Ontology-Based Support for Graph Algorithms in Online Exploration Workflows

Thomas Hornung[1] and Wolfgang May[2]

[1] Institut für Informatik, Universität Freiburg
hornungt@informatik.uni-freiburg.de
[2] Institut für Informatik, Universität Göttingen
may@informatik.uni-goettingen.de

Abstract. Application domains often include notions that are inherently based on graph structures. In this paper, we propose CGDT as a comprehensive generic ontology and API for graphs, which is geared towards online exploration of potentially large graphs.

1 Introduction

A recurring motive when designing informational workflows is the computation of (parts of) transitive closures in graphs. In the context of the Web these graphs are neither known nor materialized a priori, but can be *explored* only at runtime, using one or more Web data source. Often, even the graph data itself is dynamic, which does not allow for materialization or caching. These characteristics require completely different algorithms where the exploration and expansion strategy for the graph itself is the central issue.

We present *CGDT (Configurable Graph DataType)* that provides an ontology and an API for *configurable* graphs. The design of CGDT combines generic graph behavior (insertion of edges etc.) with application-specific configurability. CGDT allows to encode the maintenance of the stored graph data inside the graph by (i) assigning properties to vertices, edges, and paths, and (ii) specifying how paths are obtained from existing edges and paths during the exploration process. For details and a sample use case refer to the long version of this paper[1].

2 An Ontology for Graphs in Online Algorithms

The basic notions of any graph ontology are vertices, edges, and paths. While in the usual notion of graphs, the set of paths is defined as the transitive closure of edges, the set P of relevant paths in a configurable graph is a certain *subset* of all existing paths in the graph that satisfy additional constraints.

A central feature of CGDT is that vertices, edges and paths can be adorned with (typed) properties, which can optionally be specified in terms of view definitions over other properties, or by external queries. When new edges are added,

[1] http://www.dbis.informatik.uni-goettingen.de/Publics/

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 13–14, 2009.

the emerging new paths (wrt. the path insertion conditions) are computed, and their path properties are derived, e.g. by an inductive definition over the length of the path. To control the insertion of vertices and edges or the extension of paths, conditions can be stated that need to be satisfied.

Signature and Operations. The operations of CGDT are divided into a *Data Definition Language (DDL)* where the properties and the constraints are *defined*, and a *Data Manipulation Language (DML)* that provides *generic* update and query operations.

The DDL. While in SQL and related languages, the DDL has an own syntax, the DDL of CGDT is actually the *ontology language* RDF that *declaratively* specifies which properties exist, together with their definitions, and with the constraints how to expand the graph.

In contrast to SQL, where the main notion of the schema is the *table*, CGDT is based on three subschemas each of which defines some properties and optionally some constraints that guide the exploration process (cf. Figure 1).

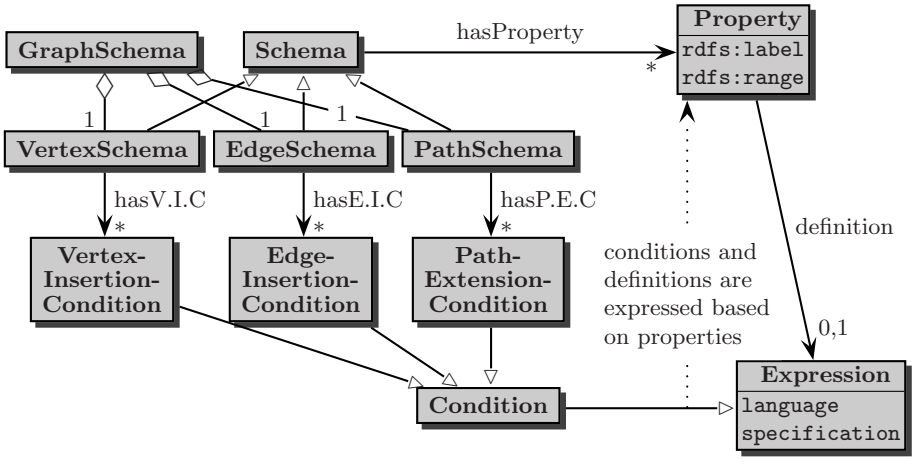

Fig. 1. Basic Notions of the CGDT Ontology

A concrete application-specific CGDT specification then defines

- the names and datatypes of the application-specific properties,
- the definitions of the derived properties,
- conditions to configure the exploration process.

The DML. The DML is also independent from the actual application domain (similar to e.g. SQL as DML for relational databases). The modifiers allow to add items to the graph and the accessors return newly reachable vertices based either on breadth-first or A^* best-first search.