Foundations of Data and Knowledge Bases

Graph Databases and Regular Path Queries

Joachim Niehren

Links: Linking Dynamic Data
Inria Lille

November 10, 2020

Outline

OReilly Book

title: Graph Databases
subtitle: New Opportunities for Connected Data
authors : Ian Robinson, Jim Webber & Emil Eifrem
first edition: 2013
second edition: 2015

Graph Structures and Queries in Neo4J

Page 20 : nice example graph database
Page 28 : first Cypher pattern
Page 29 : second Cypher pattern
Page 29: first Cypher query (match)
page 69: big Cypher query with aggregation

Questions

What is a Neo4J Graph database formally?
How can we formalize the core of its query language?
How can we add recursive paths to the query language of Neo4J

Outline

Data Graphs

Alphabets

Δ	finite alphabet for strings
L	ranked alphabet of node labels
R	finite set of edge labels

Data Value Annotations

$w \in \Delta^*$	data values
$a(w_1, \ldots, w_n)$	node annotations
	where $a \in L$, $\mathrm{ar}(a) = n$, $w_1, \ldots, w_n \in \Delta^*$

Colored Graphs Annotated with Data Values

- every node carries a data value annotation, i.e., a color which is a node label and a tuple of data values.
- every edge carries an edge label $r \in R$

Outline

Conjunctive Regular Path Queries: R2PQs

variables	x, y, z	::=	...	
string pattern	pat	::=	_	matches any string
		|	$'w'$	matches string 'w'
node filter	f	::=	$a(pat_1, ..., pat_n)$	where $a \in L,\ n = \mathrm{ar}(a)$
step	s	::=	r	forwards edge with $r \in R$
		|	r^-	backwards edge with $r \in R$
		|	$s[f]$	step with node filter
		|	$s[p]$	step with path filter
path	p	::=	s	
		|	ϵ	empty path
		|	p/p	path composition
		|	$p + p'$	path choice
		|	p^*	path repetition
queries	q	::=	$x\ p\ y$	path literal
		|	$q \wedge q'$	conjunction
		|	$\exists x.q$	existential quantification

Example Queries

edge labels $R = \{friend, address, \ldots\}$, node labels $L = \{person, \ldots\}$

1. y is a friend of x: $x\ friend\ y$
2. y is accessible over friends from x: $x\ friend^*\ y$
3. there is a person named $'Ulman'$ that is a friend of x and whose adress is y: $x\ friend[person(\_,'Ulman',\_)/address\ z$
4. x is a friend of a person y named 'Ulman' and having an address in 'Los Angelos':
   $x\ friend^-[person(\_,'Ulman',\_)][address[city :' LosAngelos']]$

Exercises

1. Define a query in the calss R2PQ that states that x is a friend of a friend of y and that y is also a friend of a friend of x.
2. Define a query in the class R2PQ that states that x is a friend of a friend of y, or x is friend of a friend of a friend of y.

Can one express conjunctive path queries in FO?

yes if we rule out recursive path definitions p^*
but no otherwise

What is the Relational Structure of Data Graphs

domain $= \mathit{Nodes} \uplus \Delta^*$

signature=
$\{\mathit{edge}_r, \mathit{lab}_a, \mathit{data}_i \mid a \in L, r \in R,\ 1 \leq i \leq \mathit{max}\{\mathit{arity}(a) \mid a \in L\}\}$

edge_r = set of pairs of nodes linked by edge labeled by $r \in R$
lab_a = set of nodes that are labeled by $a \in L$
data_i = binary relation between a node of the graph and its i-th data value

Example of Translation to FO

$$x \; friend[person(\_,' Ulman', \_)]/address \; y$$

becomes

$$\exists z.edge_{friend}(x, z) \wedge lab_{person}(z) \wedge data_2(z,' Ullman') \wedge edge_{address}(z, y)$$

General Translation to FO

Node pattern

$[\![a(p_1 ... p_n)]\!]_x = lab_a(x) \wedge \wedge_{1 \leq i \leq n, p_i \neq \_} data_i(x, ' w')$

Steps

$[\![r]\!]_{xy} = edge_r(x, y)$
$[\![r^-]\!]_{xy} = edge_r(y, x)$
$[\![s[f]]\!]_{xy} = [\![s]\!]_{xy} \wedge [\![f]\!]_x$
$[\![s[p]]\!]_{xy} = [\![s]\!]_{xy} \wedge \exists z.[\![p]\!]_{xz}$

General Translation continued

Paths

$[\![\epsilon]\!]_{xy} = (x = y)$
$[\![p/p']\!]_{xy} = \exists z.[\![p]\!]_{xz} \wedge [\![p']\!]_{zy}$
$[\![p + p']\!]_{xy} = [\![p]\!]_{xy} \vee [\![p']\!]_{xy}$
$[\![p*]\!]_{xy} =???$

Conjunctive queries

$[\![xpy]\!] = [\![p]\!]_{xy}$
$[\![q \wedge q']\!] = [\![q]\!] \wedge [\![q']\!]$
$[\![\exists x.q]\!] = \exists x.[\![q]\!]$

Outline

How to extend FO for recursive steps?

Simple Idee: add $edge_r^*$ to alphabet

- but then one cannot say $(edge_r[p])^*$ except if graph is acyclic
- neither can we say $(edge_r + edge_{r'})^*$
- neither can we say $(edge_r/edge_r)^*$

Full Solution: Add recursion operator * to FO-logic

$$\phi ::= .... \mid \{(x, y) \mid \phi\}^*$$

Regular axis can then be expressed as follows:

$$[\![p^*]\!]_{xy} = \{(x, y) | [\![p]\!]_{xy}\}^*$$