Explaining Graph Navigational Queries

Valeria Fionda\textsuperscript{1}, Giuseppe Pirrò\textsuperscript{2}

\textsuperscript{1} DeMaCS, University of Calabria, Italy\textsuperscript{1}
\textsuperscript{2} Institute for High Performance Computing and Networking, ICAR-CNR, Italy

ESWC 2017
Problem: Explaining Graph Navigational Queries

**In:** A navigational query $Q$ to be evaluated on a graph $G$

**Out:** A graph interlinking pair of entities that belong to the results of $Q$
Related Work

**Relatedness Explanations**
- Connectivity Graphs
- REX [4]
- RelFinder [3], Explass [2], RECAP[1]


**Navigational Languages**
- Property Paths
- Extended Property Paths
- Nested Regular Expressions

**Relatedness Explanations**: given a pair, explain their relatedness

**Navigational Languages**: find pairs of entities given a specification
Related Work

**Relatedness Explanations**
- Connectivity Graphs
- REX [4]
- RelFinder [3], Explass [2], RECAP[1]

**Navigational Languages**
- Property Paths
- Extended Property Paths
- Nested Regular Expressions


**Relatedness Explanations:** given a pair, explain their relatedness
**Navigational Languages:** find pairs of entities given a specification

Our goal is to explain the results of navigational queries
Table: Evaluation Semantics of Nested Regular Expressions.

\[
\begin{align*}
[\epsilon]_G &= \{(u, u) \mid u \in G\} \\
[a]_G &= \{(u, v) \mid (u, a, v) \in G\} \\
[^{\wedge}]_G &= \{(u, v) \mid (v, a, u) \in G\} \\
[e_1/e_2]_G &= [e_1]_G \circ [e_2]_G \\
[e_1|e_2]_G &= [e_1]_G \cup [e_2]_G \\
[e^*]_G &= [\epsilon]_G \cup [e]_G \cup [e \cdot e]_G \cup [e \cdot e \cdot e]_G \cup \ldots \\
[[e]]_G &= \{(u, u) \mid \exists v \ s.t. (u, v) \in [e]_G\}
\end{align*}
\]

The symbol \(\circ\) denotes the composition of binary relations, \(\cup\) union, and \([\ ]\) handles nesting defined as an existential test.
Table: Evaluation Semantics of Nested Regular Expressions.

\[
[\epsilon]^G = \{(u,u) \mid u \in G\}
\]
\[
[a]^G = \{(u,v) \mid (u,a,v) \in G\}
\]
\[
[^\wedge]^G = \{(u,v) \mid (v,a,u) \in G\}
\]
\[
[e_1/e_2]^G = [e_1]^G \circ [e_2]^G
\]
\[
[e_1\mid e_2]^G = [e_1]^G \cup [e_2]^G
\]
\[
[e^\ast]^G = [\epsilon]^G \cup [e]^G \cup [e \cdot e]^G \cup [e \cdot e \cdot e]^G \cup \ldots
\]
\[
[[e]]^G = \{(u,u) \mid \exists v \text{ s.t.} (u,v) \in [e]^G\}
\]

The symbol $\circ$ denotes the composition of binary relations, $\cup$ union, and $[\ ]$ handles nesting defined as an existential test.

- The evaluation semantics tells us that the answer to a query $Q$ is a pair of nodes in the graph $G$ connected by a path that spells the language defined by the query $Q$
- SPARQL Property Paths, Nested Regular Expressions, Extended Property Paths work under this assumption
The symbol $\circ$ denotes the composition of binary relations, $\cup$ union, and $[\ ]$ handles nesting defined as an existential test.

- The evaluation semantics tells us that the answer to a query $Q$ is a *pair of nodes* in the graph $G$ connected by a path that spells the language defined by the query $Q$.
- SPARQL Property Paths, Nested Regular Expressions, Extended Property Paths work under this assumption.

**Connectivity information is lost**
Co-authors between 2002 and 2015.

S. Staab’s co-authors 2002-2015

- (S. Staab, C. d’Amato)
- (S. Staab, W. Nejdl)
- (S. Staab, R. Dividino)
- (S. Staab, Y. Sure)
- (S. Staab, P. Mika)
- (S. Staab, R. Troncy)
- (S. Staab, M. Thimm)

C. Gutierrez’s co-authors 2002-2015

- (C. Gutierrez, M. Arenas)
- (C. Gutierrez, J. Pérez)
- (C. Gutierrez, G. Pirró)
- (C. Gutierrez, V. Fionda)
- (C. Gutierrez, R. Angles)
- (C. Gutierrez, R. Rosati)
- (C. Gutierrez, E. Franconi)

(a) (b)
To Explanation Graphs

Example

Co-author network between 2002 and 2015.

Some pairs of results

(S. Staab, C. d'Amato)
(S. Staab, W. Nejdl)
(S. Staab, R. Dividino)
(S. Staab, Y. Sure)
(S. Staab, P. Mika)
(S. Staab, R. Troncy)
(S. Staab, M. S. Pini)
(S. Staab, M. Thimm)
(S. Staab, M. Ehrig)
(S. Staab, O. Gorlitz)

(S. Staab, S. Spirou)
(S. Staab, P. Mika)
(S. Staab, R. Troncy)
(S. Staab, Y. Sure)
(S. Staab, M. Thimm)
(S. Staab, M. Ehrig)
(S. Staab, O. Gorlitz)

C. Gutierrez's
ISWC co-authors 2002-2015

Some pairs of results

(C. Gutierrez, M. Arenas)
(C. Gutierrez, J. Pérez)
(C. Gutierrez, G. Pirró)
(C. Gutierrez, V. Fionda)
(C. Gutierrez, R. Angles)
(C. Gutierrez, R. Rosati)
(C. Gutierrez, E. Franconi)

……
Contributions

Main contributions

- A language called GeL, which can provide query explanations
- Novel query evaluation semantics that capture the structure of the graph $G$ needed to explain the result of a query $Q$
- Efficient automata-based evaluation algorithms
- A GeL2CONSTRUCT translation
- A visual tool
GeL’s Syntax

\[
gexp := \tau \# \text{exp} \quad (\tau \in \{\text{FULL}, \text{FILTERED}, \text{set}\})
\]

\[
\text{exp} := a \ \text{gtest} (a \in \Sigma) \mid ^\wedge a \ \text{gtest} (a \in \Sigma) \mid \text{exp} / \text{exp} \mid \text{exp} | \text{exp} \mid \text{exp}^* \mid \text{exp}\{l, h\}
\]

\[
\text{gtest} := \text{gtest} && \text{gtest} \mid \text{gtest} || \text{gtest} \mid (\text{gtest}) \mid [\text{exp}] \mid \{\text{op val}\}
\]

\[
\text{op} := > | < | = | \neq
\]

- $\tau$ controls the type of output; one keeping the whole portion of the graph “touched” during the evaluation (FULL) and the other keeping only paths leading to results (FILTERED).

- $^\wedge$ denotes backward navigation, / path concatenation, | path union, \{l,h\} denotes repetition of an exp between l and h times; && and || conjunction and disjunction of gtest, respectively.
\[ e = \text{associatedBand} / \text{genre} \]
Query Semantics

Full semantics

\[ e = \text{associatedBand} / \text{genre} \]
Filtered semantics

\[ e = \text{associatedBand} / \text{genre} \]
(Explanation Graph). Given a graph $G$, a GeL expression $e$ and a set of starting nodes $S \subseteq \text{nodes}(G)$, an EG is a quadruple $\Gamma = (V, E, S, T)$ where $V \subseteq \text{nodes}(G)$, $E \subseteq \text{triples}(G)$ and $T \subseteq V$ is a set of ending nodes, that is, nodes reachable from nodes in $S$ via paths satisfying $e$. 
(Explanation Graph). Given a graph $G$, a GeL expression $e$ and a set of starting nodes $S \subseteq \text{nodes}(G)$, an EG is a quadruple $\Gamma = (V, E, S, T)$ where $V \subseteq \text{nodes}(G)$, $E \subseteq \text{triples}(G)$ and $T \subseteq V$ is a set of ending nodes, that is, nodes reachable from nodes in $S$ via paths satisfying $e$.

$$e = (:\text{knows}:/:\text{knows}) | (:\text{co-author}:/:\text{co-author}).$$

$$\Gamma = (\text{nodes}(G), \text{edges}(G), \{a, b\}, \{c, d\})$$
Definition

**G-Soundness**. An EG $\Gamma$ is G-sound if, and only if, each ending node is reachable in $\Gamma$ from each starting node via a path satisfying $e$.

Definition

**G-Completeness**. An EG $\Gamma$ is G-complete if, and only if, all nodes reachable from some starting nodes, via a path satisfying $e$, are in the ending nodes.
Query Semantics

**Definition**

**(G-Soundness).** An EG $\Gamma$ is G-sound if, and only if, each ending node is reachable in $\Gamma$ from each starting node via a path satisfying $e$.

**Definition**

**(G-Completeness).** An EG $\Gamma$ is G-complete if, and only if, all nodes reachable from some starting nodes, via a path satisfying $e$, are in the ending nodes.

\[ e = (\text{:knows}/\text{:knows}) | (\text{:co-author}/\text{:co-author}) \]

\[ \Gamma = (\text{nodes}(G), \text{edges}(G), \{a, b\}, \{c, d\}) \]

⇒ violates G-soundness $a$ and $d$ are not connected by a path satisfying $e$. 
An Example of G-soundness

**Lemma**

*(G-Sound and G-Complete EGs).* Explanation Graphs having a single starting node \( v \in \text{nodes}(G) \) are G-sound and G-complete.

**Definition**

*(Query Explanation).* Given a GeL expression \( e \) and a graph \( G \), a query explanation \( E^Q \) is a set of G-sound and G-complete EGs \( \Gamma_v \).
An Example of G-soundness

Lemma

(G-Sound and G-Complete EGs). Explanation Graphs having a single starting node $v \in \text{nodes}(G)$ are G-sound and G-complete.

Definition

(Query Explanation). Given a GeL expression $e$ and a graph $G$, a query explanation $\mathcal{E}^Q$ is a set of G-sound and G-complete EGs $\Gamma_v$.

\[ e = (:\text{knows}/:\text{knows}) | (:\text{co-author}/:\text{co-author}). \]

\[ a \quad :\text{knows} \quad b \quad :\text{co-author} :\text{co-author} \]

\[ f \rightarrow \quad c \quad \quad d \]

\[ a \quad :\text{knows} \quad f \quad :\text{knows} \quad c \]

\[ b \quad :\text{co-author} :\text{co-author} \quad d \]
Operators over Graphs
Extend operators for binary relations to work with graphs

**Definition** (EGs operators). Let \( \Gamma_i = (V_i, E_i, s_i, T_i) \), \( i = 1, 2 \) be EGs and \( \Gamma_\bot = (\emptyset, \emptyset, \bot, \emptyset) \) denote the empty EG, where \( \bot \) is a symbol not in the universe of nodes.

Composition (\( \circ \)) and union (\( \cup \)) of EGs:
\[
\Gamma_1 \circ \Gamma_2 = \begin{cases} 
\Gamma_\bot & \text{if } s_2 \notin T_1, \\
(V_1 \cup V_2, E_1 \cup E_2, s_1, T_2) & \text{if } s_2 \in T_1.
\end{cases}
\]

V. Fionda, G. Pirrò
Explaining Graph Navigational Queries
ESWC 2017 15 / 26
Operators over Graphs
Extend operators for binary relations to work with graphs

Definition (EGs operators). Let $\Gamma_i = (V_i, E_i, s_i, T_i)$, $i = 1, 2$ be EGs and $\Gamma_{\perp} = (\emptyset, \emptyset, \perp, \emptyset)$ denote the empty EG, where $\perp$ is a symbol not in the universe of nodes.

Composition ($\circ$) and union ($\cup$) of EGs:

$$\Gamma_1 \cup \Gamma_2 = \begin{cases} (V_1 \cup V_2, E_1 \cup E_2, s_1, T_1 \cup T_2) & \text{if } s_1 = s_2, \\ \Gamma_1 & \text{if } s_1 \neq s_2 \land \Gamma_2 = \Gamma_{\perp}, \\ \Gamma_2 & \text{if } s_1 \neq s_2 \land \Gamma_1 = \Gamma_{\perp}, \\ \text{not defined} & \text{if } s_1 \neq s_2 \land \Gamma_1, \Gamma_2 \neq \Gamma_{\perp}. \end{cases}$$
Operators over Graphs
Extend operators for binary relations to work with graphs

Definition (Operations over sets of EGs). Let $S_1$ and $S_2$ be two sets of EGs.

1. For each $\text{op} \in \{\circ, \cup\}$ we define $S_1 \text{ op } S_2 = \{ \Gamma_1 \text{ op } \Gamma_2 \mid \Gamma_1 \in S_1, \Gamma_2 \in S_2 \}$.
2. (Disjoint union, direct sum) : $S_1 \oplus S_2 = \{ \Gamma \mid \Gamma \in S_1 \lor \Gamma \in S_2 \}$. 
The evaluation produces a set of $\Gamma$ (explanation graphs); one for each starting node.

Each syntactic construct has a counterpart in the semantics.

The semantics is defined recursively.
### Semantics - an overview

<table>
<thead>
<tr>
<th>Expression</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi[\text{a gtest}]_v^G$</td>
<td>$\Phi[\text{a gtest}]_v^G = (v, a, v') \in G \mid \Phi[\text{gtest}]_v^G \cup (v, v') \in \Phi[\text{gtest}]_v^G$</td>
</tr>
<tr>
<td>$\Phi[\neg \text{a gtest}]_v^G$</td>
<td>$\Phi[\neg \text{a gtest}]_v^G = (v', a, v) \in G \mid \Phi[\text{gtest}]_v^G \cup (v, v') \in \Phi[\text{gtest}]_v^G$</td>
</tr>
<tr>
<td>$\Phi[\text{exp}_1 / \text{exp}_2]_v^G$</td>
<td>$\Phi[\text{exp}_1 / \text{exp}_2]_v^G = \Phi[\text{exp}_1]_v^G \circ \left( \Phi[\text{exp}_2]_v^G \cup (v', \Gamma) \in G \mid \Phi[\text{exp}_2]_v^G \cup (v, v') \in \Phi[\text{exp}_2]_v^G \right)$</td>
</tr>
<tr>
<td>$\Phi[\text{exp}_1</td>
<td>\text{exp}_2]_v^G$</td>
</tr>
<tr>
<td>$\Phi[[\text{exp}]]_v^G$</td>
<td>${v}, \emptyset, v, {v} \cup \Gamma$ if $\exists \Gamma \in \Phi[[\text{exp}]]_v^G \land \Gamma.\Gamma \neq \emptyset$ otherwise</td>
</tr>
<tr>
<td>$\Phi[\text{exp}_1 \lor \text{exp}_2]_v^G$</td>
<td>${v}, \emptyset, v, {v} \cup \Gamma$ if $\exists \Gamma \in (A</td>
</tr>
<tr>
<td>$\Phi[\text{exp}_1 \land \text{exp}_2]_v^G$</td>
<td>${v}, \emptyset, v, {v}$ if $\text{Evaluate}(v, \text{op}, \text{val}) = \text{true}$ otherwise</td>
</tr>
<tr>
<td>$\Phi[\text{gtest}_1 &amp;&amp; \text{gtest}_2]_v^G$</td>
<td>${v}, \emptyset, v, {v}$ if $(A</td>
</tr>
<tr>
<td>$\Phi[\text{gtest}_1 | \text{gtest}_2]_v^G$</td>
<td>${v}, \emptyset, v, {v}$ if $(A</td>
</tr>
</tbody>
</table>
Evaluation algorithm

- Automata-based
- Works in two steps:
  1. Building the product automata - cost: $O(|G| \times |e|)$;
  2. Marking of the product automata (only for the filtered semantics) -
     cost: $O(|G| \times |e|)$
- Total cost: $O(|\text{nodes}(G)| \times |G| \times |e|)$
### Translation into SPARQL

<table>
<thead>
<tr>
<th>Construct</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta^c(\text{root})$</td>
<td><code>CONSTRUCT{\'\Theta^c(\text{root}.child(1))\'} WHERE {\'\Theta^w(\text{root}.child(1))\'}</code></td>
</tr>
<tr>
<td>$\Gamma(n)$</td>
<td><code>n.s n.p n.o</code></td>
</tr>
<tr>
<td>$\Theta^c(n/n)$</td>
<td>$\Theta^c(n.child(1))$</td>
</tr>
<tr>
<td>$\Theta^c(n[n^u/u]gtest)$</td>
<td>$\Theta^c(n.child(1))$ $\Theta^c(n.child(2))$</td>
</tr>
<tr>
<td>$\Theta^w(n/n)$</td>
<td><code>{\'\Theta^w(n.child(1))\'} UNION{\'\Theta^w(n.child(2))\'}</code></td>
</tr>
<tr>
<td>$\Theta^w(n[n])$</td>
<td>$\Gamma(n)$ <code>FILTER(\'n.p=\'u\'). $\Theta^t(n.child(1))$</code></td>
</tr>
<tr>
<td>$\Theta^t(n[n&amp;&amp;])$</td>
<td>$\Theta^t(n.child(1))$ $\Theta^t(n.child(2))$</td>
</tr>
<tr>
<td>$\Theta^t(n[n|])$</td>
<td><code>{\'\Theta^t(n.child(1))\'} UNION{\'\Theta^t(n.child(2))\'}</code></td>
</tr>
<tr>
<td>$\Theta^t(n[nexp])$</td>
<td><code>FILTER EXISTS{\Theta^w(n.child(1))\'}</code></td>
</tr>
<tr>
<td>$\Theta^t(n[\text{op val}])$</td>
<td><code>FILTER(\'n.o op val\')</code></td>
</tr>
</tbody>
</table>

- To make explanations available in existing SPARQL processors
- Each GeL expressions has associated a chunk of SPARQL code
- The translation occurs by traversing the expression’s parse tree and recursively applying the rules reported in the table
## Translation into SPARQL

<table>
<thead>
<tr>
<th>Construct</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Theta^e(\text{root}))</td>
<td>‘CONSTRUCT {’(\Theta^e(\text{root.child(1)}))‘} WHERE {’(\Theta^w(\text{root.child(1)}))‘}’</td>
</tr>
<tr>
<td>(\Gamma(n))</td>
<td>n.s n.p n.o‘.‘</td>
</tr>
<tr>
<td>(\Theta^c(n/</td>
<td>\mid</td>
</tr>
<tr>
<td>(\Theta^c(n(u\mid</td>
<td>u)g_{test}))</td>
</tr>
<tr>
<td>(\Theta^w(n/</td>
<td>))</td>
</tr>
<tr>
<td>(\Theta^w(n</td>
<td>))</td>
</tr>
<tr>
<td>(\Theta^w(n(u\mid</td>
<td>u)g_{test}))</td>
</tr>
<tr>
<td>(\Theta^t(n&amp;&amp;&amp;))</td>
<td>(\Theta^t(n.child(1)) \Theta^t(n.child(2)))</td>
</tr>
<tr>
<td>(\Theta^t(n||))</td>
<td>‘{’(\Theta^t(n.child(1)))‘} UNION {’(\Theta^t(n.child(2)))‘}’</td>
</tr>
<tr>
<td>(\Theta^t(n[^{\text{nexp}}]})</td>
<td>‘FILTER EXISTS{’(\Theta^w(n.child(1)))‘}’</td>
</tr>
<tr>
<td>(\Theta^t(n{\text{op \ val}}))</td>
<td>‘FILTER(’n.o \text{ op val}‘)’</td>
</tr>
</tbody>
</table>

- **Filtered semantics only**
- **Explanations are only G-complete:** it is not possible to keep separate the explanation for each node in the result of a CONSTRUCT while it can be done in GeL by using Explanation Graphs
**Goal**: measure the overhead of outputting graphs as a result of navigational queries and build query explanations.
**Goal:** measure the overhead of outputting graphs as a result of navigational queries and build query explanations.

- FOAF network (4M triples) obtained by crawling from 10 different seeds foaf:knows predicates up to distance 6 and then merging the graphs.
- The overhead of generating explanations about Tim Berners-Lee (TBL) along with the size of the explanation (\#nodes, \#edges) generated under the filtered and full semantics.
- Similar behavior when considering explanations related to other people in the FOAF network (e.g., A. Polleres, N. Lopes)

<table>
<thead>
<tr>
<th>FOAF</th>
<th>filtered</th>
<th>full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>0.434</td>
<td>0.278</td>
</tr>
<tr>
<td>Q2</td>
<td>0.738</td>
<td>0.234</td>
</tr>
<tr>
<td>Q3</td>
<td>0.985</td>
<td>0.534</td>
</tr>
<tr>
<td>Q4</td>
<td>1.155</td>
<td>0.849</td>
</tr>
<tr>
<td>Q5</td>
<td>1.665</td>
<td>1.145</td>
</tr>
<tr>
<td>Q6</td>
<td>1.785</td>
<td>1.257</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FOAF</th>
<th>filtered</th>
<th>full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>(6,5)</td>
<td>(17,17)</td>
</tr>
<tr>
<td>Q2</td>
<td>(18,37)</td>
<td>(20,45)</td>
</tr>
<tr>
<td>Q3</td>
<td>(18,44)</td>
<td>(25,53)</td>
</tr>
<tr>
<td>Q4</td>
<td>(23,51)</td>
<td>(55,90)</td>
</tr>
<tr>
<td>Q5</td>
<td>(36,64)</td>
<td>(149,236)</td>
</tr>
<tr>
<td>Q6</td>
<td>(177,111)</td>
<td>(190,139)</td>
</tr>
</tbody>
</table>

**Overhead (secs).**

**Size of the explanation.**
**Goal**: measure the overhead on SPARQL processors.
Goal: measure the overhead on SPARQL processors.

---

**DBpedia time (ms)**

- SELECT
- CONSTRUCT

**DBpedia size**

- Pairs (SELECT)
- Triples (CONSTRUCT)

**Yago time (ms)**

- SELECT
- CONSTRUCT

**Yago size**

- Pairs (SELECT)
- Triples (CONSTRUCT)
Concluding Remarks and Future Work

Summary
- Enhancing navigational languages with explanation capabilities
- Novel query evaluation semantics
- Efficient Algorithms
Summary
- Enhancing navigational languages with explanation capabilities
- Novel query evaluation semantics
- Efficient Algorithms

Future Work
- Negative information
- Why queries
Thanks