UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Databases and Electronic Commerce

Master's Thesis

Submitted to the Databases and Electronic Commerce Research Group
in Partial Fullfilment of the Requirements for the Degree of

Master of Science

Extended Grammar-based Graph Compression

by
Matthias Dürksen

Thesis Supervisor:
Prof. Dr. Stefan Böttcher

Paderborn, May 2, 2019

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum	Unterschrift

Abstract. We present a full documentation of the Paderborn University Computer Science thesis template (UPB-CS-TT) and how to use it. This document also serves as a demonstrator to show what documents UPB-CS-TT produces. Have fun!

Contents

1 Introduction

todo

2 Concept

Equivalence classes

Equivalence classes for nodes

If several nodes are replaced by a Nonterminal node, then it must still be possible to identify to which inner node an edge is incident.

Fig. 2.1 show which edge is incident to V and which edge is incident to X. The following figure shows which edge is incident to X.

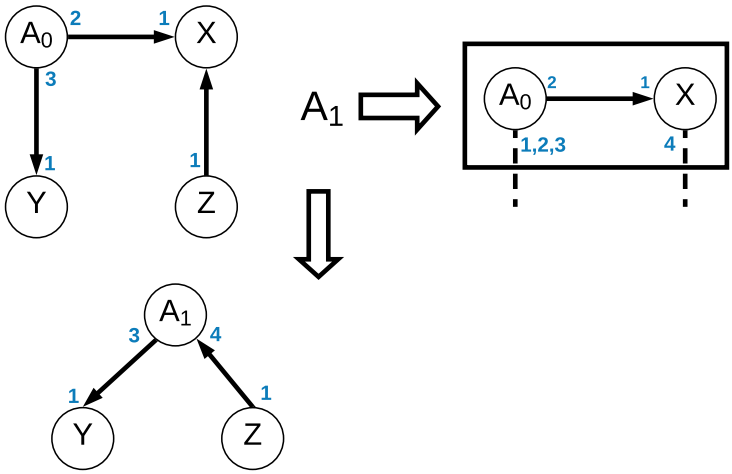

Figure 2.1: Visualization of equivalence classes of the nodes

Therefore, the inner nodes of a nonterminal node rule are numbered consecutively, where we call the numbers of the inner nodes equivalence classes. Thus, incident edges of a nonterminal symbol are labeled with the equivalence class of the inner node to which the edge was incident before its replacement.

If the node is a terminal, it has equivalence class 1. To simplify matters, the equivalence classes for terminals are often not displayed in the representations.

Equivalence classes for edges

If multiple edges are replaced by one nonterminal hyperedge, the information about which of the inner edges were incident to which node must still be available.

In Fig. 2.2 we see a replacement of several edges by a hyperedge. If there were no equivalence classes for the edges, one would not know to which node the inner edge with the label d must be incident. No information about the direction of the edge ends is required, so the hyperedge is undirected. Therefore, the edge ends are numbered, where we call the numbers equivalence classes for edges.

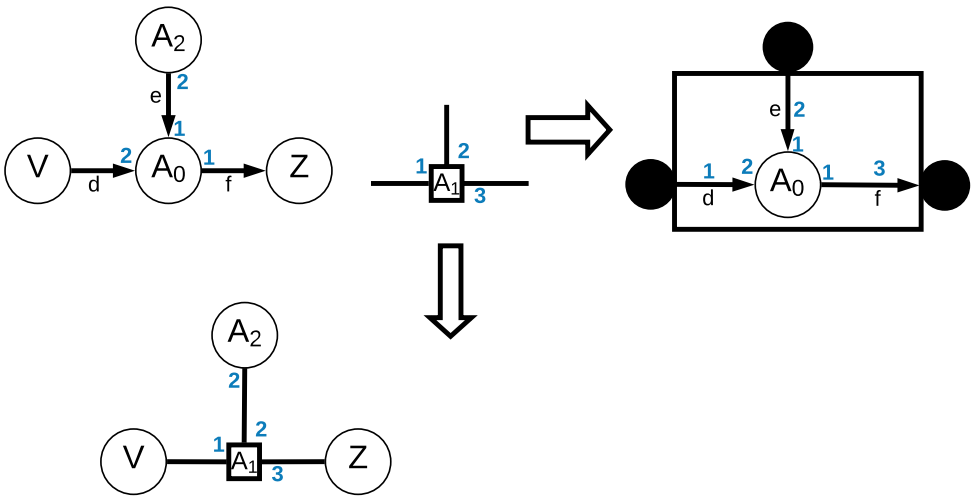

Figure 2.2: Visualization of equivalence classes of the edges

If two internal edges contain the same information (same direction, label and incident with the same equivalence class at the inner node), the edge ends have the same equivalence class. If there is now another incoming edge with label "e" and the equivalence classes are the same as the other edge with label "e", it can be replaced with the same digram, since the new edge also gets the edge equivalence class "2".

In the special case that an edge is not a hyperedge, the edge has two ends. Thus the edge can be represented normally and the equivalence classes 1 and 2 can be represented over one direction of the edge, see Fig. 2.3. The end, which is incident to the start node, has the equivalence class 1 and the end, which is incident to the end node, has the equivalence class 2.

Figure 2.3: Simplified representation of an edge with equivalence classes

If a Digram has only one external node, only one edge end is required. Since each edge must have 2 ends, an edge is inserted where both ends point to the desired node. An example can be seen in Figure 3 from the document "All different Types of digram Types from Maneth and Peternek".

Parameters

It may make sense to allow a parameter for a digram. If a pattern occurs again and again, but the node label at a node, for example, varies again and again, this can be combined into

a digram and the node label can be stored as a parameter. This can significantly increase the number of occurrences found.

Any internal information of a digram rule can be parameterized. Thus the edge direction, equivalence classes, nodes/edge labels can be parameterized.

In addition, external elements can possibly become internal elements and the varying information can be parameterized. Thus, for example, the external node in the adjacency digram (Section 2.3) becomes an internal node and its node label becomes a parameter. It should be noted that the modified form of adjacency digram will be a double application of basic digrams (Section 2.1). But this digram (Section 2.3) will probably not be created only with basic digrams, because the middle node label varies and therefore no frequently occurring basic digram can be found. This characteristic will probably also occur with other variations.

2.1 Basic digram

The basic digram (Fig. 2.4) consists of two adjacent nodes and the edge that connects these nodes. Thus, two adjacent nodes and the incident edge can be replaced by one nonterminal node. Each node is assigned with an equivalence class. For each occurrence the node labels (x and y), the edge label (e), and the equivalence classes (i and j) of the inner edge must match as defined in the rule.

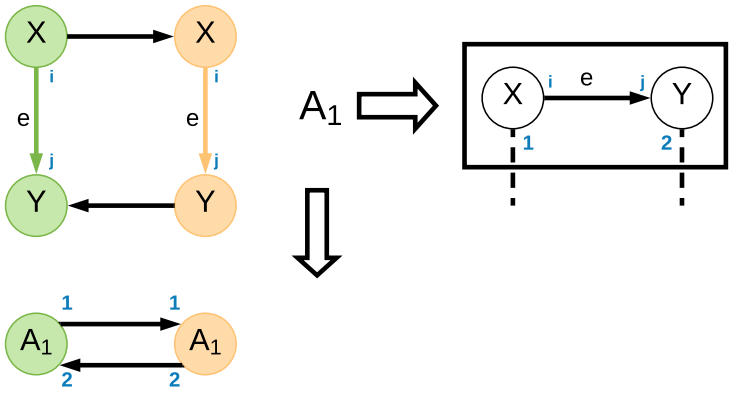

Figure 2.4: Application of a replacement of a basic digram

2.2 Circle digram

The circle digram (Fig. 2.5) contains a node and an edge, which is a self-cycle of the node. The node and the edge are replaced by a nonterminal node. For each occurrence the node label x, the edge label e, and the equivalence classes (i and j) of the edge must match as defined in the rule.

2.3 Adjacency digram

The adjacency digram (Fig. 2.6) is formed by two nodes which both have a common adjacency node U. Thus, the adjacency digram contains the two nodes and the two edges that are incident to the node U. This digram can be replaced by a nonterminal node and an edge from the nonterminal node to the node U. Each inner node and the external node is assigned an equivalence class. The incident edges must have the same equivalence classes k at the external node, but

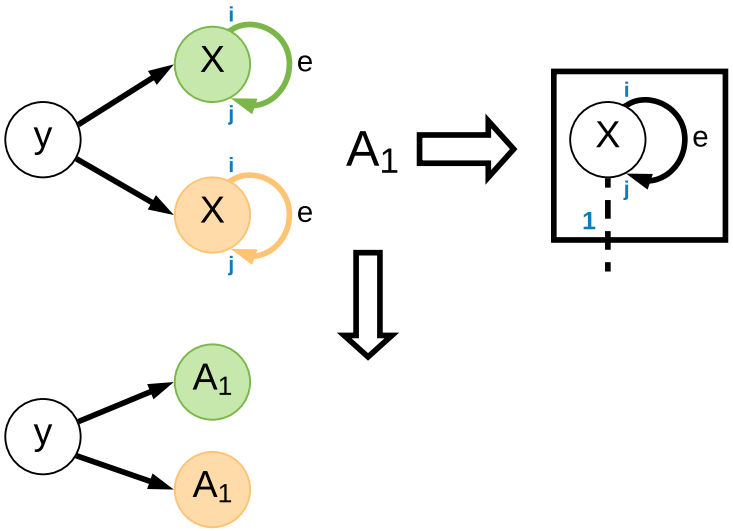

Figure 2.5: Application of a replacement of a circle digram

the equivalence class can differ for each occurrence of the digram. This is necessary because the two internal edges are replaced by one edge so that only one equivalence class can be assigned to the new edge at one end.

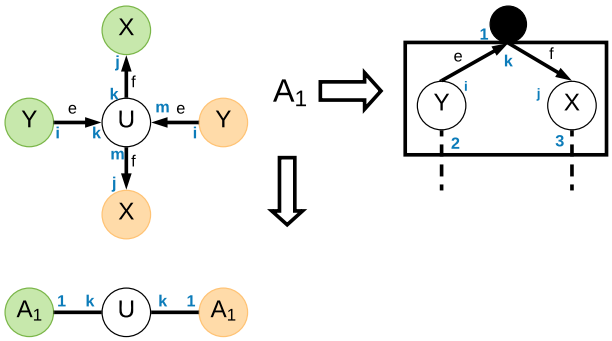

Figure 2.6: Application of a replacement of an adjacency digram

2.4 Node association digram

The node association digram (Fig. 2.7) is a generalization of an adjacency digram. This digram consists of two nodes, which are combined to one node. Each node is assigned with an equivalence class. In this way, frequently occurring nodes are replaced by a nonterminal node and these do not have to be adjacent. However, this replacement increases the number of incident edges per node, since the new node unites the incident edges of the old nodes.

2.5 Clique digram

The clique digram (Fig. 2.8) is a generalization of an adjacency digram. This clique digram consists of two nodes, which are strongly connected. The occurrences are replaced by a node. Each node is assigned an equivalence class. The digram can also be generated by applying the basic digram (Chapter 2.1) and then the circle digram(Chapter 2.2). **However, the Digram**

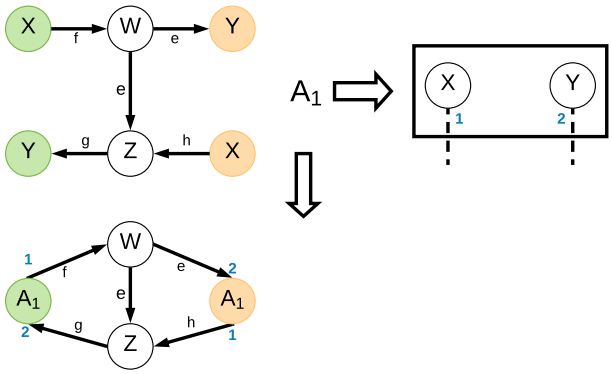

Figure 2.7: Application of a replacement of a node association digram

Clique offers only one advantage if all edges of the clique have the same label. Then the advantage is that the edges in the Digram do not have to be saved in the rule call, because it is clear that for every inner node from one node to every other inner node an edge exists. If the edges now have different labels, the labels must be able to be assigned to the edges again, so that the clique digram no longer offers any advantage. If one of the two nodes is already a nonterminal symbol, the other node should be strongly connected to each inner node.

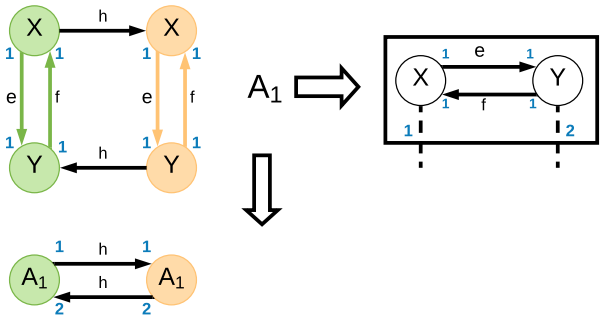

Figure 2.8: Application of a replacement of a clique digram

As the digram can be reproduced by other digrams, this type of digram is not taken into account when searching for and replacing the most frequent digram, but possibly only when pruning the existing rules to more efficient rules.

2.6 Basic edge digram

The basic edge digram (Fig. 2.9) is the opposite of the basic digram. Here, two edges and a node, where both edges are incident to the node, are replaced by one edge. The directions of the edges and the edge labels must be the same in all occurrences. In addition, the equivalence classes at the inner node must match. Furthermore, the inner node must not have any other incident edges than the two internal edges.

Incedent edges to an inner node

The basic edge digram described above only works if the inner node has no other incident edges. This is probably rarely the case in reality, so the rule has to be extended so that there can be an arbitrary number of nodes be incident to the edge.

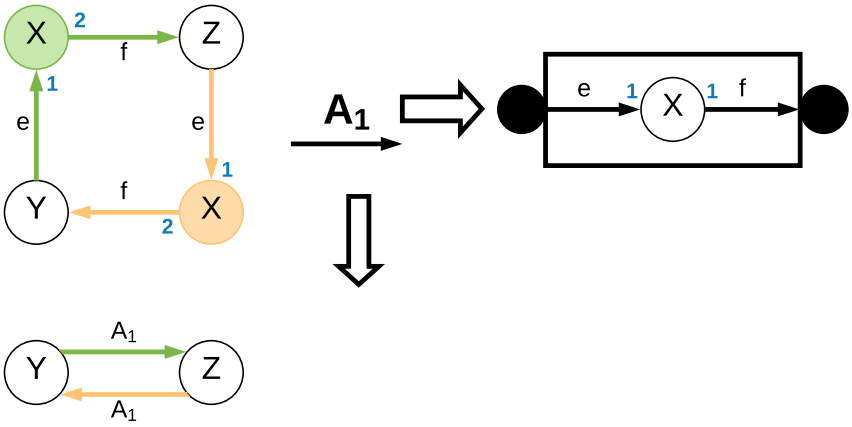

Figure 2.9: Application of a replacement of a basic edge digram

I currently have two approaches, both of which have their downsides.

Case 1: All incident edges are transformed into a hyper edge

The two internal edges (with the labels e and f) have been transformed into a simple edge A_1 so far. In addition, all incident edges of the node x are now added to the new edge A_1, so that the edge is a hyper edge if there are external incident edges. However, the information (edge direction, edge label and equivalence class to the internal node) of the individual edges must not be lost.

Figure 2.10 illustrates how the edge directions for all external incident edges can be handled. Therefore incoming edges get the equivalence class 3 and outgoing edges get the equivalence class 4.

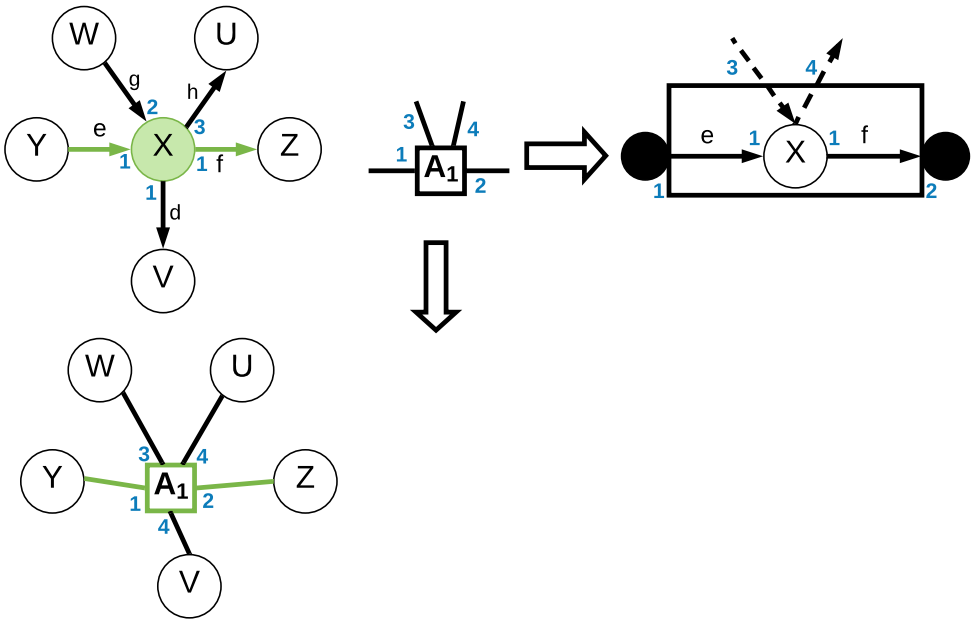

Figure 2.10: Application of a replacement of a basic edge digram with incident edges to the internal node

Thus, the edge label and the equivalence class must be parameterized for each external incident edge. You must also make sure that the parameters can be assigned to the correct edge

end of the hyper edge. Since the number of external incident edges is arbitrary, the number of parameters can be arbitrarily large.

In this case, the parameter must therefore be included:

- W: Label is g; equivalence class is 2
- U: Label is h; equivalence class is 3
- V: Label is d; equivalence class is 1
- Which end contained which labels

These parameters could then be saved as extended edge labels, for example. So the label would not only contain the nonterminal name but also the parameters. If the edge is to be replaced in another digram, the complete label must match so that the parameters match.

Case 2: External incident edges of the internal node become incident edges of an external node.

All external incident edges to X (see Fig. 2.11) become incident to the external nodes, which is the start node of the edge (in this case Y). This changes the number of incident edges to Y, where you have to be able to distinguish which of the edges were incident to X before. Therefore, the equivalence classes in Y must be changed so that this is possible.

Thus an existing node is changed and also the new nonterminal edge. Therefore, two persisting elements are changed by the replacement of the digram, whereby this type of digram turns the graph into a non context-free grammar. Furthermore it should be clarified how in node Y the label is not lost, but the reference to the rule A_1, which is normally represented by the label of the element, is still present. Additionally, by changing external nodes, the subsequent compression is made more difficult, because parts of the occurrences of Y are modified and are therefore no longer synonymous with unmodified Y occurrences.

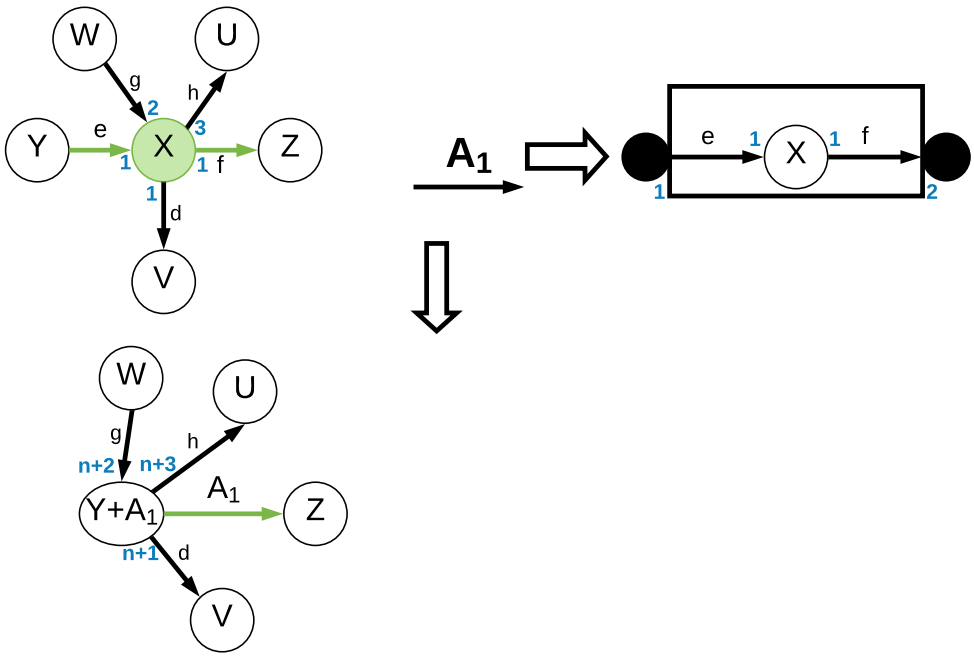

Figure 2.11: Application of a replacement of a basic edge digram with incident edges to the internal node. 'n' is the number of different equivalence classes of Y before substitution

2.7 Edge association digram

This edge association digram (Fig. 2.12) combines two adjacent edges to one hyper edge

Here only the two edges are internal elements of the rule. The equivalence classes at the middle external node must be the same. The replacement results in a hyper edge which in turn has equivalence classes of its own. The directions of the edges and the edge labels must be the same in all occurrences.

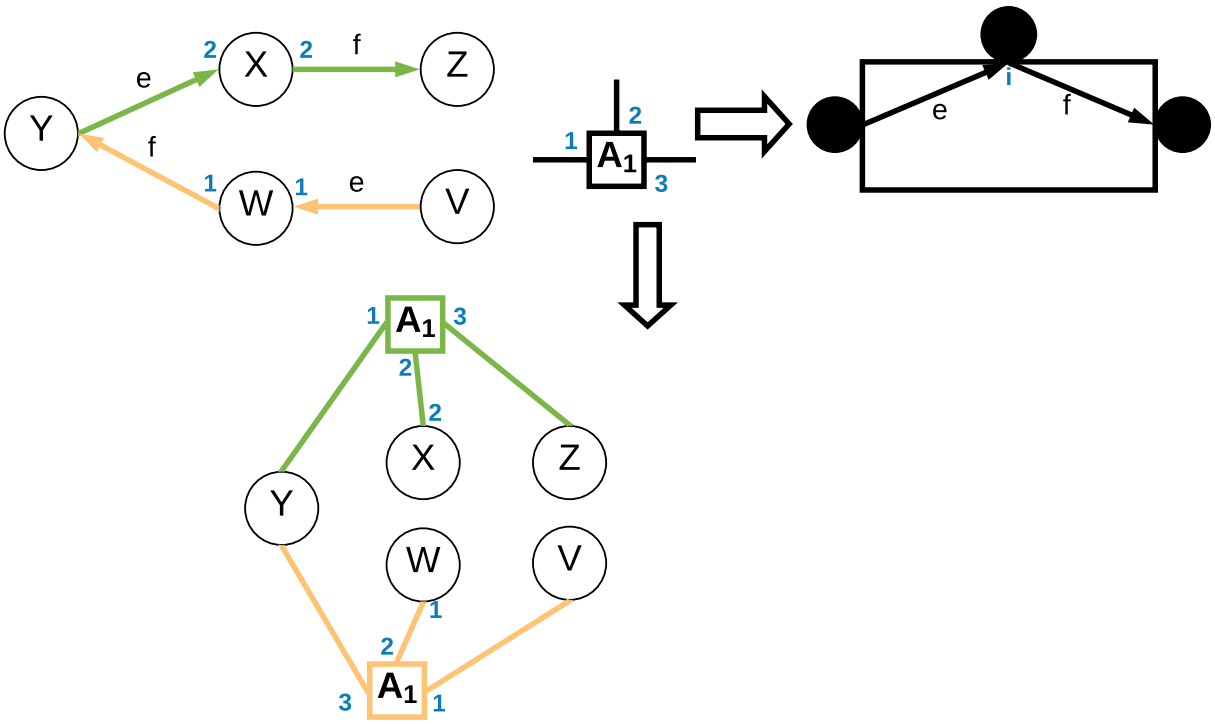

Figure 2.12: Application of a replacement of an edge association digram

2.8 Other digram types

All other digram types from Maneth can be adopted analogously. First, you should definitely determine how incident edges can be treated to internal nodes (Section. 2.6), since this case also occurs in the other types.

2.9 Combine substitutions of different digram types

In this section different digram types were presented, but the question arises whether all digram types can be combined. If a digram type is applied to a graph, the graph will be changed to a hyper edge, for example. If now other digram types are to be applied, it must not come to cases by the changed graph that something goes wrong with the further replacements. Thus it is considered here whether all Digram types are compatible with all others. Some digram types behave analogously to others. When replacing occurrences there are therefore only three different types of resulting objects.

The first possible result of the substitution is a substitution to nodes. The graph elements will be replaced from the occurrences to a node. The graph structure will not be violated. The external incident edges are connected from the internal nodes of the occurrence to the new nonterminal node. It can be that the equivalence classes of these edges change at the corresponding end. Thus nodes and edges are replaced by a node and the equivalence classes at the adjacent edges change. However, this is no reason why other digram types cannot be used.

ll matching
mtypes

Furthermore, there are digram types which occur when replacing edges. Here we distinguish the cases of a hyper edge (with more than 2 ends) and a simple edge (exactly 2 ends).

These result from the substitution of digrams to an edge. The internal edges and nodes contained in the digram occurrence are removed from the graph and a new edge is inserted so that the graph structure is preserved. When inserting the new edge, the edge gets the nonterminal of the digram rule as label. If the edge is a simple edge, it is directed and does not pose a problem for the new replacements.

In the case of a hyper edge substitution, an undirected hyper edge with more than two ends is inserted into the graph. Since not all Digram types can cope with the replacement of hyper edge, this can have a negative performance influence on further replacements, but no logical errors occur with the following replacements, so that for example a lossy compression results. The equivalence classes of the edges also do not lead to any errors in the subsequent compression.

In summary it can be said that each digram type can be used with any other digram type without hesitation, but from the point of view of Performance it may make sense in some cases not to use some digram types or to use them only after other digram types have been applied.

2.10 Search for Digram occurrences