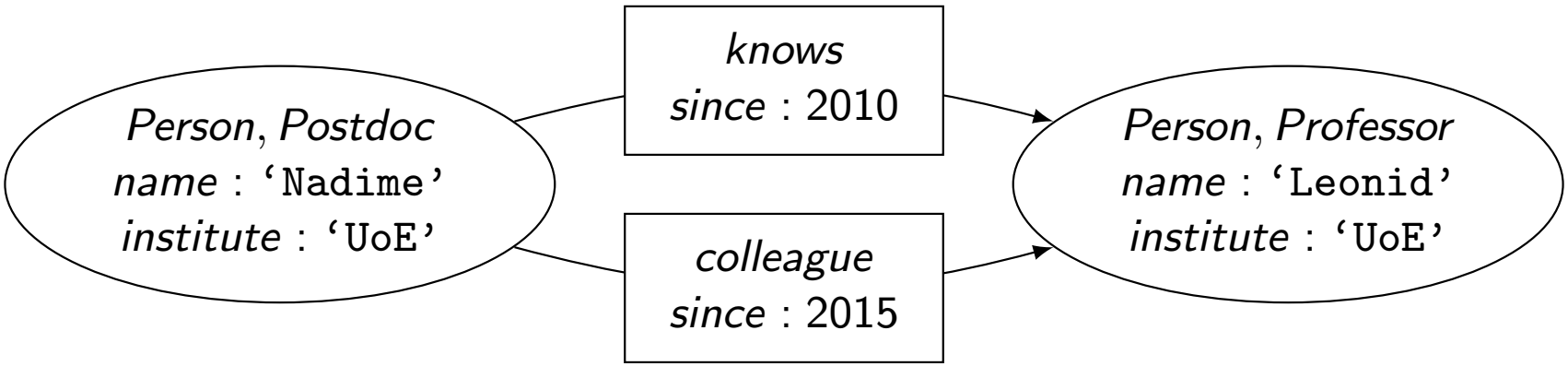
Formal Specification of Cypher

Nadime Francis

University of Edinburgh

Wednesday, May, 10th

Property Graphs



A property graph is a tuple $G = (N, R, s, t, \iota, \lambda, \tau)$, where:

- $N \subseteq \mathcal{N}$: finite set of nodes
- $R \subseteq \mathcal{R}$: finite set of relationships
- $s : R \to N$: maps each relationship to its source
- $t : R \to N$: maps each relationship to its target
- $\iota : (N \cup R) \times \mathcal{K} \to \mathcal{V}$: maps each x and k to $x.k$.
- $\lambda : N \to 2^{\mathcal{L}}$: associates a set of label to each node
- $\tau : R \to \mathcal{T}$: associates a type to each relationship

Records and Tables

A record is a tuple with named fields : $(a_1 : v_1, \ldots, a_n : v_n)$.
A table is a bag of uniform records.

Example: $\left\{ \begin{array}{l} (a : 1, b : 3), (a : \texttt{`oCIM 2'}, b : \texttt{`London'}), \\ (a : \texttt{`oCIM'}, b : \texttt{`Walldorf'}), (a : 1, b : 3) \end{array} \right\}$

a	b
1	3
`oCIM 2`	`London`
`oCIM`	`Walldorf`
1	3

$=$

a	b
`oCIM`	`Walldorf`
1	3
`oCIM 2`	`London`
1	3

Operations and Expressions

An Example

$$\texttt{MATCH}\ (n : \mathit{Person}) - [: \mathit{knows}] -> (m : \mathit{Person})$$
$$\texttt{WHERE}\ n.\mathit{institute} = m.\mathit{institute}$$
$$\texttt{RETURN}\ n.\mathit{name}, m.\mathit{name}, n.\mathit{institute}\ \texttt{AS}\ \mathit{institute}$$

Operations and Expressions

An Example

- $$\texttt{MATCH}\ (n : \mathit{Person}) - [: \mathit{knows}] -> (m : \mathit{Person})$$
  $$\texttt{WHERE}\ n.\mathit{institute} = m.\mathit{institute}$$
  $$\texttt{RETURN}\ n.\mathit{name}, m.\mathit{name}, n.\mathit{institute}\ \texttt{AS}\ \mathit{institute}$$

Operations and Expressions

An Example

- `MATCH` $(n : Person) - [: knows] -> (m : Person)$
  `WHERE` $n.institute = m.institute$
  `RETURN` $n.name, m.name, n.institute$ `AS` $institute$

n	m
$\{name : \text{`Nadime'}, institute : \text{`UoE'}\}$	$\{name : \text{`Leonid'}, institute : \text{`UoE'}\}$
$\{name : \text{`Paolo'}, institute : \text{`UoE'}\}$	$\{name : \text{`Nadime'}, institute : \text{`UoE'}\}$
$\{name : \text{`Nadime'}, institute : \text{`UoE'}\}$	$\{name : \text{`Stefan'}, institute : \text{`Neo'}\}$
$\{name : \text{`Alastair'}, institute : \text{`Neo'}\}$	$\{name : \text{`Stefan'}, institute : \text{`Neo'}\}$

Operations and Expressions

An Example

MATCH $(n : Person) - [: knows] -> (m : Person)$

▶ WHERE $n.institute = m.institute$

RETURN $n.name, m.name, n.institute$ AS $institute$

n	m
$\{name : \text{`Nadime'}, institute : \text{`UoE'}\}$	$\{name : \text{`Leonid'}, institute : \text{`UoE'}\}$
$\{name : \text{`Paolo'}, institute : \text{`UoE'}\}$	$\{name : \text{`Nadime'}, institute : \text{`UoE'}\}$
$\{name : \text{`Alastair'}, institute : \text{`Neo'}\}$	$\{name : \text{`Stefan'}, institute : \text{`Neo'}\}$

Operations and Expressions

An Example

```
MATCH (n : Person) − [: knows]−> (m : Person)
WHERE n.institute = m.institute
RETURN n.name, m.name, n.institute AS institute
```

n.name	*m.name*	*institute*
'Nadime'	'Leonid'	UoE
'Paolo'	'Nadime'	UoE
'Alastair'	'Stefan'	Neo

Operations and Expressions

$$Q = \boxed{\begin{array}{l} (\alpha)\ \texttt{MATCH}\ (n : \mathit{Person}) - [: \mathit{knows}] -> (m : \mathit{Person}) \\ (\beta)\ \texttt{WHERE}\ n.\mathit{institute} = m.\mathit{institute} \\ (\gamma)\ \texttt{RETURN}\ n.\mathit{name}, m.\mathit{name}, n.\mathit{institute}\ \texttt{AS}\ \mathit{institute} \end{array}}$$

Operations and Expressions

$$Q = \boxed{\begin{array}{l} (\alpha)\ \texttt{MATCH}\ (n : \mathit{Person}) - [: \mathit{knows}] -> (m : \mathit{Person}) \\ (\beta)\ \texttt{WHERE}\ n.\mathit{institute} = m.\mathit{institute} \\ (\gamma)\ \texttt{RETURN}\ n.\mathit{name}, m.\mathit{name}, n.\mathit{institute}\ \texttt{AS}\ \mathit{institute} \end{array}}$$

Operations

- $[\![op]\!]_G : \mathit{Tables} \to \mathit{Tables}$
- Semantics of a query by composition

Ex: $[\![Q]\!]_G = [\![\alpha]\!]_G \circ [\![\beta]\!]_G \circ [\![\gamma]\!]_G$

- Answers to Q on G: $[\![Q]\!]_G(\{\})$

Operations and Expressions

$$Q = \boxed{\begin{array}{l} (\alpha)\ \texttt{MATCH}\ (n : \mathit{Person}) - [: \mathit{knows}] -> (m : \mathit{Person}) \\ (\beta)\ \texttt{WHERE}\ n.\mathit{institute} = m.\mathit{institute} \\ (\gamma)\ \texttt{RETURN}\ n.\mathit{name}, m.\mathit{name}, n.\mathit{institute}\ \texttt{AS}\ \mathit{institute} \end{array}}$$

Operations

- $[\![op]\!]_G : \mathit{Tables} \to \mathit{Tables}$
- Semantics of a query by composition

Ex: $[\![Q]\!]_G = [\![\alpha]\!]_G \circ [\![\beta]\!]_G \circ [\![\gamma]\!]_G$

- Answers to Q on G: $[\![Q]\!]_G(\{\})$

Expressions

- $[\![exp]\!]_{G,u} \in \mathcal{V}$ where u is a record, giving binding to variables

Ex: $[\![\beta]\!]_G(T) = \left\{ u \in T \mid [\![n.\mathit{institute} = m.\mathit{institute}]\!]_{G,u} = \texttt{true} \right\}$

Pattern Matching

Rigid pattern satisfaction

- Rigid path pattern: no variable length edge patterns.

Ex: $(n : \mathit{Person}) - [: \mathit{knows} * 2] -> () - [: \mathit{likes}] -> (m : \mathit{Movie})$

- Unique way for a path p to satisfy a rigid pattern π wrt G, u. Notation: $(p, G, u) \models \pi$

Pattern Matching

Rigid pattern satisfaction

- Rigid path pattern: no variable length edge patterns.

Ex: $(n : Person) - [: knows * 2] -> () - [: likes] -> (m : Movie)$

- Unique way for a path p to satisfy a rigid pattern π wrt G, u. Notation: $(p, G, u) \models \pi$

Variable-length paths and free variables

- $\text{rigid}(\pi) = \{\pi' \mid \pi' \text{ is rigid and } \pi \sqsupset \pi'\}$

Ex: $() - [*2] -> () - [*4] -> () \sqsubset () - [*1..3] -> () - [*] -> ()$

- $\text{free}(\pi, u)$: all names that occur in π and not in u

Pattern Matching

Rigid pattern satisfaction

- Rigid path pattern: no variable length edge patterns.

Ex: $(n : \mathit{Person}) - [: \mathit{knows} * 2] -> () - [: \mathit{likes}] -> (m : \mathit{Movie})$

- Unique way for a path p to satisfy a rigid pattern π wrt G, u. Notation: $(p, G, u) \models \pi$

Variable-length paths and free variables

- $\text{rigid}(\pi) = \{\pi' \mid \pi' \text{ is rigid and } \pi \sqsupset \pi'\}$

Ex: $() - [*2] -> () - [*4] -> () \sqsubset () - [*1..3] -> () - [*] -> ()$

- $\text{free}(\pi, u)$: all names that occur in π and not in u

$$[\![\texttt{MATCH}\ \pi]\!]_G(T) = \bigcup_{\substack{\pi' \in \text{rigid}(\pi) \\ u \in T,\ p \in \text{paths}}} \left\{ (u, u') \;\middle|\; \begin{array}{l} u' \text{ is uniform with } \text{free}(\pi', u) \\ \text{and } (p, G, (u, u')) \models \pi' \end{array} \right\}$$

Ambiguous and Edge Cases

Nulls in Patterns

```
MATCH (n : Person {name : null})
RETURN (n)
```

Nulls in Patterns

```
MATCH (n : Person {name : null})
RETURN (n)
```

1. Every node n with a name property?
2. Every node n such that $n.name$ `IS NULL` $=$ `true`?
3. Nothing?

Nulls in Patterns

```
MATCH (n : Person {name : null})
RETURN (n)
```

1. Every node n with a name property?
2. Every node n such that $n.name$ `IS NULL` $=$ `true`?
3. Nothing!

Because Q is actually equivalent to:

```
MATCH (n : Person)
WHERE n.name = null
RETURN (n)
```

Map Comparisons

When does $\{k_1 : v_1, \ldots, k_n : v_n\} = \{\ell_1 : w_1, \ldots, \ell_m : w_m\}$ return `true`, `false` or `null`?

$\{name : null\} = \{\}$

$\{a : 1, b : 2\} = \{b : 2, a : 1\}$

$\{name : null\} = \{name : null\}$

$\{a : 1, a : 2\} = \{a : 2\}$

Map Comparisons

When does $\{k_1 : v_1, \ldots, k_n : v_n\} = \{\ell_1 : w_1, \ldots, \ell_m : w_m\}$ return `true`, `false` or `null`?

`false`

$\{name : null\} = \{\}$

`true`

$\{a : 1, b : 2\} = \{b : 2, a : 1\}$

`true`

$\{name : null\} = \{name : null\}$

`true`

$\{a : 1, a : 2\} = \{a : 2\}$

Neither purely syntactic, nor $\forall k, m_1.k = m_2.k$.

Setting Properties using a Map

```
WITH {name : null} AS map
CREATE (n)
SET n = map
RETURN (n)
```

Setting Properties using a Map

```
WITH {name : null} AS map
CREATE (n)
SET n = map
RETURN (n)
```

Returns n as $\{\}$.
The property map of n is not equal to the map it was set to.
In particular, $n\{.*\} = map$ returns `false`.

MATCH with no Free Variables

```
MATCH ()
RETURN *
```

```
MATCH ()
RETURN 1
```

MATCH with no Free Variables

Fail

```
MATCH ()
RETURN *
```

RETURN * is not allowed with no variable in scope.

Pass

```
MATCH ()
RETURN 1
```

Returns as many copies of 1 as nodes in the database.

After MATCH (), the active table is a bag containing multiple copies of the empty record.

Incomplete and Inconsistent Cases

Repeating UNWINDs

```
UNWIND [1, 2, 3] AS r
UNWIND r AS s
RETURN s
```

```
UNWIND [[1, 2], 3] AS r
UNWIND r AS s
RETURN s
```

Repeating UNWINDs

Fail

```
UNWIND [1, 2, 3] AS r
UNWIND r AS s
RETURN s
```

Type mismatch, expected List but was Integer.

Pass

```
UNWIND [[1, 2], 3] AS r
UNWIND r AS s
RETURN s
```

Returns a column with 1, 2 and 3 as rows.

Repeating UNWINDs

Fail

```
UNWIND [1, 2, 3] AS r
UNWIND r AS s
RETURN s
```

Type mismatch, expected List but was Integer.

Pass

```
UNWIND [[1, 2], 3] AS r
UNWIND r AS s
RETURN s
```

Returns a column with 1, 2 and 3 as rows.

```
UNWIND [[1, 2], 3] AS r
UNWIND r AS s
UNWIND s AS t
UNWIND t AS u
RETURN u
```

Actually works, and returns a column with 1, 2 and 3 as rows.

Violating Cyphermorphism

$$Q_0 = \begin{array}{l} \texttt{MATCH}\ (x) - [r] - (y) - [r] - (z) \\ \texttt{RETURN}\ x, y, z \end{array}$$

Violating Cyphermorphism

$$Q_0 = \begin{array}{l} \texttt{MATCH}\ (x) - [r] - (y) - [r] - (z) \\ \texttt{RETURN}\ x, y, z \end{array}$$

Error: cannot use the same relationship variable ‘r’ for multiple patterns.

Violating Cyphermorphism

$$Q_0 = \begin{array}{l} \texttt{MATCH}\ (x) - [r] - (y) - [r] - (z) \\ \texttt{RETURN}\ x, y, z \end{array}$$

Error: cannot use the same relationship variable ‘r’ for multiple patterns.

$$Q_1 = \begin{array}{l} \texttt{MATCH}\ (x) - [r*] - (y) - [r*] - (z) \\ \texttt{RETURN}\ x, y, z \end{array}$$

Violating Cyphermorphism

$$Q_0 = \begin{array}{l} \texttt{MATCH } (x) - [r] - (y) - [r] - (z) \\ \texttt{RETURN } x, y, z \end{array}$$

Error: cannot use the same relationship variable 'r' for multiple patterns.

$$Q_1 = \begin{array}{l} \texttt{MATCH } (x) - [r*] - (y) - [r*] - (z) \\ \texttt{RETURN } x, y, z \end{array}$$

Works and enforces the paths from x to y and from y to z to use the same sequence of relationships, violating Cyphermorphism.

Violating Cyphermorphism

$$Q_0 = \begin{array}{l} \texttt{MATCH } (x) - [r] - (y) - [r] - (z) \\ \texttt{RETURN } x, y, z \end{array}$$

Error: cannot use the same relationship variable ‘r’ for multiple patterns.

$$Q_1 = \begin{array}{l} \texttt{MATCH } (x) - [r*] - (y) - [r*] - (z) \\ \texttt{RETURN } x, y, z \end{array}$$

Works and enforces the paths from x to y and from y to z to use the same sequence of relationships, violating Cyphermorphism.
It is not included in the query Q_2 below:

$$Q_1 = \begin{array}{l} \texttt{MATCH } (x) - [r*] - (y) - [s*] - (z) \\ \texttt{RETURN } x, y, z \end{array}$$

Two Small Issues: Nulls as Indices and Keys, and Naming

```
RETURN 1 AS `0`, 0
```

Two Small Issues: Nulls as Indices and Keys, and Naming

Fail

```
RETURN 1 AS `0`, 0
```

Multiple result columns with the same name are not supported.
Need to specify how expression naming is handled.

Two Small Issues: Nulls as Indices and Keys, and Naming

Fail

```
RETURN 1 AS `0`, 0
```

Multiple result columns with the same name are not supported.
Need to specify how expression naming is handled.

$[1, 2, 3][\texttt{null}] \qquad [1, 2, 3][\texttt{null..4}]$

$\{\mathit{name} : \texttt{'Nadime'}\}[\texttt{null}]$

Two Small Issues: Nulls as Indices and Keys, and Naming

Fail

```
RETURN 1 AS `0`, 0
```

Multiple result columns with the same name are not supported.
Need to specify how expression naming is handled.

Fail

$[1, 2, 3][\texttt{null}] \qquad [1, 2, 3][\texttt{null..4}]$

$\{name : \texttt{'Nadime'}\}[\texttt{null}]$

No error, never terminates.

Thank you!