Proposal for the Master's Thesis Grammar-based (Re-)Compression with Adjacent Digram

Matthias Dürksen

November 9, 2018

1 Motivation

The majority of data on the Internet is unstructured because it is mainly text. That makes it difficult for machines to extract knowledge from the data and answer specific queries. In the context of the Semantic Web, an attempt is made to build a knowledge base using structured data. Here the data is stored as graphs. A possible format for the Semantic Web is the Resource Description Framework (RDF)[1], in which a graph is represented by triples of the form $(subject, predicate, object)$, where *subject* and *object* are nodes and *predicate* is an edge of the graph.
In reality, such knowledge graphs can become very large. Therefore we are looking for a method to compress them. It should also be possible to execute queries directly on the compressed graphs. This may be faster on the compressed graphs than on the original graph due to the reduced size. For such a compression there is already an approach called HDT [2], where the graph is converted into a binary format.
The aim of this master's thesis is to develop a grammar-based compression method for RDF graphs. The advantage of a grammar-based compression is that the compressed data is still a graph. It is therefore possible that queries can be answered faster than with HDT. It is also possible that a higher compression rate can be achieved than with HDT. These two questions will be answered in the thesis.

2 Fundamentals

Grammar-based compression of graphs is a quite unexplored topic. However, there is an approach by Maneth and Peternek [3], which should form the basis of this work and is illustrated in Fig. 1. In the graph shown there are two occurrences of the same pattern, namely that a node has the incoming edge e_1 and the outgoing edge e_2. This pattern is

[1] https://www.w3.org/RDF/

then stored in the digram A and the two occurrences are each replaced by a new edge with the label A.
The compression algorithm works as follows. In each iteration of the loop, the digram occurring most often is found. Then all occurrences of that digram are being replaced. After that the loop continues. The loop stops if there is no digram occurring more than once in the graph.

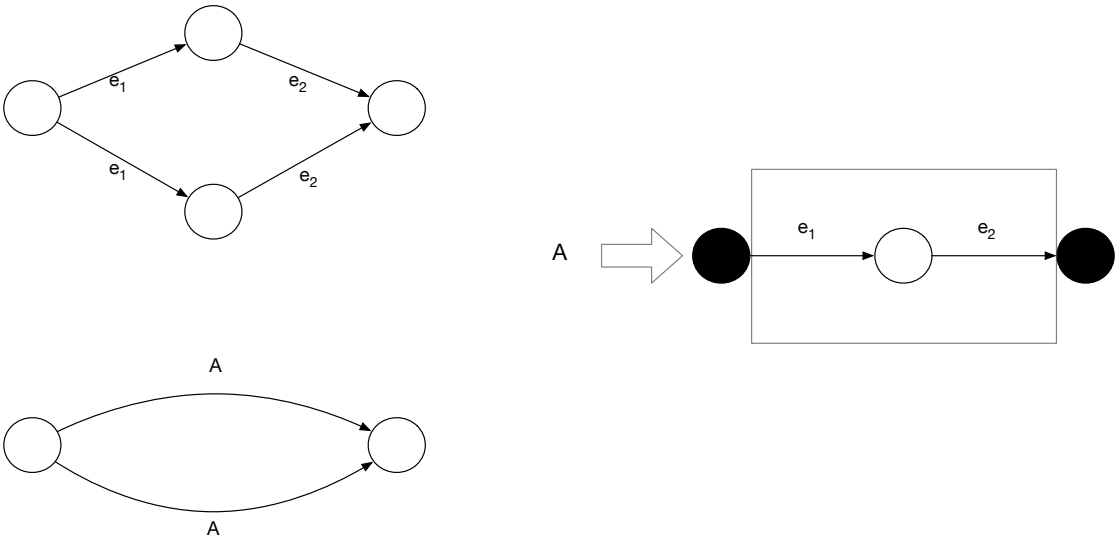

Figure 1: Replacing two different occurrences of the digram A [3]

Apart from [3] there is another approach for a grammar-based compression of graphs by Duerksen [1] . With this approach the graph is first transformed, as illustrated in Fig. 2. There for each edge a new node (with the former label of the edge) is introduced. Thus two nodes which were previously connected via the edge e, are now indirectly connected via the node with the label e. The result of the transformation is a graph that only has node labels and no edge labels anymore, which makes compression easier.
The advantage of the transformation can be seen in Fig. 3. After the transformation, the four different classes of digrams merge into two different classes. The first of the two different classes is the adjacency digram and the other is the base digram. Adjacency digrams are based on adjacent nodes (digram type 1) or indirectly connected edges (digram type 2). The second type corresponds exactly to the digram of approach [3]. So you can simulate algorithm [3] with algorithm [1]. The advantage of [1], however, is that further digram types are possible. These include base digrams, which are based on the fact that a node has an incident edge with the same label incoming (digram type 4) or outgoing (digram type 3). Such patterns are rather untypical in RDF graphs, but can sometimes occur.

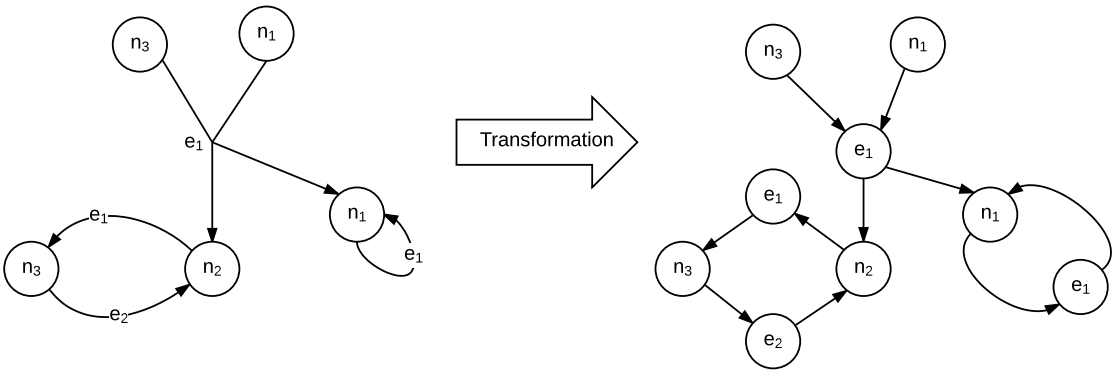

Figure 2: Transformation of a graph [1]

3 This Thesis

First, it should be investigated which one of the approaches [3] and [1] works better for RDF graphs. In [3] a disadvantage is that the nodes have no labels. Therefore we would have to adapt [3] for this thesis. The adaption can be seen in Fig. 4. Here the nodes have labels. The digram A now has a parameter v_1 which represents the node v_1 between the two edges of A. Since the different occurrences of A can have different middle nodes, such a parameter is necessary. The two occurrences here each have the middle nodes v_2 and v_4. Therefore the occurrences are replaced by an edge with the label $A(v_2)$ and $A(v_4)$, respectively.
For [1] no adaption is necessary.
After the better algorithm has been selected (mostly depending on the compression ratio), the next step is to optimize the algorithm in terms of RDF-specific features. For example, there are symmetrical predicates in RDF, i.e. you can reverse the edge direction (without changing the semantics of the graph) to achieve a possibly higher compression rate.
When the graph has finally been fully compressed by the algorithm (no digrams occur multiple times), the compressed graph must be represented by efficient encoding to minimize memory requirements in order to compare the compression ratio to the one of HDT. The encoding method, of course, depends of the selected compression algorithm. For [3] it would be possible to adapt the already existing method of [3] for the newly introduced parameterization.
For [1] it yet only exists an encoding method for basis digrams, not for adjacency digrams. The latter one would have to be developed.
A completely different approach would be to pass the compressed graph to the HDT encoding. HDT would have to be adapted to be able to handle digrams.
The second big question of this thesis is about the speed of queries on the compressed graph. These are so-called SPARQL[2] queries. SPARQL is a query language for graphs.

[2] https://www.w3.org/TR/rdf-sparql-query/

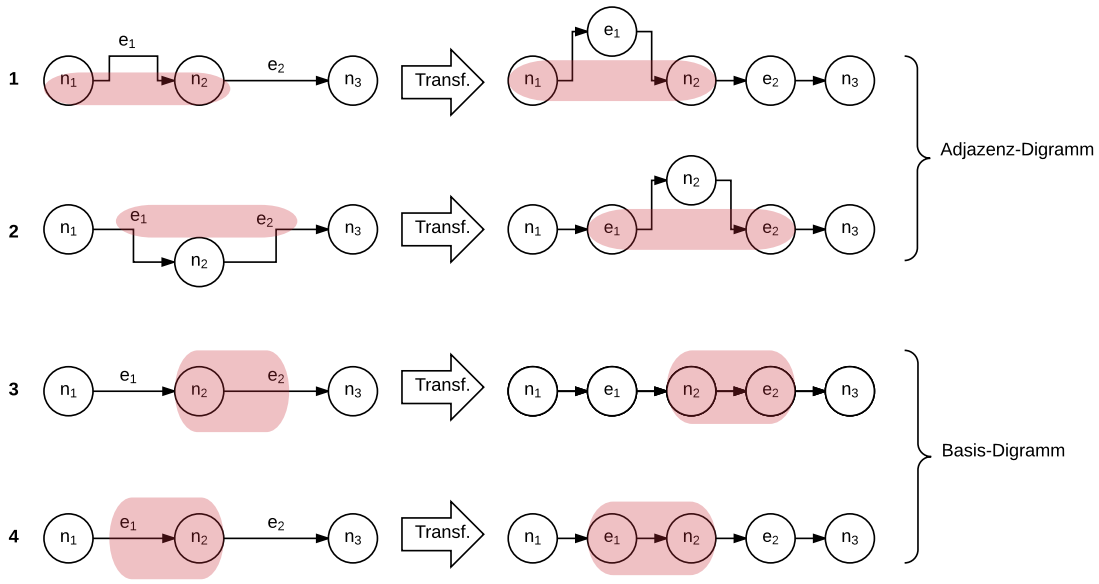

Figure 3: Different classes of digrams before and after the transformation from [1].

The queries are based on triples of the form (S, P, O) (Subject, Predicate, Object). One or two components of the triple can be variable (e.g. $(S, ?, O)$). Concrete values are then inserted for all non-variable components of the triple. The return value for such a query is then all triples of the graph that correspond to the query pattern.
As already mentioned, a graph compressed by a grammar-based algorithm is still a graph, so it is conceivable that certain types of queries can be executed faster than with a graph compressed by HDT.
For a sophisticated evaluation real queries made by users are available on the Internet. In the evaluation queries could also be divided in different classes to see which classes are answered faster on which compression algorithm.
In the course of the thesis, an algorithm for answering queries on compressed graphs must firstly be created. There are already approaches in [3] and [1] for answering queries. It remains to be investigated how well they can be used for SPARQL queries.

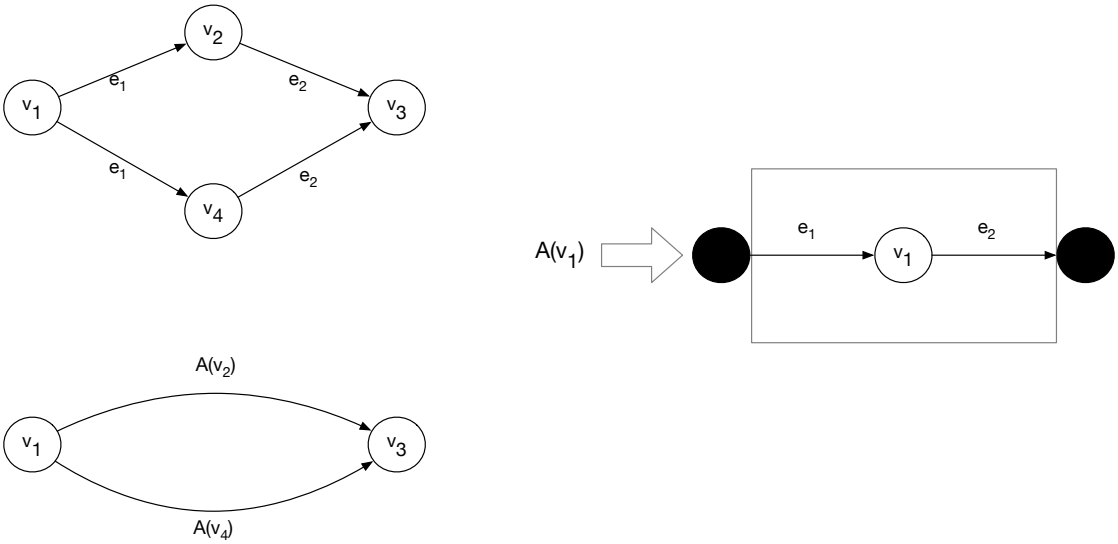

Figure 4: Replacing two different occurrences of the digram A with labeled nodes. Adapted version of [3].

References

[1] Matthias Duerksen. "Grammatik-basierte Graphkompression". Bachelor's Thesis. University of Paderborn, 2016.

[2] Javier D. Fernández et al. "Binary RDF representation for publication and exchange (HDT)". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 19 (2013), pp. 22–41. ISSN: 1570-8268. DOI: https://doi.org/10.1016/j.websem.2013.01.002. URL: http://www.sciencedirect.com/science/article/pii/S1570826813000036.

[3] Sebastian Maneth and Fabian Peternek. "Compressing Graphs by Grammars". English. In: *Proceedings of the 32nd International Conference on Data Engineering – ICDE 2016*. IEEE, June 2016. DOI: 10.1109/ICDE.2016.7498233.