obtained solutions.
Data Model

- Data and Queries are represented by labeled directed graphs with \( A \) an infinite set of atoms and \( L \) an infinite set of labels.

- Labeled Directed Graph or \( ldg \) over a set of nodes \( N \) is a pair \( G=(N, \cdot^G) \) where \( \cdot^G \) assigns to each label \( I \), a binary relation \( I^G \) between the nodes.

- Database consists of
  - a rooted finitely branching \( ldg \) \( (O, r_D, \cdot^D) \) over a set of objects \( O \).
  - a function \( \alpha \) that maps each terminal node to an atom.
  - \( D = (O, r_D, \cdot^D, \alpha) \)
University Database

FIG. 1. A university database.
Query Language

- A query $Q = (G, F, x)$ where
  1. $G = (V, r_G, \cdot^G)$ is an ldg called the *query graph*.
  2. $F$ is a set of filter constraints.
  3. $x$ is a tuple of variables occurring in $V$.

*FIG. 2.* Query graph $G_1$ asking for course teachers and lab instructors.
Search Constraints

- We can look upon the Query graph $G$ as a set of constraints $\text{Cons}(G)$ containing:
  - a root constraint $r_G$ and
  - Edge constraints $ulv$ such that $v \in l^G(u)$.
- $D$-assignment over $V = \mu: V \rightarrow \{\bot\}$
- Total assignment $\rightarrow$ all variables in $V$ are bound.
- Otherwise it is a partial assignment.
- $\mu$ satisfies an edge constraint $ulv$ if $\mu(v) \in l^G(\mu(u))$.
- $\mu$ is a strong matching for $G$ if
  - $\mu(r_G) = r_D$
  - $\mu$ satisfies every edge constraint in $G$. 
More Matchings

- $\mu$ is a **weak matching** of $Q$ if
  - whenever $\mu$ is defined for $u$ and $v$ and $G$ contains an edge constraint $ulv$, then $\mu$ satisfies $ulv$. ($\text{Mat}_D^w(Q)$)

- $\mu$ is a **AND matching** of $Q$ if
  - whenever $\mu$ is defined for $v$, then $\mu$ satisfies all the incoming constraints of $v$. (All constraints of the form $ulv$ are incoming constraints of $v$). ($\text{Mat}_D^\wedge(Q)$)

- $\mu$ is a **OR matching** of $Q$ if
  - whenever $\mu$ is defined for $v$, then $\mu$ satisfies some incoming constraints of $v$. ($\text{Mat}_D^\vee(Q)$)

It is assumed that the root constraint is always satisfied.
Some further comments

- $\text{Mat}_D^s(Q) \leq \text{Mat}_D^\wedge(Q) \leq \text{Mat}_D^w(Q) \leq \text{Mat}_D^\vee(Q)$
- If the query graph is a tree then $\text{Mat}_D^\wedge(Q) = \text{Mat}_D^w(Q) = \text{Mat}_D^\vee(Q)$
- $\mu'$ is said to subsume $\mu$ ($\mu \leq \mu'$) if whenever $\mu(v)$ is defined, then $\mu'(v)$ is defined and $\mu(v) = \mu'(v)$.
- An assignment $\mu$ is a maximal element of a set of matchings $A$, if for all $\mu' \in A$,
  $$(\mu \leq \mu') \Rightarrow (\mu = \mu')$$
- Denote by $\text{MMat}_D^\sigma(Q)$ the set of maximal elements of $\text{Mat}_D^\sigma(Q)$. 


Example: query graph 1

<table>
<thead>
<tr>
<th>Semantics</th>
<th>( t )</th>
<th>( u )</th>
<th>( v )</th>
<th>( w )</th>
<th>( y )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>11</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>21</td>
<td>( \perp )</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>12</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>22</td>
<td>( \perp )</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>23</td>
<td>32</td>
<td>33</td>
<td>( \perp )</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>14</td>
<td>13</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>Weak</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>( \perp )</td>
<td>17</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>( \perp )</td>
<td>10</td>
<td>19</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>11</td>
<td>20</td>
<td>30</td>
<td>31</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>12</td>
<td>23</td>
<td>32</td>
<td>33</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>23</td>
<td>32</td>
<td>33</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>14</td>
<td>13</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>25</td>
<td>24</td>
</tr>
<tr>
<td>OR</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>17</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>10</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>11</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>12</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>23</td>
<td>32</td>
<td>33</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>14</td>
<td>13</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>( \perp )</td>
<td>25</td>
<td>24</td>
</tr>
</tbody>
</table>

FIG. 2. Query graph \( G_i \) asking for course teachers and lab instructors.
FIG. 1. A university database.
Results

- Note all the maximal matchings are partial assignments – **strong semantics** would not retrieve any answer to our query!
- Under **AND semantics** it queries for people who are both a course teacher and a lab instructor.
- Under **OR semantics** it queries for people who are either a course teacher or lab instructor.
- Under **weak semantics** it does not translate to any meaningful result. We will see example of weak semantics at work later on.
Filter Constraints

Filter constraints enforce some conditions on the values bound to some of the variables during the search phase.

- **Atomic constraints**: \( r(s_1, \ldots, s_n) \) like equality or inequality of values.
- **Object comparisons**: like \((u =_o v)\) or \((u \neq_o v)\), used for comparing database objects for identity and uniqueness.
- **Existence constraints**: \(!v\) which expresses the condition that variable \(v\) should be bound.
Applying Filter Constraints

Filter constraints are applied to maximal matchings, in which certain variables may be undefined.

**Strong satisfaction:**
- $\mu$ strongly satisfies an atomic constraint $r(s_1, \ldots, s_n)$ if for all $s_i$, $\mu(s_i)$ is defined and equal to a terminal database node and the tuple $(\alpha(\mu(s_1)), \ldots, \alpha(\mu(s_n)))$ is in the relation $r$.
- $\mu$ strongly satisfies the object equality $u =_o v$ if $\mu(u)$ and $\mu(v)$ are defined and $\mu(u) = \mu(v)$.

**Weak satisfaction:**
- an assignment need not be defined for all variables in the constraint; however if the assignment is defined for all variables, then the values must accord with the constraint relation.
Solutions and Answers

- \( \text{Sol}_D^\sigma(Q) = \{ \mu \in \text{MMat}_D^\sigma(Q) \mid \mu \models_s F_s \text{ and } \mu \models_w F_w \} \)
- \( \text{Ans}_D^\sigma(Q) = \{ \mu(x) \mid \text{for all } \mu \in \text{Sol}_D^\sigma(Q) \} \)
- For query graph 1, if \( Q_1 = (G_1, F_1, x) \) with \( F_1 = \{!y\} \) and \( x = (x_1, x_2, x_3, x_4, x_5) \) we get

<table>
<thead>
<tr>
<th>Semantics</th>
<th>First name</th>
<th>Family name</th>
<th>Seniority</th>
<th>Course name</th>
<th>Lab name</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>David</td>
<td>Ben-Yishay</td>
<td></td>
<td>Logics</td>
<td>Logic programming</td>
</tr>
<tr>
<td>Weak</td>
<td>David</td>
<td>Ben-Yishay</td>
<td></td>
<td>Logics</td>
<td>Logic-programming</td>
</tr>
<tr>
<td></td>
<td>Ruth</td>
<td>Efrat</td>
<td>15</td>
<td>Spectography</td>
<td>Polymers</td>
</tr>
<tr>
<td></td>
<td>Ruth</td>
<td>Efrat</td>
<td>20</td>
<td>Calculus</td>
<td>Polymers</td>
</tr>
<tr>
<td></td>
<td>David</td>
<td>Ben-Yishay</td>
<td></td>
<td>Logics</td>
<td>Logic-programming</td>
</tr>
</tbody>
</table>
Example: Weak Filter Constraints

- With weak OR semantics we would get information about people teaching a course or instructing a lab, as before, but in addition we would also receive information about a course which a lab instructor does not teach and a lab which a course teacher does not instruct!

- Example: We are interested only in staff of seniority of at least 20 years. Setting $F_s = \{y, x_3 \geq 20\}$ will reject people for whom the database does not have the seniority information. Thus it makes more sense to set the constraint $F_w = \{y, x_3 \geq 20\}$.
Example: OR semantics over weak semantics

The objective is to find all people who are either course teacher, lab instructor or chairman of a department. Weak semantics would only return chairmen!
University Database

FIG. 1. A university database.
Computing Matchings

- Tree queries (*EvalTreeQuery*):
  - Topologically sort the nodes on the basis of the edge relation, $v_0 < v_1 < ... < v_n$.
  - Assign the first node to the root of the database.
  - Traverse the list of nodes in topological sort order sequentially satisfying the edge constraints $ulv$.

- If the *EvalTreeQuery* is called with a tree query $Q$ and a database $D$, then its output $S$ is the set of maximal matchings under AND, OR and weak semantics. The runtime of the algorithm is $O(|D|*|Q|*|S|)$. 
Computing Matchings: AND/OR semantics for acyclic query graphs

Algorithm EvalDagQuery(Q, D, σ);
Input dag query Q, database D, semantics σ ∈ {∧, ∨};
Output the set of maximal σ-matchings of Q over D;
let v₀ < v₁ < ... < vₖ be a topological order on the variables of Q;
let μ₀ := [v₀/τ_D];
let S₀ := {μ₀};
for i = 1 to k do
    Sᵢ := ∅;
    for each μ ∈ Sᵢ₋₁ do
        let E := Extᵦᵢ(vᵢ)
        if E = ∅ then
            Sᵢ := Sᵢ ∪ {μ ⊕ [vᵢ/⊥]}
        else
            Sᵢ := Sᵢ ∪ {μ ⊕ [vᵢ/o] | o ∈ E}
    return Sₖ

Fig. 4. Computing maximal matchings for a dag query.

Extᵦᵢ(vᵢ) := {o ∈ O | μ ⊕ [vᵢ/o] ∈ Matᵦᵢ(Qᵢ)}.

Complexity = O(|D|*log|D|*|Q|²*|S|)
Computing Matchings: Weak Semantics

Algorithm `EvalWeakDagQuery(Q, D)`;
Input dag query Q, database D;
Output the set of maximal weak matchings of Q over D;
let \( v_0 < v_1 < \ldots < v_k \) be a topological order on the variables of Q;
let \( \mu_0 := [v_0/r_D] \);
let \( S_0 := \{\mu_0\} \);
for \( i = 1 \) to \( k \) do
  let \( S_i := \emptyset \);
  for each \( \mu \in S_{i-1} \) do
    \( S_i' := S_i' \cup \{\mu \oplus [v_i/\bot]\} \);
  \( S_i := S_i' \cup \{\text{ExtendAssignment}(Q, D, \mu, v_i, o) \mid o \in \text{Ext}_v^w(v_i)\} \);
return the maximal elements of \( S_k \);

Algorithm `ExtendAssignment(Q, D, \mu, v, o)`;
Input dag query Q, database D, partial assignment \( \mu \), query variable \( v \), database node o;
Output a weak matching created from \( \mu \) by adding \([v/o]\);
let \( \mu' := \mu [v/o] \);
for each edge \( uv \) in Q do
  if \( \mu' \) does not satisfy \( uv \) then
    \( \mu' := \mu' \oplus [u/\mu'(u)] \oplus [v/\bot] \);
while there is a node \( w \) in Q, where \( w \neq r_Q \) such that \( \mu' \) is undefined for all parent nodes of w do
  \( \mu' := \mu' \oplus [w/\mu'(w)] \oplus [w/\bot] \);
if \( \mu'(r_Q) = \bot \) then
  \( \mu'(r_Q) := r_D \);
return \( \mu' \);

FIG. 5. Computing maximal matchings of a dag query in weak semantics.

FIG. 6. Adding a node assignment to a dag query assignment in weak semantics.

Complexity = \( O(|D|^{\log|D|} \cdot |Q|^3 \cdot |S|) \)
Some theoretical results

1. If there is an algorithm that for arbitrary query graphs $Q$, computes $\text{MMat}_D^\wedge(Q)$ in time polynomial in the input and the output, then $P=NP$.

2. Under only existence constraints to tree queries, the set of maximal filter solutions under any of strong, AND, weak or OR semantics can be computed in time polynomial in the size of the query, the database, and the solution set.

3. Evaluation of dag queries with existence constraints under AND, weak or OR semantics is $NP$-complete.

4. Evaluation problem for tree queries with weak equality/inequality constraints is $NP$-complete.
A language based on these ideas has been designed and implemented at Hebrew University. The language is a part of a system to facilitate access to the World Wide Web.

In this model queries do not allow regular path expressions because it is difficult to model them using first order logic. Further work needs to be done in this direction to make this query language more practically applicable.