Querying incomplete graphs with data

Gaëlle Fontaine[1] and Amélie Gheerbrant[2]

[1] Department of Computer Science, Universidad de Chile
[2] LIAFA (Université Paris Diderot - Paris 7 & CNRS)

1 Introduction

Graph databases underlie several modern applications such as social networks and the Semantic Web. In those scenarios, integrating and exchanging data is very common, which leads to proliferation of incomplete graph data. However, the well developed models of incompleteness of data do not apply to graph data. This is mainly due to the fact that standard graph query languages concentrate on graph *topology*; this requires functionalities beyond the abilities of standard relational systems. Besides, many graph languages ignore the actual data stored.

However, recently languages combining data and topology aspects of querying have been proposed for graph databases. An example is a query *Find pairs of people in a social network connected by professional links restricted to people of the same age*). Formalisms developed to handle such queries include *regular expressions with memory* (REM), *regular expressions with equalities* (REE) [5], their extensions [1], as well as variants of XPath [4].

Handling incompleteness by languages dealing with pure graph topology has been studied in [2]. In this short note, we present preliminary results on dealing with incompleteness at the levels of both data and topology, using some of the recently proposed query languages.

2 Preliminaries

Let Σ be a finite alphabet Σ, let $\mathcal{N}$ be a countable set of node ids and let $\mathcal{D}$ be an infinite alphabet of data values. A *data graph* G (over Σ, $\mathcal{N}$ and $\mathcal{D}$) is a tuple (V, E, ρ), where $V \subseteq \mathcal{N}$ is a finite set of nodes, $E \subseteq V \times \Sigma \times V$ is a set of Σ-labeled edges, and $\rho : V \to \mathcal{D}$ assigns a data item to each node. A *path* π in G is a sequence $v_0 a_0 v_1 a_1 v_2 \cdots v_{k-1} a_{k-1} v_k$ such that $(v_{i-1}, a_{i-1}, v_i) \in E$, for each $i \leq k$. The *data path* associated with π is the word $\rho(v_0)a_0\rho(v_1)a_1 \cdots a_{k-1}\rho(v_k)$.

The *regular expressions with equality* (REE) [5] are defined by the grammar

$$e ::= \epsilon \mid a \mid e.e \mid e \cup e \mid e^+ \mid \mid e_= \mid e_{\neq},$$

where a ranges over Σ. Given a REE e, $L(e)$ is defined by induction by the following rules

$$L(\epsilon) = \{d : d \in \mathcal{D}\}, \qquad L(e_=) = \{d_1 a_1 \dots d_{n_1} a_{n-1} d_n \in L(e) : d_1 = d_n\}$$
$$L(a) = \{dad' : d, d' \in \mathcal{D}\}, \qquad L(e_{\neq}) = \{d_1 a_1 \dots d_{n_1} a_{n-1} d_n \in L(e) : d_1 \neq d_n\},$$

and the standard rules for $L(e.e')$, $L(e \cup e')$ and $L(e^+)$. REE expressions without $\neq$ are called *positive*. The *evaluation* $e(G)$ of an REE e on a data graph G gives pairs of nodes (u, v) such that there is a data path between them that belongs to $L(e)$. The combined complexity of REEs is PTIME, while the data complexity is in NLOGSPACE. REEs are a subclass of a more expressive class REM whose data complexity remains the same but combined complexity is in PSPACE [5].

Let $\mathcal{V}_{data}$ be an infinite set of label variables, let $\mathcal{V}_{node}$ be an infinite set of nodes variables and let $\mathcal{V}_{label}$ be an infinite set of data variables. A *data graph pattern* (over Σ, $\mathcal{N}$ and $\mathcal{D}$) is a data graph over the alphabets $\Sigma \cup \mathcal{V}_{label}$, $\mathcal{N} \cup \mathcal{V}_{node}$ and $\mathcal{D} \cup \mathcal{V}_{data}$. That is, a data graph pattern is a data graph with constant nodes and node variables, whose edges can be labeled with variables and elements of Σ, and in which the data values are taken from $\mathcal{D} \cup \mathcal{V}_{data}$.

Given a data graph pattern $G = (V, E, \rho)$ and a data graph $G' = (V', E', \rho')$, a *homomorphism* is a triple (h_1, h_2, h_3) of mappings $h_1 : V \to V'$, h_2 that maps label variables in G to letters in Σ, h_3 that maps data variables in G to data values in $\mathcal{D}$ such that

- $h_1(n) = n$ for all $n \in \mathcal{N}$,
- for every edge $(p, x, p') \in E$, the edge $(h_1(p), h_2(x), h_1(p'))$ belongs to E',
- for every pair $(p, x) \in \rho$, the pair $(h_1(p), h_3(x))$ belongs to ρ'.

The data graph G' is an *homomorphic image* of the data graph pattern G if $V' = h_1(V)$.

Given a data graph pattern G and a REE e, the *certain answers* of e over G are defined as [3]

$$\bigcap \{e(G') : \ G' \text{ is an homomorphic image of } G\}.$$

3 Evaluation of REEs over incomplete graphs

Theorem 1. *The combined and data complexities of finding certain answers of REEs over incomplete graph databases are* coNP*-complete. This remains true for incomplete graphs without label variables.*

Membership in coNP follows from tractability of REEs. Hardness is by reduction to Positive 1-in-3 SAT. The proof extends to show that data complexity of finding certain answers of REMs is coNP-complete, and their combined complexity is PSPACE-complete.

Given a formula ϕ in CNF with 3 variables in each clause, we design an incomplete graph data such that an homomorphic images with data in $\{0, 1\}$ corresponds to a valuation of the variables in ϕ. We define a REE e such that e is false in an homomorphic image G' iff the data of G' belong to $\{0, 1\}$ and the

[3] The reader familiar with incompleteness may notice that we adopted the closed world semantics. Given the definition of REE (and also of REM), it is easy to see that the open world semantics and the closed world semantics coincide.

valuation associated with G' makes exactly one variable true in each clause of ϕ.

For positive REE, we can obtain a tractable algorithm by using a so-called *naive evaluation*, cf. [3]. That is, we directly evaluate the REE over the incomplete data graph, treating the label variables as regular labels and the data variables as regular data.

Proposition 1. *The combined complexity of finding the certain answers of positive REEs over incomplete graph databases is tractable.*

We can also show, using naive evaluation, that data complexity of certain answers of positive REMs is tractable. The exact combined complexity though is open.

4 Future work

Many questions are left open. Instead of REM and REE, we could investigate incompleteness for other languages designed to query graph databases. Natural candidates would be versions of XPath for graph databases [4] and register logic [1]. Another direction of research is to consider a more general notion of incomplete data graph (as in [2]), where we also allow the labels of the edges to be regular expressions, as opposed to just label variables.

References

1. P. Barceló, G. Fontaine and A. Widjaja Lin. Expressive Path Queries on Graphs with Data. In *LPAR 2013*, pages 71–85.
2. P. Barceló, L. Libkin and J. L. Reutter. Querying graph patterns. In *PODS 2011*, pages 199–200.
3. T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.
4. L. Libkin, W. Martens and D. Vrgoc. Querying graph databases with XPath. In *ICDT 2013*, pages 129–140.
5. L. Libkin and D. Vrgoc. Regular path queries on graphs with data. In *ICDT 2012*, pages 74 – 85.