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
Abstraction levels for data analysis

**Pattern Analysis**

Most people who buy cameras buy color printers!

Most cameras were bought by men!

**Pattern Match Analysis**

Which persons bought expensive jewelry?
Abstraction levels for data analysis

Pattern Analysis
Most people who buy cameras buy color printers!

Pattern Match Analysis
Which persons bought expensive jewelry?

Subgraph Analysis
Peter is linked to Mr. Badguy through checking account!!!
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?

<table>
<thead>
<tr>
<th>Micropapillary Serous Carcinoma</th>
<th>Epithelial Ovarian Cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mucinous Carcinoma</td>
<td></td>
</tr>
<tr>
<td>Fibroma</td>
<td></td>
</tr>
<tr>
<td>Clear cell adenoma</td>
<td>Serous Carcinoma</td>
</tr>
<tr>
<td>Up-regulated</td>
<td>Differentially regulated</td>
</tr>
<tr>
<td>Down-regulated</td>
<td></td>
</tr>
</tbody>
</table>
How can semantics help?

Micropapillary Serous Carcinoma
Mucinous Carcinoma
Fibroma
Clear cell adenoma
Serous Carcinoma
Epithelial Ovarian Cancer

Up-regulated
Down-regulated

Is_a
Is_a
Is_a
Is_a

Is
Is

Differentially regulated
How can semantics help?

- XML: semantics implied by document structure – too optimistic for heterogeneous environments like the Web
- Semantic Web standards allows us express the “meaning” of relationships in an unambiguous way
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)

Path Join nodes
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\textsuperscript{1}
Let the set of RDF terms i.e. IRIs, literals and blank nodes be called $T$. 

\(^1\)
Let the set of RDF terms i.e. IRIs, literals and blank nodes be called $T$

A term variable $?x$ ranges over $T$
Formal algebraic syntax for SPARQL

- 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 $(?x, \text{email}, ?y)$
Formal algebraic syntax for SPARQL$^1$

- 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
Formal algebraic syntax for SPARQL

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

\( (?x, \text{email}, ?y) \) AND \( (?x, \text{age}, ?z) \)

\( (?x, \text{email}, ?y) \) AND \( (?x, \text{age}, ?z) \) FILTER \( (?z > 15) \)
SPARQ2L- concepts
SPARQ2L- concepts

- **A path** $pt$ - sequence of *connecting* triples
  
  - $(s_1, p_1, o_1), (s_2, p_2, o_2),..., (s_k, p_k, o_k) : s_i = o_{i-1}$
  
  - $s_1 / o_k$ 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
  
  - `containsAny(listofresources), containsAll(listofresources), containsPattern(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]
A mapping $\mu$ is a partial function from term variables to RDF Terms $T$

**Example triple pattern - $t$**

$(?X, \text{course\_title}, ?Y)$
A mapping $\mu$ is a partial function from term variables to RDF Terms $T$

**Example triple pattern - $t$**

$(?X, \text{course}_\text{title}, ?Y)$

**Example database**

$\mu_1(C1, \text{course}_\text{title}, \text{"Semantic Web"})$,  
$(U1, \text{offers}, C1)$,  
$(U2, \text{offers}, C2)$,  
$(S1A1, \text{author}_of, Pu1)$,  
$(P1, \text{advises}, S1A1)$,  
$(P1, \text{speaksAt}, Co1)$,  
$(Co1, \text{title}, \text{"Int’l Semantic Web Conf."})$,  
$(P1, \text{pcmember}, Ct1)$,  
$(Ct1, \text{trackof}, Co2)$,  
$(S1A1, \text{author}_of, Pu2)$,  
$(S1A1, \text{enrolled}_in, C1)$,  

$\mu_2(C2, \text{course}_\text{title}, \text{"Data mining"})$,  
$(C3, \text{taught}_by, P1)$,  
$(S1A1, \text{enrolled}_in, C3)$,

$\mu_3(C3, \text{course}_\text{title}, \text{"Databases"})$.
A mapping \( \mu \) is a partial function from term variables to RDF Terms T

**Example triple pattern - t**

\( (?X, \text{course}_\text{title}, ?Y) \)

**Evaluation of t**

<table>
<thead>
<tr>
<th>?X</th>
<th>?Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Semantic Web</td>
</tr>
<tr>
<td>C2</td>
<td>Semantic Web and Databases</td>
</tr>
<tr>
<td>C3</td>
<td>Databases</td>
</tr>
</tbody>
</table>

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

**Example database**

\[ \mu_1(C1, \text{course}_\text{title}, \text{“Semantic Web”}), \]
\[ (U1, \text{offers}, C1), \]
\[ (U2, \text{offers}, C2), \]
\[ (S1A1, \text{author}_\text{of}, Pu1), \]
\[ (P1, \text{advises}, S1A1), \]
\[ (P1, \text{speaksAt}, Co1), \]
\[ (Co1, \text{title}, \text{“Int’l Semantic Web Conf.”}), \]
\[ (P1, \text{pcmember}, Ct1), \]
\[ (Ct1, \text{trackof}, Co2), \]
\[ (S1A1, \text{author}_\text{of}, Pu2), \]
\[ (S1A1, \text{enrolled}_\text{in}, C1), \]
\[ \mu_2(C2, \text{course}_\text{title}, \text{“Data mining”}), \]
\[ \mu_3(C3, \text{course}_\text{title}, \text{“Databases”}), \]
\[ (C3, \text{taught}_\text{by}, P1), \]
\[ (S1A1, \text{enrolled}_\text{in}, C3), \]

A evaluation of \( t \) is the set \([[]]\) of mappings that cause \( t \) to match the graph.

Extended semantics for SPARQ2L

A pmapping $\omega$ is a partial function $\omega$ from $VT \cup VP$ to $2^T$ such that for:
$x \in VT$, $\omega(x)$ maps to a singleton set of $2^T$.

Example tp-pattern $tp$
(S1A1, ??P, P1)

Evaluation of $tp$

<table>
<thead>
<tr>
<th>??P</th>
</tr>
</thead>
<tbody>
<tr>
<td>{S1A1, advisor, P1}</td>
</tr>
<tr>
<td>{S1A1, enrolled_in, C3, taught_by, P1}</td>
</tr>
</tbody>
</table>

The evaluation of $tp$ is the set of pmappings that cause $tp$ to “match” the graph

An example database
(C1, course_title, “Semantic Web”),
(U1, offers, C1),
(S1A1, author_of, Pu1),
(S1A1, advisor, P1),
(S1A1, author_of, Pu2),
(S1A1, enrolled_in, C3),
(C3, taught_by, P1),
(C3, course_title, “Databases”),
(U2, offers, C2),
(C2, course_title, “Semantic Web and Databases”)
Compatible\(^1\) pmappings: Mappings that agree on their shared variables.

\[
\begin{array}{c|c|c}
\text{?X} & \text{?Y} & \text{?U} \\
\hline
C1 & \text{Semantic Web} & \text{Bk1} \\
\hline
C3 & \text{Database} & \text{Bk1} \\
\end{array}
\]
Compatible\(^1\) pmappings: Mappings that agree on their shared variables.

Compatible pmappings can be merged

\[ \omega_2 \cup \omega_3 \]

<table>
<thead>
<tr>
<th>(\exists X)</th>
<th>(\exists Y)</th>
<th>(\exists U)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Semantic Web</td>
<td>Bk1</td>
</tr>
<tr>
<td>C3</td>
<td>Database</td>
<td>Bk1</td>
</tr>
<tr>
<td>C1</td>
<td>Semantic Web</td>
<td>Bk1</td>
</tr>
<tr>
<td>C3</td>
<td>Database</td>
<td>Bk1</td>
</tr>
</tbody>
</table>
Compatible\(^1\) pmappings: Mappings that agree on their shared variables.

Compatible pmappings can be merged

\[ \omega_2 \cup \omega_3 \]

Given sets of mappings \( M_1, M_2 \)
\( M_1 \triangleright \triangleleft M_2 \): extends the mappings of \( M_1 \) with compatible mappings of \( M_2 \)
Compatible\(^1\) pmappings: Mappings that agree on their shared variables.

Compatible pmappings can be merged

\[ \omega_2 \cup \omega_3 \]

Given sets of mappings \(M_1, M_2\)
\(M_1 \mid\rangle\langle M_2\): extends the mappings of \(M_1\) with compatible mappings of \(M_2\)

Let \(M_1\) and \(M_2\) be the evaluations of \(PP_1\) and \(PP_2\) resp. Then evaluation of \((PP_1 \text{ AND } PP_2)\) is \(M_1 \mid\rangle\langle M_2\)
Semantics of PATHFILTER

A pmapping \( _{\pi} \) satisfies \( |= \) the conditions:

- \( F \) is \( \text{containsAny}(\pi, L') \), if \( L' \neq _{\pi} \).
- \( F \) is \( \text{containsAll}(\pi, L') \), if \( L' \subseteq _{\pi} \).
- \( F \) is \( \text{containsPattern}(\pi, tr) \) if \( \text{ground}(tr) \) is a subpath of \( _{\pi} \).
- \( F \) is \( \text{isSimple}(\pi) \) if \( x, y \in _{\pi}, x \neq y \).
- \( F \) is \( \neg F_1 \), \( _{\pi} \not\models F_1 \)
- \( F \) is \( F_1 \land F_2 \), \( _{\pi} \models F_1 \) and \( _{\pi} \models F_2 \)
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
Foundations of evaluation framework\textsuperscript{2}

Given a directed graph $G = (V, E)$
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

\((u, p_1, w) \cup (u, p_2, w) \circ (w, p_3, v)\) is an \(p\)-expression of type \((u, v)\).

The Path Sequence for a graph \(G\) is the sequence

\((P_1, s_1, d_1), (P_2, s_2, d_2), (P_3, s_3, d_3), ..., (P_f, s_f, d_f), ..., (P_g, s_g, d_g), ..., (P_l, s_l, d_l)\)

\[p = p_1, p_2, ..., p_k\]

for any non-empty path \(p\) in \(G\).
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 
\((u, p_1, w) \cup (u, p_2, w) \cdot (w, p_3, v)\) is an \(p\)-expression of type \((u, v)\).

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

\[ P_{ij} < 2 < f < g < l \]

\(p = p_1, p_2, \ldots, p_k\)

for any non-empty path \(p\) in \(G\).
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)
Path Sequence PS for G

1: (c, 1, 2)  2: (d, 1, 3)  3: (a \cdot c^*, 2, 2)
4: (a \cdot d, 2, 3)  5: (e \cup (a \cdot c)^* \cdot 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 \geq v:
paths from u to v with no intermediate vertex w > v.
Path Sequence $PS$ for $G$

1: \((c, 1, 2)\)  
2: \((d, 1, 3)\)  
3: \((a \cdot c)^*, 2, 2)\)  
4: \((a \cdot d, 2, 3)\)  
5: \((e \cup (a \cdot c)^* \cdot e, 2, 5)\)  
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 $w < u$.

$u \geq v:$
paths from $u$ to $v$ with no intermediate vertex $w > v$.

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

\((2, 2) = \lambda\)
Solving \((2, 6)\)

\[(2, 2) = \lambda\]

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

\[(2, 2) \cdot (1, 3) = (2, 3) = \emptyset\]
Solving (2, 6)

(2, 2) = \lambda

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

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

(2, 2) \cup (2, 2) = (2, 2) = \emptyset \cup (a \cdot c)^*
\begin{itemize}
  \item 1: \((c, 1, 2)\)
  \item 2: \((d, 1, 3)\)
  \item 3: \((a \cdot c)^*, 2, 2)\)
  \item 4: \((a \cdot d, 2, 3)\)
  \item 5: \((e \cup (a \cdot c)^* \cdot e, 2, 5)\)
  \item 6: \((f, 3, 4)\)
  \item 7: \((d, 3, 5)\)
  \item 8: \((h, 3, 6)\)
  \item 9: \((g, 5, 4)\)
  \item 10: \((a, 2, 1)\)
\end{itemize}

**Solving \((2, 6)\)**

\begin{align*}
(2, 2) &= \lambda \\
(2, 2) \cdot (1, 2) &= (2, 2) = \emptyset \\
(2, 2) \cdot (1, 3) &= (2, 3) = \emptyset \\
(2, 2) \cup (2, 2) &= (2, 2) = \emptyset \cup (a \cdot c)^* \\
(2, 3) &= (2, 3) \cup (2, 2) \cdot (2, 3) = \emptyset \cup ((a \cdot c)^* \cdot a \cdot d) \\
(2, 6) &= (2, 3) \cup (2, 2) \cdot (3, 6) = \emptyset \cup (a \cdot c)^* \cdot a \cdot d \cdot h \\
\end{align*}
Solving (2, 6)

(2, 2) = \lambda

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

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

(2, 2) \cup (2, 2) = (2, 2) = \emptyset \cup (a \cdot c)^*
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
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 $G$. 
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 \( pe \).
Labeling a path sequence
Nodes of a strong component are prunable equivalent e.g. 8, 9 and 10
Labeling a path sequence

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.

OST Only those edges in the graph that lie on the *longest* path from root to a node.
Interval of tree subgraph identifiers sids disjoint from that of non-tree sids
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)\_i, (s, l, t)\_i) \diamond \text{PS}_i]\]
- for scc\(_x\), scc\(_y\) connected by edges e\(_1\), e\(_2\), .. e\(_k\), \[\(((s, l, t)\_x, (s, l, t))\_y \diamond \{\text{pe}_{e_1}, \text{pe}_{e_2}, \ldots \text{pe}_{e_k}\}]\]
- 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 – $O \Sigma n_i^3$
Properties of a 2-Color code

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

**NonReachability Property**: for $(s_u, l_u, t)_u, (s_v, l_v, t)_v$
- $s_u \neq s_v \iff$ result is empty.
- $l_u \geq l_v \iff$ 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

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

<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>
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)

Encoding table

<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>

<table>
<thead>
<tr>
<th>Code</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>001011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>011011</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>001111</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>011111</td>
<td></td>
<td></td>
<td></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**
Semantic Searching of a Different Kind

Query Specification

Select Search Mode: 50%

Conventional Discovery

Enter the two resources

SWEET_198959 SWEET_307295

Relevance Specification (separated with spaces)

Search Reset Close Previous Next
SSARK

Semantic Searching of a Different Kind

Query Specification

Select Search Mode: 50%

Conventional

Discovery

Enter the two resources

SWEET_198959
SWEET_307295

Relevance Specification (separated with spaces)

Search  Reset  Close  Previous  Next

adjustable search mode
adjustable search mode
High Information Gain
High Refraction Count
High S-Match

Low Information Gain
Low Refraction Count
High S-Match

SSARK
Semantic Searching of a Different Kind

Select Search Mode: 50 %
Conventional
Discovery

Enter the two resources

SWEET_198959
SWEET_307295

Relevance Specification (separated with spaces)

Search  Reset  Close  Previous  Next

adjustable search mode
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

 général 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


- Angela Maduko, **Kemafor Anyanwu**, Amit Sheth, Estimating the Cardinality of RDF Graph Patterns. *WWW2007 poster paper*
Publications contd.

**Application**


**Workflow management**


Thank you!!