Show-Me the Links: Supporting Path Extraction Queries in Semantic Web Databases

Kemafor Anyanwu
LSDIS lab, University of Georgia

“... Everything’s connected, all along the line. Cause and effect. That’s the beauty of it. Our job is to trace the connections and reveal them.”

Jack in Terry Gilliam’s 1985 film - “Brazil”

Outline
- Motivation
  - Why database support for subgraph analysis?
- Background
  - Semantic Web databases
- SPARQ2L
  - Formal syntax and semantics
- Evaluating path extraction queries
  - constraints
- Query result management
  - result ranking
- Future plans

Abstraction levels for data analysis

Pattern Analysis
Most people who buy cameras buy color printers!

Pattern Match Analysis
Which persons bought expensive jewelry?

Abstraction levels for data analysis
Subgraph analysis

- **Subgraph Analysis** – finds interesting subgraphs
- **Applications in**
  - E-science
  - Bioinformatics
    - pathway analysis
  - Homeland security
    - Is a flight passenger or suspect linked to a terrorist organization?
- **Interestingness may be based on the**
  - presence/absence specific nodes, edges, patterns
  - or substructures e.g. center-piece subgraphs

Subgraph analysis contd.

- Can be viewed as a query “extracting a subgraph” from a data graph
  - Retrieve the interaction network for all genes known to be *differentially regulated in advanced stage epithelial ovarian cancer*
- Correct interpretation of **constraints** may need machine processible semantics

How can semantics help?

- **Differentially regulated**
  - Micropapillary Serous Carcinoma
  - Serous Carcinoma
  - Mucinous Carcinoma
  - Epithelial Ovarian Cancer
  - Fibroma
  - Clear cell adenoma
  - Differentially regulated
- **Up-regulated**
- **Down-regulated**
How can semantics help?

Semantic Web languages - RDF

- SW layer cake - layers of increasingly expressive languages
  - XML – syntax
  - Resource Description Framework (RDF) -- semantics
- In the RDF model
  - Each SW resource (entity/relationship) has an IRI
  - Statements about resources made in the form of triples (subject, property, object)
    - e.g. Kemafor has email address “anyanwu@cs.uga.edu”
    - (X:kemafor, Y:hasEmail, “anyanwu@cs.uga.edu”)
    - X and Y are aliases for namespaces e.g. http://somelong.url/
- RDF Schema provides a metaschema for describing classes and relation types

The anatomy of an RDF database

An example RDF database
Querying RDF databases

- SPARQL – W3C’s graph pattern matching query language
- SPARQ2L as an extension to SPARQL with constructs for expressing subgraph extraction queries
- Directed paths as basis with operators for constructing more complex subgraphs (Anyanwu et al, 01, 02, 03)

Outline

- Motivation
  - Why database support for subgraph analysis?
- Background
  - Semantic Web databases
- SPARQ2L
  - Formal syntax and semantics
- Evaluating path extraction queries
  - With constraints
- Query result management
  - Result ranking
- Future plans

Formal algebraic syntax for SPARQL

- Let the set of RDF terms i.e. IRIs, literals and blank nodes be called $T$
Let the set of RDF terms i.e. IRIs, literals and blank nodes be called $T$

- A term variable $?x$ ranges over $T$

A triple pattern is a triple with a term variable

- $(?x, \text{email}, ?y)$

Triple patterns may be combined to form graph patterns using the following operators

- AND, FILTER, UNION, OPTIONAL

Examples

- $(?x, \text{email}, ?y) \land (?x, \text{age}, ?z)$
- $(?x, \text{email}, ?y) \land (?x, \text{age}, ?z) \land (?z > 15)$
A path \( pt \) - sequence of connecting triples
\[
(s_1, p_1, o_1), (s_2, p_2, o_2), \ldots, (s_k, p_k, o_k) : s_i = o_{i-1}
\]
\( s_i / o_k \) are source /destination of resp.

A path variable \( ??p \) ranges over subsets of \( T \)

A tp-pattern is a triple pattern with a path variable in the predicate position
\[
(x, ??p, y)
\]

PATHFILTER using built-in path functions
\[
\text{containsAny} (\text{listOfResources}), \text{containsAll} (\text{listOfResources}), \text{containsPattern} (\text{regularPathExpression})
\]

A SPARQ2L graph pattern – SPARQL graph patterns, tp-patterns and PATHFILTER conditions (Anyanwu et al 07)

---

Constraints in path extraction queries

- **Examples**
  - **paths must contain a specific node type**
    - paths from an MTB surface molecule to *via a Phosphoinositide 3-Kinase enzyme* to a cellular response event. [Hsing et al *Current Bioinformatics* 2006:1]
  - **paths lengths are bounded**
    - *close connections (less than 4 hops)* between SalesPersonA and CIO-Y. [Business Week April 2006]
  - **paths must contain a specific pattern**
    - paths from authorX to reviewerY that *involves a “knows • coauthor” pattern* [Aleman et al *WWW2006*]

---

Foundation for SPARQ2L semantics

A mapping \( \mu \) is a partial function from term variables to RDF Terms \( T \)

**Example triple pattern** - \( t \)

\[
(?X, \text{course_title}, ?Y)
\]
Foundation for SPARQ2L semantics

A mapping \( \mu \) is a partial function from term variables to RDF Terms \( T \).

**Example triple pattern - \( t \)**

\( (\mathcal{X}, \text{course	extunderscore title}, \mathcal{Y}) \)

**Extended semantics for SPARQ2L**

A pmapping \( \omega \) is a partial function from \( \mathcal{V}_T \cup \mathcal{V}_P \) to \( 2^\mathcal{T} \) such that for:

\( x \in \mathcal{V}_T \), \( \omega(x) \) maps to a singleton set of \( 2^\mathcal{T} \).

**Example tp-pattern \( tp \)**

\( (\mathcal{S}_1\mathcal{A}_1, \mathcal{P}_\omega, \mathcal{P}_1) \)

**Evaluation of \( tp \)**

\( \omega_1 \)

\( \omega_2 \)

\( \{\mathcal{S}_1\mathcal{A}_1, \text{advisor, P1}\} \)

\( \{\mathcal{S}_1\mathcal{A}_1, \text{enrolled	extunderscore in}, \mathcal{C}_3, \text{taught	extunderscore by}, \mathcal{P}_1\} \)

The evaluation of \( tp \) is the set of pmappings that cause \( tp \) to "match" the graph.

Incompatible pmappings: Mappings that agree on their shared variables.
Foundation semantics contd.

Compatible mappings: Mappings that agree on their shared variables.

<table>
<thead>
<tr>
<th>?X</th>
<th>?Y</th>
<th>?U</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Semantic Web</td>
<td>Bk1</td>
</tr>
<tr>
<td>C3</td>
<td>Bk1</td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>Database</td>
<td></td>
</tr>
</tbody>
</table>

Compatible mappings can be merged:

<table>
<thead>
<tr>
<th>?X</th>
<th>?Y</th>
<th>?U</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Semantic Web</td>
<td>Bk1</td>
</tr>
<tr>
<td>C3</td>
<td>Bk1</td>
<td></td>
</tr>
</tbody>
</table>

Let \( \omega_2 \cup \omega_3 \) be the sets of mappings \( M_1 \) and \( M_2 \) that can merge.

Semantics of PATHFILTER

A mapping \( _\_ \) satisfies \( \models \) the conditions:

- \( F \) is containsAny(\( \_\_P, L' \)), if \( L' \neq \_\_P \).
- \( F \) is containsAll(\( \_\_P, L' \)), if \( L' \subseteq \_\_P \).
- \( F \) is containsPattern(\( \_\_P, tr \)) if \( \text{ground}(tr) \) is a subpath of \( \_\_P \).
- \( F \) is isSimple(\( \_\_P' \)) if \( x, y \in \_\_P, x \neq y \).
- \( F \) is (\( \neg F_1 \)), if \( _\_ \not\models F_1 \)
- \( F \) is (\( F_1 \land F_2 \)), if \( _\_ \models F_1 \) and \( _\_ \models F_2 \)

Let \( M_1 \) and \( M_2 \) be the evaluations of \( PP_1 \) and \( PP_2 \) resp. Then evaluation of (\( PP_1 \land PP_2 \)) is \( M_1 \models M_2 \).
Semantics of PATHFILTER

A pmapping _ satisfies |= the conditions:
- F is containsAny(??P,L'), if L' _ (??P) ≠ _.
- F is containsAll(??P, L'), if L'⊆ _(??P).
- F is containsPattern(??P, tr) if ground(tr) is a subpath of _ (??P).
- F is isSimple(??P') if x, y ∈ _ (??P), x ≠ y.
- F is (¬F1), _ |≠ F1
- F is (F1 ∧ F2), _ |= F1 and _ |= F2

The evaluation of (PP PATHFILTER F) is the set of pmappings that “satisfy” F.

Outline

- Motivation
  - Why database support for subgraph analysis?
- Background
  - Semantic Web databases
  - SPARQ2L
    - Formal semantics
- Evaluating path extraction queries
  - constraints
- Query result management
  - result ranking
- Future plans

Path query evaluation - issues

- Most approaches focus on main memory dbs
- Limited support of constraints i.e. path filter conditions
- Our goal:
  - a good linear representation for general graphs
  - provide good performance for different classes of path extraction queries (Anyanwu et al, 07)

Representation wish list

- All queries should be answerable using a single scan
- Labeling scheme for efficient pruning
- Can contain partial path information
- Clustering of “related” path information
- Compact representation of path information
A P-Expression of type \((u, v), (P, u, v)\), is a regular expression \(P\) over \(E\) such that \(s \in L(P)\) represents a path from \(u\) to \(v\).

**Example**
Assume \(E = (u, p_1, w), (u, p_2, w), (w, p_3, v)\) then
\((u, p_1, w) \cup (u, p_2, w) \ast (w, p_3, v)\) is an p-expression of type \((u, v)\).

The Path Sequence for a graph \(G\) is the sequence
\((P_1, s_1, d_1), (P_2, s_2, d_2), (P_3, s_3, d_3), \ldots, (P_f, s_f, d_f), \ldots, (P_g, s_g, d_g), \ldots, (P_l, s_l, d_l)\)

\[ p = p_1, \quad p_2, \ldots, \quad p_k \]
for any non-empty path \(p\) in \(G\).
LU decomposition of a graph’s matrix!!!

Path Sequence PS for G

1: (c, 1, 2)  2: (d, 1, 3)  3: ((a • c)*, 2, 2)
4: (a • d, 2, 3)  5: (e • (a • c)* • e, 2, 5)  6: (f, 3, 4)
7: (d, 3, 5)  8: (h, 3, 6)  9: (g, 5, 4)
10: (a, 2, 1)

u < v:
paths from u to v with no intermediate vertex w < u.

u ≥ v:
paths from u to v with no intermediate vertex w > v.

(2, 2) = λ

PS = p-expressions with u ≤ v in increasing order of u, followed p-expressions u > v in decreasing order of u.
Solving (2, 6)

(2, 2) = \lambda

(2, 2) \cdot (1, 2) = (2, 2) = \emptyset

(2, 2) \cdot (1, 3) = (2, 3) = \emptyset

O(\text{pathsequence length}) !!
Indexing a path sequence

1. Index a path sequence using B+tree and answer queries using extended range queries.
2. Cluster keys based on the notions of prunability and prunability equivalence

Let $Q = (s, d)$ be a query and $PS$ be the path sequence for a data graph $G$.

Prunability – A p-expression $pe$ is said to be prunable from $PS$ if $Q$ can be solved using $PS - pe$

Two p-expressions $pe_1$, $pe_2$ are prunable equivalent with respect to $Q$ if determining the prunability of $pe_1$ leads us to conclude the prunability of $pe_2$
Nodes of a strong component are prunable equivalent e.g. 8, 9 and 10

Edges across strong components are also prunable equivalent

We can refine the partitioning of nodes and edges in nontree subgraphs using an optimal spanning tree (OST). Only those edges in the graph that lie on the longest path from root to a node are tree-induced prunability equivalent.
2-Color code

- Label p-expression for each scc with three identifiers: subgraph identifier, tree-level identifier and a traversal identifier.
- 2-Color code is a sequence of key-value pairs:
  - for scc_i, [(s, l, t)], (s, l, t)]◊ PS]
  - for scc_x, scc_y connected by edges e_1, e_2, .. e_k, [((s, l, t)_x, (s, l, t)_y)◊ (pe_e1, pe_e2, .. pe_ek)]
- 2-Color code preserves path sequence ordering.

Cost of Preprocessing

- Find strong components of G - O(n + m)
- Find roots of dangling trees - O(n + m)
- Find optimal spanning tree - O(n + m)
- Find PS for each strong component i in increasing order of level in OST – O Σn_i^3
Properties of a 2-Color code

◆ Order Property: for $G_N / G_T$
  - $u \in V(G_N), v \in V(G_T) \implies \text{label}(u) \text{ precedes} \text{label}(v)$.
  - $e = (u, v) \in E(G_T), u \in V(G_N) \implies \text{label}(u) \text{ precedes} \text{label}(e)$
  - $u$ in $V(G_T)$ and $e = (v, u) \in E(G_T) \implies \text{label}(u) \text{ succeeds} \text{label}(e)$

◆ NonReachability Property: for $(s_u, l_u, t_u), (s_v, l_v, t_v)$
  - $s_u \neq s_v \implies \text{result is empty}$.
  - $l_u \geq l_v \implies \text{result is empty}$.
  - for query $(u, v)$ with levels $i, j$, any node $w$ with level $k$ with $k < i$ or $k > j$ is prunable

Evaluation

◆ Strategy
  - 2-color code vs. relational databases using joins
    - strawman comparison
  - 2-color code vs. randomly chosen topological orderings

◆ Datasets

<table>
<thead>
<tr>
<th></th>
<th>UBA6</th>
<th>SWETO_DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>118,566</td>
<td>724,874</td>
</tr>
<tr>
<td>Number of edges</td>
<td>357,950</td>
<td>836,555</td>
</tr>
<tr>
<td># of strong components</td>
<td>118,256</td>
<td>723,669</td>
</tr>
<tr>
<td># of p-expressions</td>
<td>476,448</td>
<td>1,561,008</td>
</tr>
</tbody>
</table>

◆ Queries
  - 6 query classes
  - NT-NT, NT-T, T-T
  - Positive, Negative
  - $40 \times 6$ queries

Constrained Path Extraction Queries

◆ Inline and Post-filtering approaches

◆ Bounded length paths – inline
  - associate $(P, u, v)$ with 2 mappings: cost, shortestpath (Tarjan81)

◆ ContainsAll(list) – postfiltering
  - Related to DisjointPaths, HamPath problems

◆ Our approach
  - Compute $(P, u, v)$
  - Check if given nodes/edges satisfy constraints
  - Then filter
Bit encoding of p-expressions

(1, a, 2) (2, c, 3)
(1, a, 2) (2, d, 3)
(1, b, 2) (2, c, 3)
(1, b, 2) (2, d, 3)

→ (a ∪ b) • (c ∪ d)

1, 2

U

1, 3

a

b

U

1, 2

2, 3
c
d

LU / U 00
RU 01
LD 10
RD / D 11

<table>
<thead>
<tr>
<th>State</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU / U</td>
<td>00</td>
</tr>
<tr>
<td>RU</td>
<td>01</td>
</tr>
<tr>
<td>LD</td>
<td>10</td>
</tr>
<tr>
<td>RD / D</td>
<td>11</td>
</tr>
</tbody>
</table>

Example contains \( \text{ALL}(p, \{a, b\}) \)
a and b will agree on some suffix beginning with a 1 in an even position but ….
will disagree on a preceding odd bit position

Outline

- **Motivation**
  - Why database support for subgraph analysis?
- **Background**
  - Semantic Web databases
  - Semantic Web query languages
  - W3C’s SPARQL
- **SPARQ2L**
  - Formal semantics
- **Evaluating path extraction queries**
  - With constraints
- **Query result management**
  - Result ranking
- **Future plans**
SemRank

◆ Modulative Ranking
◆ Metrics (Anyanwu et al 05)
  ◦ Refraction Count
    ◦ Measures how different a path is from the paths at the schema layer?
  ◦ Semantic Information Gain - SIG
    ◦ How much information does a user gain or how much uncertainty is removed by informing a user of a result?
  ◦ S-Match
    ◦ Best semantic match with user description (if provided)
  ◦ NodeRank (Anyanwu et al 08**)
    ◦ biased PageRank vectors to generate query-specific ranking
◆ Human based evaluation

Other Stuff

◆ Temporal and Spatial constraints
  ● Matt Perry – Wright State University
◆ Semantic Query Optimization of Complex OLAP
  ● Senior collaborators – Umesh Dayal – HPLabs - Fellow
◆ Modeling Inter-organization workflows
  ● Wil van der Aalst – Technische Universiteit Eindhoven
◆ Graph Pattern Mining and Summarization
  ● Angela Maduko, LSDIS lab
◆ Keyword and Natural Query Support in Semantic Web Databases
  ● Sujeeth Thirumalai, Ravi Pavagada (Verizon wireless)
Future Plans

- General theme – Semantic Web Database support for Xinformatics
- Modeling
  - Model transformations to semantic graph models
- Querying
  - Pattern Matching queries - ongoing
  - Semantic Similarity Matching queries
  - Data mining coupling: Annotation, Storing and Querying of Mined Patterns
- Usability of analysis tools
  - Keyword and natural languages queries on RDF databases - ongoing
  - Visual and interactive query interfaces

Publications

Core Thesis publications
- Kemafor Anyanwu, Angela Maduko, Amit Sheth. From Link Analysis Ranking to Relationship Analysis Ranking - Adding Semantics to the Mix. (to be submitted for second round reviews for the Journal of Web Semantics)
- Angela Maduko, Kemafor Anyanwu, Amit Sheth, Estimating the Cardinality of RDF Graph Patterns. WWW2007 poster paper

Application

Workflow management


Thank you!!