The case for Regular Path Queries

Tobias Lindaaker @ neo4j
CIR-2017-179
Why should we add Path Queries to Cypher?

- Find complex connections
  - Repetitions of patterns and not just single edge label
  - Alternatives between patterns and not just single edge label

- Express patterns as patterns instead of combinations of queries
  (Using \texttt{UNION} to express disjunctions hides what is actually going on)
Path Pattern Queries

*RPQs in Cypher*

Tobias Lindaaker @ neo4j

CIP 2017-02-06
Path Patterns in Cypher

- **Predicates on Edge Label:** ()-/:FOO/-()
- **Predicates on Nodes:** ()-/:Alpha{beta:'gamma'}/-()
- **Alternatives:** ()-/:FOO | :BAR | :BAZ/-()
- **Sequence:** ()-/:FOO :BAR :BAZ/-()
- **Grouping:** ()-/:FOO | [:BAR :BAZ]/-()
- **Direction:** ()-/<:FOO :BAR <:BAZ>/->()
- **Any Edge:** ()-/-/-()
- **Repetition:** ()-/:FOO? :BAR+ :BAZ* :FOO*3.. :BAR*1..5/-()
- **Predicates on Edge Properties:** ()-/[ - {some:'value'}] /-()
  (applies to all edges matched by the group)
  (for more complex predicates use Named Path Patterns)
- **Negation:** ()-/[ ^:FOO :BAR] /-()
  (negation groups only allows edge labels - no other combinators or properties)
- **Named Path Patterns:** ()-/~foobar/-()
  after declaring: PATH PATTERN foobar = ()-/:FOO :BAR/-()
Example

() -/[^:FOO :BAR]/-()

() -/[^:FOO|:BAR]/-()

() -/~not_foo_or_bar/-()

PATH PATTERN not_foo_or_bar = ()-[x]-(
WHERE label(x) <> 'FOO' AND label(x) <> 'BAR'
Atomic Patterns vs Path Patterns

Edge Patterns are delimited by square brackets \[ \ldots \] and denotes a single edge
Edge Patterns can be used both for matching an edge and for creating an edge
Edge Patterns can bind a matched edge to a variable

Path Patterns are delimited by slashes / \ldots / and denotes a path of zero or more edges
Path Patterns can only be used for matching paths, and cannot be used for creating entities in a graph
Path Patterns can not bind individual edges (or nodes) to variables
Compared to Regular Expressions (over text)

Regular expression over text
- Atoms: text literals
- Disjunction: | (OR)
- Concatenation: juxtaposition
- Grouping: ( ... )
- Quantification:
  - ? - zero or one
  - * - zero or more
  - + - one or more
  - {n, m} - at least n, at most m
- Wildcard: . (any character)
- String start/end
- Disjunction of atoms short form: [ ... ]
- Negation of disjunction of atoms: [^...] (EXAMPLE: [^a-zA-Z0-9])
- Greedy, Lazy, Possessive

Cypher Path Patterns
- Atoms: Edge / Node (label) predicates
- Disjunction: | (OR)
- Concatenation: juxtaposition
- Grouping: [ ... ]
- Quantification:
  - ? - zero or one
  - * - zero or more
  - + - one or more
  - *n..m - at least n, at most m
- Wildcard: - (any edge)
- Not applicable
- Not available
- Not supported
- Not explored
Compared to GXPath *(path expressions)*

- $\square g^G = \{(v,v) \mid v \in V\}$ - **yes:** $v-/()/->v$
- $\square \square^G = \{(v,w) \mid (v,a,w) \in E \text{ for some } a\}$ - **yes:** $(v)-/-/->(w)$
- $\square a^G = \{(v,w) \mid (v,a,w) \in E\}$ - **yes:** $(v)-:/a/->w$
- $\square a^-^G = \{(v,w) \mid (w,a,v) \in E\}$ - **yes:** $(v)-/<:a/->w$
- $\square a^*^G = \text{ reflexive transitive closure of } a$
  - **yes:** $(v)-/[a^*]_->($
- $\square a \cdot \beta^G = \square a^G \square \beta^G$ - **yes:** $(v)-[/a \beta]->($
- $\square a \cup \beta^G = \square a^G \cup \square \beta^G$ - **yes:** $(v)-/[a \beta]->($
- $\neg a^G = \{v,w \mid (w,a,v) \in E\}$ - **no! the PATH PATTERN syntax is limited to connected patterns**
- $\square \emptyset^G = \{(v,v) \mid v \in \emptyset^G\}$ - **yes:** $v-/\emptyset^G->v$
- $\square a^n..m^G = \bigcup_{k=n}^m (\square a^G)^k$ - **yes:** $(v)-/[a^*n..m]->($
- $\square a^G = \{(v,w) \in \square a^G \mid \rho(v) = \rho(w)\}$
  - **yes:** PATH PATTERN $alpha_eq = (v)-[a]->(w)$ WHERE $v.\rho = w.\rho$
- $\square a^G = \{(v,w) \in \square a^G \mid \rho(v) \neq \rho(w)\}$
  - **yes:** PATH PATTERN $alpha_not_eq = (v)-[a]->(w)$ WHERE $v.\rho <> w.\rho$
- Conjunctions *for CRPQs*: $\square a \wedge \beta^G = \square a^G \cup \square \beta^G$
  - **yes:** PATH PATTERN $alpha_and_beta = (v)-[a]->(w)$ WHERE EXISTS $\{ (v)-[\beta]->(w) \}$
Cartesian product with negation

MATCH (a), (b)
WHERE NOT EXISTS {
  (a)-/.../- (b)
}
Compared to GXPath (node tests)

- **has alpha**
  \[ \alpha_G = \{ v \mid \exists w \ (v,w) \in \alpha_G \} \]
  - yes: PATH PATTERN has_alpha = (v) WHERE EXIST \{ (v)-[ \alpha ]->() \}

- **not phi**
  \[ \neg \phi_G = V - \phi_G \]
  - yes: PATH PATTERN not_phi = (v) WHERE NOT \phi(v)

- **phi and psi**
  \[ \phi \land \psi_G = \phi_G \cap \psi_G \]
  - yes: PATH PATTERN phi_and_psi = (v) WHERE \phi(v) AND \psi(v)

- **phi or psi**
  \[ \phi \lor \psi_G = \phi_G \cup \psi_G \]
  - yes: PATH PATTERN phi_or_psi = (v) WHERE \phi(v) OR \psi(v)

- **rho_is_c**
  \[ c_G = \{ v \in V \mid \rho(v) = c \} \]
  - yes: PATH PATTERN rho_is_c = (v) WHERE v.\rho = c

- **rho_is_not_c**
  \[ c^*_G = \{ v \in V \mid \rho(v) \neq c \} \]
  - yes: PATH PATTERN rho_is_not_c = (v) WHERE v.\rho <> c

- **alpha_eq_beta**
  \[ \alpha = \beta_G = \{ v \in V \mid \exists w,y \ (v,w) \in \alpha_G, (v,y) \in \beta_G, \rho(w)=\rho(y) \} \]
  - yes: PATH PATTERN alpha_eq_beta = (v) WHERE EXIST \{ MATCH (v)-[\alpha ]-(w), (v)-[\beta ]-(y) WHERE w.\rho = y.\rho \}

- **alpha_not_eq_beta**
  \[ \alpha \neq \beta_G = \{ v \in V \mid \exists w,y \ (v,w) \in \alpha_G, (v,y) \in \beta_G, \rho(w)\neq\rho(y) \} \]
  - yes: PATH PATTERN alpha_not_eq_beta = (v) WHERE EXIST \{ MATCH (v)-[\alpha ]-(w), (v)-[\beta ]-(y) WHERE w.\rho <> y.\rho \}
Paths where property value differs from first

PATH PATTERN not_first_foo = (first)-/-*/-() 
   WHERE all( n IN nodes(not_first_foo) 
       WHERE n = first OR n.foo <> first.foo )

MATCH x = (a)-/~not_first_foo/- (b) 
RETURN a, b, length(x)
Remaining work

- Incorporate negation into CIP
- Comparison with Conjunctive Context-Free Path Queries
- Are there things we can express in Named Path Patterns (WHERE) that cannot be implemented using REMs?
- Make sure that the referenced papers are linked from the CIP
- Idea: :- instead of - for (edge label) wildcard
- Binding groups (paths), a better syntax than:
  
  ```
  PATH PATTERN foo = ( ) - [x] - (b), p=(b) - / - * / - ()
  ```

  (at least a discussion section on it - either including or explaining why not)
  - Can we use bare words for that?
- Quite possibly juxtaposition will be used elsewhere for conjunction
  so perhaps we should allow that here as well, and use a separate operator for sequences
Further reading

- The Cypher Improvement Proposal: CIP 2017-02-06 (rendered text)
- Presentation from oCIM 1
- Short presentation on Path Patterns
- Presentation from oCIM 2