Topological Analysis of Hydraulic Systems

Benno Stein and André Schulz

tr-ri-98-197

Topological Analysis of Hydraulic Systems

Technical Report
June, 1997

Benno Stein and André Schulz
email: stein@uni-paderborn.de

This work was supported by the DFG

Project number: KI 529/7-1; Schw 120/56-1

University of Paderborn
Department of Mathematics / Computer Science
Knowledge-based Systems
33095 Paderborn, Germany

Report tr-ri-98-197
Series Computer Science
April, 1998

Abstract

The concept of a hydraulic axis plays a central role within the design and analysis of hydraulic systems. A hydraulic axis comprises working, control, and supply elements that realize a subfunction of a hydraulic system.

Analyzing a system's functional structure means to identify the hydraulic axes along with their couplings. The paper in hand addresses this problem; it presents concepts and algorithms that tackle a sophisticated analysis task: the automatic identification of hydraulic axes and their related coupling types.

Aside from the theoretical elaboration of the analysis problem, a large part of the described concepts has been operationalized. The algorithms were evaluated with an extensive circuit library, and more than 95% of the hydraulic axes in these circuits have been identified correctly.

Contents

1 Introduction

The concept of a hydraulic axis plays a central role within the design and analysis of hydraulic systems. A hydraulic axis comprises working, control, and supply elements that realize a subfunction of a hydraulic system (cf. [11, 10]). Figure 1 gives a few examples for hydraulic axes. To realize complex driving processes, hydraulic axes are coupled. Together the hydraulic axes form the *functional structure* of a hydraulic system.

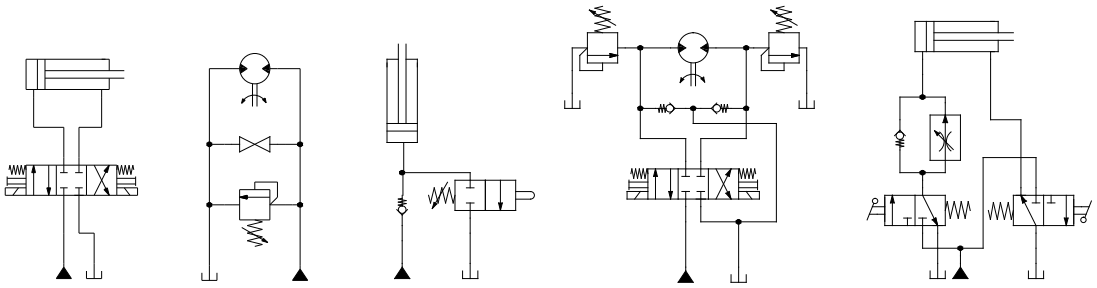

Figure 1: Examples for hydraulic axes.

Analyzing a system's functional structure means to identify the hydraulic axes along with their couplings. The paper in hand addresses this problem; it presents concepts and algorithms that tackle a sophisticated analysis task: the automatic identification of hydraulic axes and their related coupling types.

The paper is organized as follows. Section 2 discusses the hydraulic-axes-analysis-problem from an algorithmic point of view. It then outlines the analysis strategy for the detection of hydraulic axes and presents graph-theoretical as well as engineering foundations. To realize the identification of hydraulic axes, the complexity of the hydraulic graph must be reduced; section 3 develops the appropriate concepts. Section 4 presents an algorithm for the identification of hydraulic axes by means of path search, while section 5 outlines an alternative strategy: the identification of hydraulic axes by means of template embedding. Given a preprocessed graph, section 6 describes how the coupling level between hydraulic axes can be determined.

1.1 Operationalization

The main matter of this paper relates to graph theory rather than to hydraulic engineering. In this sense, theory and concepts from the field of graph theory have been outlined in the appropriate places. They form the base for our developments, that is, the selection, the adaptation, and the combination of graph theoretical algorithms which tackle the automatic identification of hydraulic axes and coupling types.

Aside from the theoretical elaboration of the analysis problem, a large part of the described concepts has been operationalized. In particular, the substances of the sections 3, 4, and 6 have been a matter of implementation.

The algorithms were evaluated with our circuit library, which contains more than 150 circuits at the moment. More than 95% of the hydraulic axes in these circuits are identified correctly by the algorithms. The solutions of the remaining cases are not completely off the track but contain a small number of incorrectly assigned components.

1.2 The Rationale of a Topological Analysis

Why is a topological analysis—especially respecting hydraulic axes—useful?

The view to hydraulic axes reveals basic design decisions. With respect to a well-founded analysis of hydraulic systems, their identification and classification plays a role regarding the following engineering tasks [9]:

- *Structure Envision.*
- *Demand Formulation and Interpretation.*
- *Smart Simulation.*
- *Optimization.*
- *Control Concept Selection and Evaluation.*
- *Diagnosis.*
- *Didactics.*

Within the normal design process, hydraulic axes are not used as explicit building blocks. The reasons for this are twofold: (*i*) It is not always possible to design a hydraulic system in a top-down manner, starting with hydraulic axes, which are refined within subsequent steps; (*ii*) both the experience and the ability of human designers to automatically derive function from structure enable them to construct a hydraulic system at the component level.

Note that the main working document for a designer is the technical drawing, and there is no tradition or standardized method to additionally specify the functional structure of a hydraulic system. This situation emphasizes the need for an automatic identification of the desired structural information.

2 Foundations

This section discusses the hydraulic axes analysis problem from an algorithmic point of view. It then outlines the analysis strategy for the detection of hydraulic axes and presents graph-theoretical as well as engineering foundations.

2.1 The Analysis Problem

When working with hydraulic engineering experts it becomes clear that a definition for the term "hydraulic axis" must stay imprecise up to a certain degree: The informal definition "A hydraulic axis realizes a subfunction of a hydraulic system." leaves a scope of interpretation—e.g., regarding the components which actually must be count to an axis and which not. Thus a precise definition of the hydraulic axes analysis problem cannot be stated.

The consequences are: (*i*) A human expert has the final say whether or not the result of an analysis algorithm is correct. (*ii*) The result of an analysis algorithm might not be absolutely correct or wrong, but it might be correct up to a certain degree.

We overcame the problems that result from this situation by acquiring (and encoding) analysis knowledge direct from domain experts, and by creating a library to prove the quality of our algorithms. This library contains a large range of hydraulic circuits from various engineering applications[1].

While the analysis problem cannot be defined exactly, the result of an analysis algorithm can:

- *Result of a Hydraulic Axes Analysis.* Starting point of each hydraulic axes analysis is a hydraulic circuit C. The result of the analysis are a set of numbers $\{1, \ldots, n\}$, denoting the n hydraulic axes found, a mapping from the components of C onto $\mathcal{P}(\{1, \ldots, n\})$, the power set[2] of $\{1, \ldots, n\}$, and a tree which defines the coupling hierarchy between the n axes.

2.2 Analysis Strategy

The analysis procedure that we have developed is comprised of the following three steps:

1. *Preprocessing.* The preprocessing step starts with an abstraction from a circuit C onto its related hydraulic graph G_h (G_h is defined in the next subsection). To reduce G_h's complexity—but, in first place, to make axes identification possible at all, G_h is simplified by means of merging, deletion, and condensation rules. Figure 2 illustrates the application of such rules. Section 3 develops the techniques that are necessary to tackle preprocessing.

2. *Axes Identification.* Identifying a hydraulic axis means to search for a set of nodes in the hydraulic graph whose counterpart in the circuit realizes a particular function. Each such set must contain at least one working element and one supply element. Moreover, all components that also belong to the hydraulic axis must lie on some path between the working and the supply element. This consideration forms the

[1] The internal report "Hydraulic Circuit Library" gives an overview of the current state of this library. It gives an idea of rather simple as well as of complex hydraulic circuits.

[2] A component can belong to several hydraulic axes.

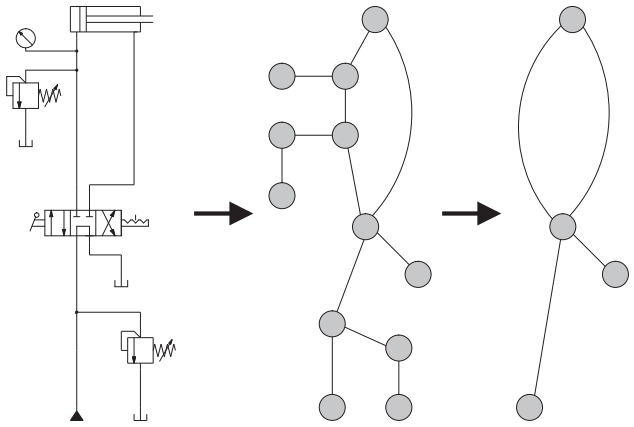

Figure 2: Example for the abstraction and simplification of a hydraulic circuit.

basis of our first identification approach, which relies on shortest-path-algorithms for the main part, and which is described in section 4.

Another approach founds on the idea of graph embedding: Given is a template of a hydraulic axis, that is to say, a graph of a particular topology. The task is to embed this graph within the original hydraulic graph. If such an embedding can be found, all nodes that are covered by the embedding belong to an instantiation of the template axis. Section 5 elaborates on the embedding strategy.

3. *Coupling Type Determination.* The type of coupling between hydraulic axes can only be determined, if all components of a circuit have been assigned to at least one axis. In most cases, coupling type determination requires the comparison of supply paths between the working elements of the axes. The last part of this section defines the different coupling types; section 6 shows how the coupling type can be determined when given two axes.

2.3 Graph-theoretical Foundations

The following definitions from [1], [2], and [4] are of importance with respect to the subsequent sections:

(*i*) A *multigraph* G is a triple $\langle V, E, g\rangle$ where $V, E \neq \emptyset$ are finite sets, $V \cap E = \emptyset$, and $g : E \to 2^V$ is a mapping with $2^V = \{U \,|\, U \subseteq V,\ |U| = 2\}$. V is called the set of points, E is called the set of edges, and g is called the incidence map.

To work with a hydraulic circuit C as an ordinary multigraph $G(V, E, g)$ a mapping rule is required. The related hydraulic graph $G_h(C)$ defines such a mapping for C.

(*ii*) A related *hydraulic graph* $G_h(C)$ of a circuit C is a multigraph $\langle V_C, E_C, g_C\rangle$ whose elements are defined as follows. (*i*) V_C is a set of points, and there is a bijective mapping from the set of non-pipe components in C onto V_C. (*ii*) E_C is a set of edges, and there is a bijective mapping from the set of pipe components in C onto E_C. (*iii*) $g : E_C \to 2^{V_C}$ is a function that maps an $e \in E_C$ onto $(v_i, v_j) \in 2^{V_C}$, if and only if there is a pipe between the components associated with v_i, v_j, and if e is associated with this pipe.

We need multigraphs instead of graphs since components of a hydraulic system may be connected in parallel. Figure 3 depicts a circuit and its related hydraulic graph.

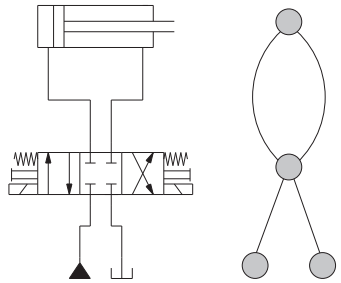

Figure 3: Circuit and its related hydraulic graph.

Remarks. For each circuit C there exists exactly one hydraulic graph $G_h(C)$. Notice the following topological simplifications of C: (*i*) Substructures within (directional) valves are comprised to one single point v, hence making all connected pipes incident to v. (*ii*) Variations of the topology coming along with valve switchings are neglected. (*iii*) Directional information that results from the behavior of particular hydraulic components is dropped.
These simplifications have no effect on the classification of hydraulic axes couplings.

(*iii*) A graph $H = \langle V_H, E_H, g_H \rangle$ will be called *subgraph* of $G = \langle V, E, g \rangle$, if $V_H \subseteq V$, $E_H \subseteq E$, and g_H is the restriction of g to E_H. A subgraph will be called an *induced subgraph* on V_H, if $E_H \subseteq E$ contains exactly those edges incident to the points in V_H. For $T \subset V$, $G \setminus T$ denotes the subgraph induced on $V \setminus T$. Figure 4 gives an example.

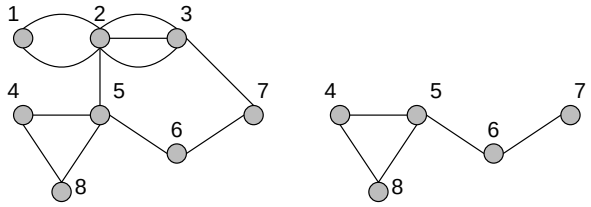

Figure 4: Graph and induced subgraph on the points $\{4, \ldots, 8\}$.

(*iv*) A tuple $(e_1, \ldots, e_n)$ will be called a *walk* from v_0 to v_n, if $g(e_i) = \{v_{i-1}, v_i\}$, $v_i \in V$, $i = 1, \ldots, n$. A walk will be called a *path*, if the v_i are mutually distinct. Instead of using a tuple of edges, a path may also be specified by a tuple of points, $(v_0, \ldots, v_n)$. Figure 5 gives an example.

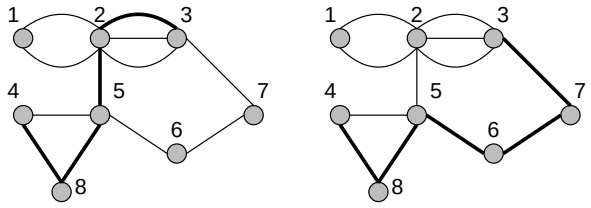

Figure 5: Paths in a graph.

(*v*) G will be called *connected*, if for each two points $v_i, v_j \in V$ there is a walk from v_i to v_j. If G is connected and $G \setminus v$ is not connected, v establishes an *articulation point*. The maximum connected subgraphs of G are called *connected components*.

A graph without an articulation point is called biconnected. A biconnected subgraph of a graph is called a *block*. Figure 6 and 7 illustrate the definition.

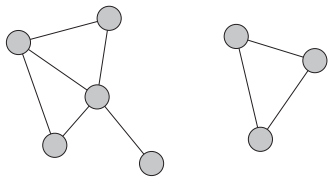

Figure 6: A graph consisting of two connected components.

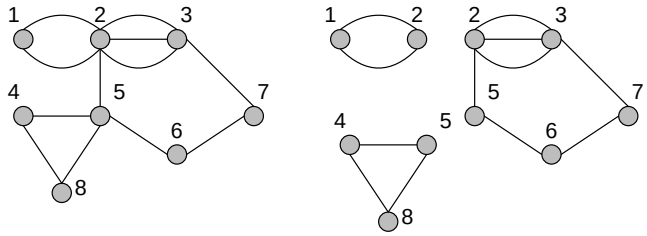

Figure 7: A graph and its blocks.

2.4 Graph-theoretical Formulations of Hydraulic Concepts

For the topological investigation of hydraulic circuits the variety of hydraulic components can be reduced to a small number of classes. We will agree upon the following classes:

- *Working Elements.* Components that act as output elements; all kinds of cylinders and motors belong to this class.
- *Control Elements.* Components that control the working elements; all directional valves belong to this class.
- *Supply Elements.* Components that provide the necessary energy in the form of pressure at flow; pumps are the only elements of this class.
- *Auxiliary Elements.* All elements which do not fall in one of the above classes make up the class of auxiliary elements; examples are tanks, pipes, hoses, or filters.

A hydraulic axis A both represents and fulfills a subfunction f of an entire hydraulic plant. A defines the connections and the interplay among those working, control, and supply elements that realize f. A consists of the element types listed above, while at least one working element and one control element is part of an axis.

Several hydraulic axes may share particular components. Figure 8 depicts a circuit consisting of two axes that share some components.

To accomplish complex manufacturing, control, or manipulation tasks, several hydraulic axes are coupled and play together. It is in the nature of things that the level of such a coupling can vary, from rather loosely coupled axes to axes that strongly depend on each other. In order to determine those components of a hydraulic system that belong to a particular axis A, couplings between A and other axes must be identified as such. A

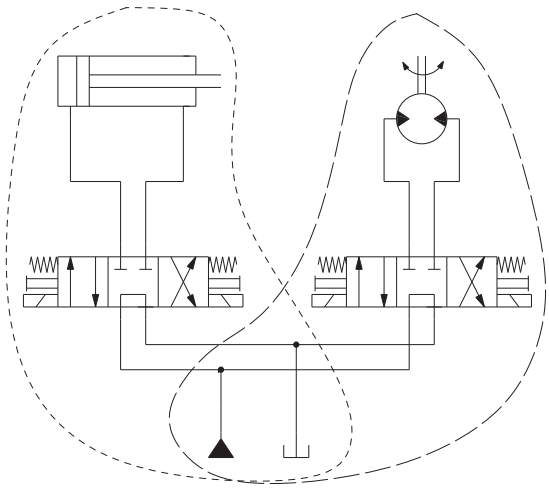

Figure 8: Two hydraulic axes that share some components.

prerequisite for the identification step thus is a classification of possible coupling types. We distinguish five different levels of couplings, which are defined in following.

Given is a hydraulic circuit C containing two sub-circuits A, B, which realize two different hydraulic axes. Moreover let $G_h(C) := \langle V_C, E_C, g_C \rangle$, $G_h(A) := \langle V_A, E_A, g_A \rangle$, and $G_h(B) := \langle V_B, E_B, g_B \rangle$ denote the related hydraulic graphs of C, A, and B respectively.

- *Level 0—No Coupling.* If $G_h(C)$ is not connected, and if $G_h(A)$ and $G_h(B)$ are subgraphs of different connected components in G_h, then the hydraulic axes A and B are not coupled.

  A and B don't have any physical connection, and thus they can be investigated independently.

- *Level 1—Informational Coupling.* Let $\{e_1, \ldots, e_n\}$ be in E and each e_i associated with a control line within C. If $G_{h'} := \langle V_C, E_C \setminus \{e_1, \ldots, e_n\}, g_C \rangle$ is not connected, and if $G_h(A)$ and $G_h(B)$ are subgraphs of different connected components in $G_{h'}$, then the hydraulic axes A and B are informationally coupled (cf. Figure 9).

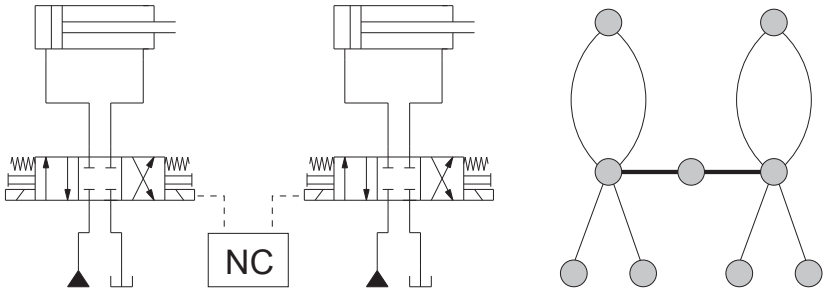

Figure 9: Circuit with informationally coupled hydraulic axes.

  Control lines can be realized by means of electrical, hydraulic, or pneumatic connections.

- *Level 2—Parallel Coupling.* Let $P_{w,s}$ be the set of all paths from a working element w to a supply element s that use no edge associated with a control line. Then A and

B are coupled in parallel if there exist two nodes, $v_a \in V_A$, $v_b \in V_B$, such that the following conditions hold:

(*i*) v_a, v_b are associated with a control element.

(*ii*) $\forall_{p \in P_{w,s}} : v_a \in p \cap V_A \sqcup v_b \in p \cap V_B$.

Figure 10 gives an example.

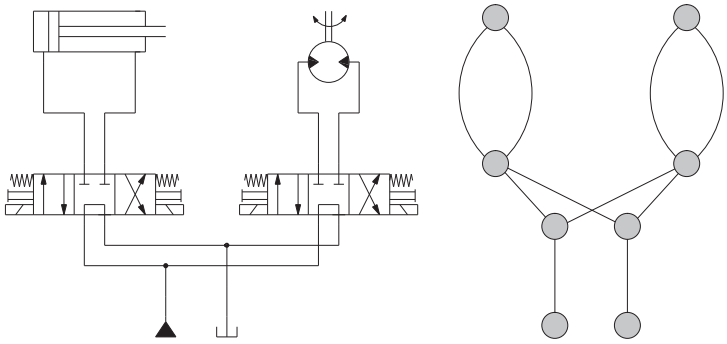

Figure 10: Circuit containing hydraulic axes coupled in parallel.

From an engineering viewpoint this definition states that each of the axes A and B is controlled by its own control element.

- *Level 3—Series Coupling.* Let $P_{w,s}$ be the set of all paths from a working element w to a supply element s that have no edge associated with a control line. Then A and B are coupled in series, if an axis $X \in \{A, B\}$ and a path $p \in P_{w,s}$ exist such that the following conditions hold:

  (*i*) p is a subgraph of X.

  (*ii*) $\exists v \in p \cap V_Y, Y \in \{A, B\} \wedge Y \neq X$: v is associated with a control element.

  Figure 11 gives an example.

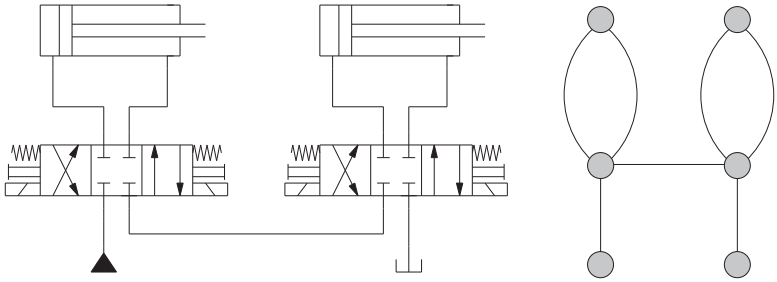

Figure 11: Circuit containing hydraulic axes coupled in series.

If several axes are coupled in series, at least one axis controls the flow of all other axes.

- *Level 4—Sequential Coupling.* A and B are sequentially coupled if the following conditions hold:

  (*i*) A and B have no coupling of type $0, \ldots, 3$.

  (*ii*) A and B establish no equal subcircuits of C.

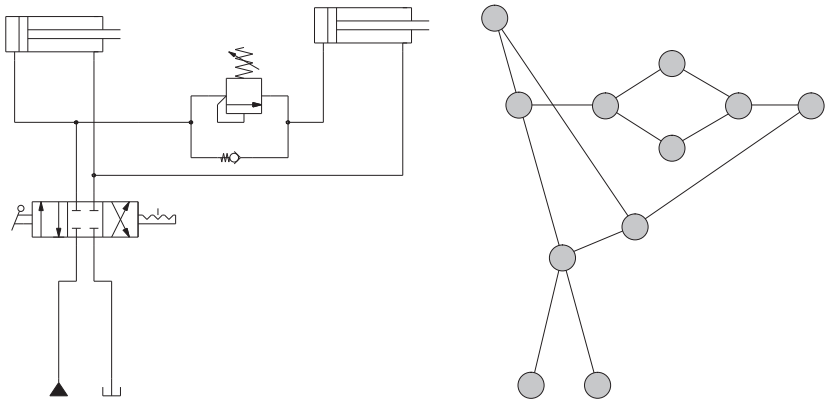

Figure 12: Circuit containing sequentially coupled hydraulic axes.

Figure 12 gives an example.

If A and B were equal subcircuits, their behavior would always be equal and they would together form one single hydraulic axis.

3 Preprocessing of Hydraulic Graphs

To realize the identification of hydraulic axes, it is necessary to reduce the complexity of the hydraulic graph. Rules that define the allowed simplification steps can be defined by means of graph grammars. Following some necessary definitions this section presents those graph simplification rules as well as auxiliary algorithms that are applied before the actual axes identification algorithm can be started.

3.1 Graph Grammars

The following definitions are necessary to introduce the concept of graph grammars. Backgrounds and related concepts can be found in [12], [6], and [7].

(*i*) *Graph Embedding.* An embedding of a graph $G_1 = \langle V_1, E_1 \rangle$ into a graph $G_2 = \langle V_2, E_2 \rangle$ is given by an injective mapping φ, $\varphi : V_1 \rightarrow V_2$. In most cases, adjacent nodes in G_1 will not become adjacent nodes in G_2 again. Then, for each edge $\{u, v\} \in E_1$ also a path $P_\varphi(u, v)$ must specified, which connects th nodes $\varphi(u)$ and $\varphi(v)$ in G_2. Note that this path must not be the shortest path between $\varphi(u)$ and $\varphi(v)$.

(*ii*) *Labeled Graph.* A labeled graph, also called *l-graph*, is a graph whose nodes $v \in V$ and edges $e \in E$ are labeled. According to [5], an l-graph on the alphabets Σ_V and Σ_E is a triple $G_l = \langle V, (\rho_a)_{a \in \Sigma_E}, \beta \rangle$ where

- V denotes a finite set of nodes.
- ρ_a defines a relation on V for arbitrary $a \in \Sigma_E$; i.e., $\rho_a \subseteq V \times V$. Here we assume that $\forall (u, v) \in \rho_a \Rightarrow (v, u) \in \rho_a$, i.e., ρ_a is reflexive, and G_l is a non-directed graph.
- β denotes a node labeling function.

Remarks. The alphabets Σ_V and Σ_E merely serve for labeling purposes. Note that the definition states that between two nodes several edges are allowed, if they are labeled differently. Also note that the node labels must no be unique. Thus it is possible to assign different nodes additional but equal information. Figure 13 shows an example for an l-graph.

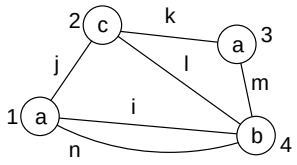

Figure 13: Example for an l-graph.

Graph grammars are a generalization of Chomsky-systems defined on strings. Similar to grammars that are defined over an alphabet, graphs grammars employ production rules to define transformation prescriptions.

(*iii*) *Graph Grammar.* A graph grammar is a set of rules. The left-hand side of a rule defines what is to be replaced, while the right-hand side defines the substitution. With respect to l-graphs such a rule specifies how a subgraph is substituted for another

subgraph. The rule does also specify in which way the subgraph of the right-hand side is embedded into the graph that shall be modified.

Consequently, a graph production (graph rule) is of the form (d_l, d_r, φ), d_l and d_r denote the left-hand side and the right-hand side respectively, and φ defines the embedding function. Figure 14 shows a graph production, φ is realized by means of the labeling.

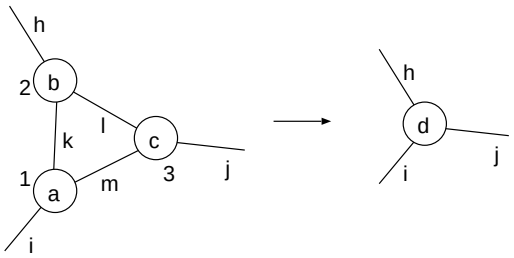

Figure 14: Example for a graph production.

Remarks. Our objective is not the derivation of new graphs starting with some initial graph (the start symbol) and applying the production rules. Instead the subsequent given graph grammar is for contraction and simplification purposes with respect to existing graphs.

3.2 A Graph Grammar for Hydraulic Graph Abstraction

The following graph grammar provides us with a collection of graph transformation rules for hydraulic graphs. The application of a rule is only allowed within a particular context.

Deletion of Non-essential Components

An obvious concept to reduce a hydraulic graph's complexity is the deletion of non-essential components [3]. Whether a component is non-essential depends on its context of use in many cases: A valve, for instance, which has been defined as control element (cf. section 2.4, page 6) may be used in an auxiliary position.

The identification and deletion of non-essential components is a special case of "Identification and Deletion of Sequences" treated below.

Substitution of Aggregate Components

Some components complicate the identification of hydraulic axes because they come along with a complex—but compilated—substructure. The power supply unit shown in Figure 15 is an example for such a component.

Such components do not comply with the path search strategies for the identification of hydraulic axes. A way out is the substitution of such a component by a new substructure with several simple components. In the case of the power supply unit the related graph grammar rule is depicted in Figure 16.

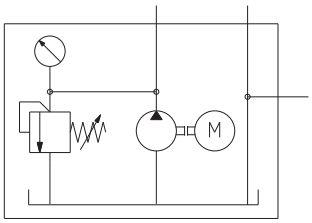

Figure 15: Power supply unit providing two tank connections.

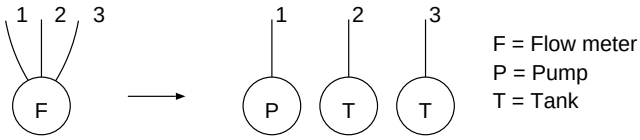

Figure 16: Substitution of an aggregate by other components.

Deletion of Control Edges

Control edges do not contribute to the detection of hydraulic axes: They are used to control the switching position of valves. Since the information on switching positions is lost in the abstracted hydraulic graph at all, control edges can be deleted in any case.

A large part of control edges can be identified and deleted easily, since the connections incident to control edges are explicitly labeled as control connections.

Comprising Tanks

Hydraulic circuits may contain a large number of tanks, leading to an overhead during the path search algorithms (cf. section 4). Thus we comprise all tanks of a circuit within a single new meta-tank.

If tanks have been used in different connected components, this transformation will connect the related subgraphs. Thus this transformation must be performed for each connected component on its own.

Identification and Deletion of Sequences

A sequence is a connected subgraph whose nodes have a degree of at most two. Sequences do not affect the topological structures that form the basis of our hydraulic axis definition. To cope with different contexts we distinguish between different types of sequences and deployed the following transformation rules:

- *Dead Branches.* Dead branches are sequences that contain one node of degree one. Usually dead branches provide no information regarding the identification of hydraulic axes. They can be deleted, if they don't contain a tank, a pump, or a working element, otherwise they can be shortened. Figure 17 gives examples of dead branches; Figure 18 shows the related production rules.
- *Sequences which Non-essential Components.* Sequences consist of non-essential components for the most part, but they may contain elements whose deletion is not

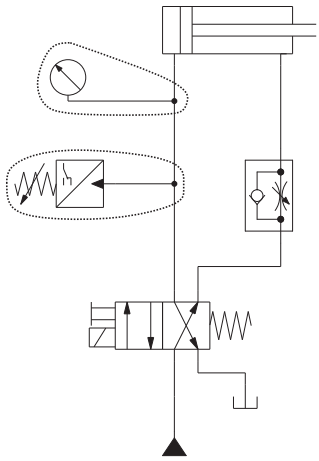

Figure 17: A circuit with dead branches.

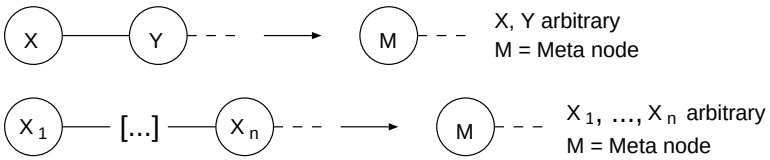

Figure 18: Rules for the deletion of dead branches.

allowed, such as tanks or working elements. The related production rules are shown in Figure 19.

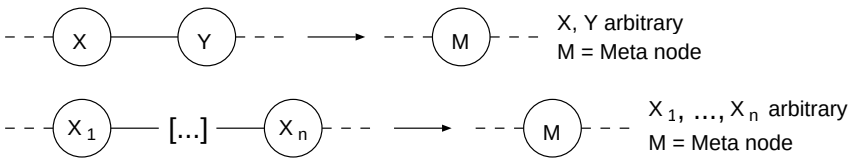

Figure 19: Rules for the deletion of non-essential components.

- *Consecutive Working Elements.* Consecutive working elements in sequences can be comprised to a single meta working element without affecting axes identification. Figure 20 gives an example for such a sequence.

Identification and Deletion of Cycles

Cycles that neither contain nor control working elements can be deleted without compromising the identification algorithms for hydraulic axes. Hydraulic circuits always contain cycles, and these cycles may contain many components. Figure 21 depicts some examples.

However, the detection of cycles can be restricted to the following cases:

- *Two Components are Connected in Parallel.* The restriction to this case is admissible, if the hydraulic graph does not contain sequences with more than one component, i.e., if the previously described steps have been performed.

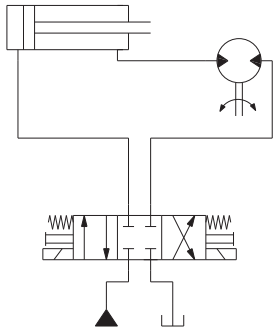

Figure 20: Consecutive working elements in a single axis.

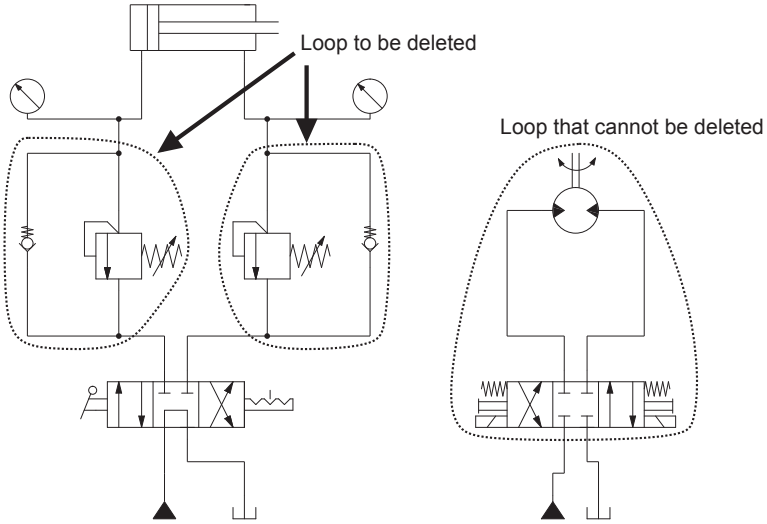

Figure 21: Cycles in a hydraulic circuit.

- *Two Edges are Connected in Parallel.* The application of this rules is possible, if sequences of the length one have been deleted.

Figure 22 shows the rules for circuit deletion. Note that cycles can be nested. To resolve nested cycles, the rules for sequence and cycle deletion must applied in a repetitive manner.

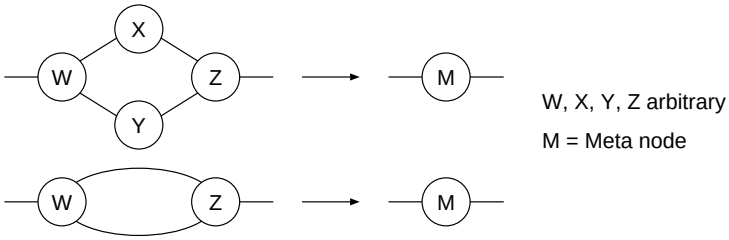

Figure 22: Rules for the deletion of cycles.

Merging T-junctions

Nodes in the hydraulic graph whose counterpart are T-junctions can be merged. Merging two such nodes results in a new meta node of the degree four. Figure 23 shows the rule.

More general, when merging two nodes u and v of the degrees j and h respectively, the resulting node will be of degree $j + h - 2$ if u and v were connected by a single edge.

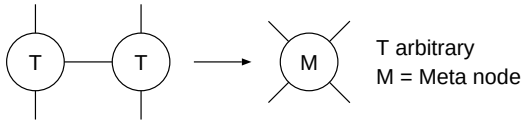

Figure 23: Rules for the merger of two T-connection nodes.

Figure 24 shows how this rule will also resolve cycles that consists of T-junction nodes only. Such cycles are the result of the application of the sequence and cycle deletion rules previously introduced.

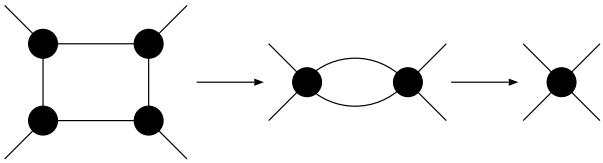

Figure 24: Deleting cycles of T-connection nodes.

Note that the application of the rule in Figure 23 can lead to difficulties during hydraulic axes identification. Section 3.3 will elaborate on that problem.

Order of Rule Application

An arbitrary application of the above rules would lead to a simplification of the hydraulic graph, but this simplification must not be optimum in any case. Examples where two different orders of rule application result in differently comprised graphs can be easily constructed.

A possible way out is a rule application based on a priority control: In a first step, each rule is tested if it can be fired; then among these rules that one with the highest priority is selected and executed.

Since the test whether or not a rule can fire is expensive, we decided to realize a fixed order of rule application. Of course such an order must consider the possible dependencies between the rules. A useful order is the following:

1. substitute aggregate components
2. delete control edges
3. comprise tanks
4. comprise consecutive working elements
5. delete dead branches
6. delete sequences of non-essential components
7. delete nodes of degree less then two
8. merge t-junctions

9. delete cycles
   If this rule has been fired, return to step 5.

3.3 Node Expansion

Note that the result of a hydraulic axes analysis is an assignment of each component to a hydraulic axis. Since also the deleted nodes must be assigned to some axis, each such node is associated with an *anchor node*, and the anchor node's axis will become the axis of the deleted node.

When deleting nodes of a degree more than one, there is the question which of the neighbored nodes will become the anchor node. A wrong choice will lead to an incorrect axis assignment of the component.

This indeterminism can be resolved by defining different *levels of attraction* for the different component types. The attraction level ϕ of the neighbored nodes then defines which of them will become the anchor node. This concept is powerful enough to model the analysis process of hydraulic experts in most cases. The order of attraction levels that we actually use is the following:

$$\phi(\text{auxiliary element}) < \phi(\text{control element}) < \phi(\text{supply element}) < \phi(\text{working element})$$

If a node to be deleted is between two nodes of the same attraction level, it will not become resolved.

3.4 Auxiliary Routines

This subsection provides auxiliary routines that are necessary to operationalize the production rules of our graph grammar. These routines will not perform structural modifications of the graph.

Identification of Blocks

The identification of cycles in a hydraulic circuit (cf. page 13) can be realized with a graph algorithm for the detection of strongly connected components or blocks.

Usually, a hydraulic graph will contain a lot of cycles from which only a subset shall be deleted. We can restrict the block identification algorithm to this subset by deleting the essential components (e.g. working elements) along with their incident edges in a first step.

The runtime complexity of an efficient block detection algorithm is $O(|E|)$ (cf. [4], [8]).

Edge Sorting

An important part of the algorithm for the hydraulic axes identification requires the detection of identical subgraphs during the application of the graph grammar rules. Note that the order of rule application depends on the order of a node's incident edges, and only a sorting of the edges of each node will ensure the same order of rule application for equal subcircuits.

Since each node is incident with at most four edges, the total sorting effort can be estimated by $O(|V|)$.

Adequate Data Structures

We use adjacency lists instead of adjacency matrices here, since the degree of the nodes in our graphs is rather small (< 5). A related adjacency matrix would have a small numbers of entries only.

A drawback of ordinary adjacency lists is that a direct access to their elements is not possible. We compensate this drawback by organizing these lists in the form of hash tables. Moreover, hash tables are also used to organize the nodes of a hydraulic graph.

4 Identifying Hydraulic Axes by Means of Path Search

In this place an algorithm for the identification of hydraulic axes is presented. This algorithm relies on path search strategies for the main part. The next but one subsection investigates the question how identical axes can be detected as such.

4.1 Path Search in the Hydraulic Graph

As mentioned earlier, each hydraulic axis must contain at least one working element and one supply element. Moreover, all components that also belong to the hydraulic axis must lie on some path between the working and the supply element. This consideration suggests to employ shortest-path algorithms to tackle the identification problem.

Efficient algorithms to solve shortest-path problems are given in [4] and [8]. Important representatives are the following:

- *Dijkstra's Algorithm.* Determines the shortest path from a designated node to all other nodes in a weighted graph. Its runtime complexity is $O(|V^2|)$ and $O(|E| \cdot log(|V|))$ when using a linear list and a heap data structure respectively.
- *Floyd's Algorithm.* This algorithm determines the distance between any pair of nodes of a graph. It follows the idea of dynamic programming and has a time complexity of $O(|V^3|)$ and a space complexity of $O(|V^2|)$.

To identify hydraulic axes, all paths between the supply elements and the working elements of a circuit must be investigated. Hence a shortest-path problem must be solved for each supply element. Each run of a shortest-path algorithm labels the edges in the form of a directed tree, encoding the successor relationship between two nodes (cf. Figure 25).

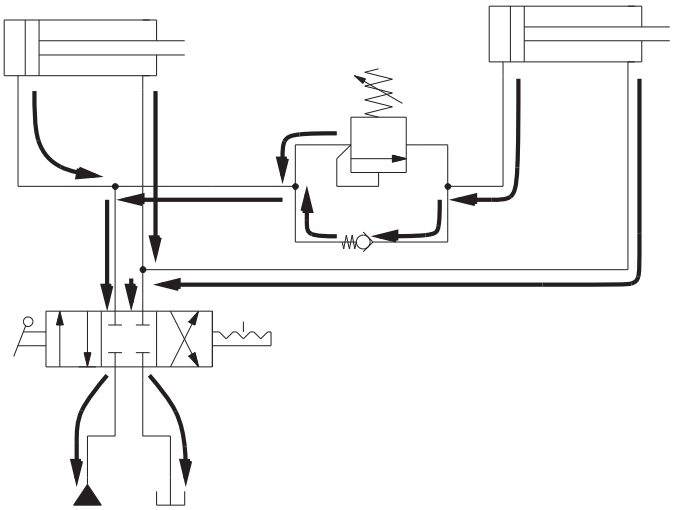

Figure 25: Circuit with successor information after a shortest-path run.

Note that all components that lie on the same path in the directed tree belong to the same hydraulic axis. Since hydraulic graphs are multigraphs there must exist two different paths from a working element to a supply element. A second path can be found by simply deleting one edge incident to the working element and then applying the shortest-path algorithm again.

Each hydraulic axis is also connected with a tank, and the components lying on the path between the tank and the working element are also count to the hydraulic axis. Hence we apply the path search algorithm in the same way for tanks as we did for the supply elements. These considerations lead to the following algorithm for the detection of all hydraulic axes. Given is a hydraulic graph $G_h(C) = \langle V_C, E_C, g_C \rangle$ of a circuit C.

1. For each supply element $s \in V_C$ do:
   (a) Compute the shortest paths from s to the working elements.
   (b) Assign all nodes of the same path the same hydraulic axis number.
   (c) Remove one edge incident to each working element.
   (d) Compute the shortest paths from s to the working elements.
   (e) Assign all nodes of the same path the same hydraulic axis number.
2. For each tank $t \in V_C$ do:
   (a) Compute the shortest paths from t to the working elements.
   (b) Assign all nodes of the same path the same hydraulic axis number.
   (c) Remove one edge incident to each working element.
   (d) Compute the shortest paths from t to the working elements.
   (e) Assign all nodes of the same path the same hydraulic axis number.

The worst case complexity for this algorithm is $O(|V|^2 \cdot |E|)$.

4.2 Checking for Identical Axes

It is necessary to detect identical axes as such since they must be comprised to one single axis when analyzing the circuit. Identical axes are composed from the same components, they have an equivalent structure, and they are controlled by a single control element. Figure 26 gives an example for identical axes.

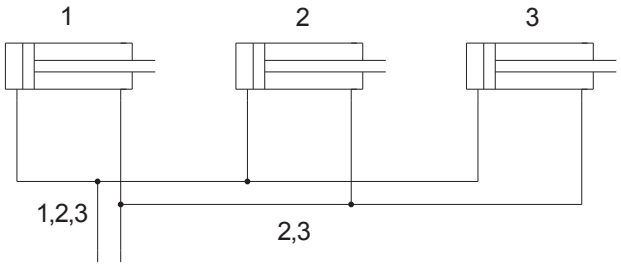

Figure 26: Subcircuit with three identical axes.

The above characterization of identical axes shows that we have to compare both two axes' components and their structures to check whether they are identical. However, with respect to T-junctions and pipes we must be flexible: These components are allowed to occur in different numbers and positions.—Putting it all together:

- The comparison between two axes must happen within the original, i.e., within the uncompressed hydraulic graph.

- Two axes can be identical without having the same number of components.
- The comparison of structures must cope with structures whose topology is different, but which produce the same behavior.

Checking for Candidate Pairs

Having passed the preprocessing phase, identical axes are still identical and, moreover, they are simplified. Thus we can restrict the identical axes investigation to a subset of all axes. As well as that, when working on the compressed axes we can exploit the successor relationship between the nodes, which has been established during the shortest-path search.

If the interesting circuit has k axes, then we have to compare $\binom{k}{2} \in O(k^2)$ possible pairs of axes. The comparison of a pair will take linear time since we simply walk along the paths that are defined by the successor relationship. Thus we have a total time complexity of $O(|V|^2 \cdot |E|)$ for this step.

Checking by Means of Step-wise Expansion

Among the candidate pairs we continue our investigation as follows. Nodes that have been deleted during preprocessing are stepwisely inserted and compared. This strategy presumes that the deletion of nodes has been happened in a definite manner—a requirement that is fulfilled since the preprocessing algorithms work on sorted data structures. Consequently, nodes of equal structures are deleted in the same succession, and the deleted nodes are pushed on a stack.

This strategy must be refined to cope with several exceptions since nodes may become incomparable: E. g., the type of a node can be changed when deleting one of its neighbors, or an axis contains components of the same type.

5 Identifying Hydraulic Axes by Means of Embedding

An alternative strategy for the detection of hydraulic axes is template embedding. This section introduces the concepts and points out related difficulties.

5.1 Structure Mapping in the Hydraulic Graph

As introduced in section 3.1, an embedding defines a mapping from the nodes of a graph G_1 onto the nodes of a graph G_2; the edges of G_1 are mapped onto paths in G_2. A *template* embedding is defined as follows.

- *Template Embedding.* A template embedding is an embedding of an abstract graph T, called the template, into another graph G. $\phi(v) = w, v \in T, w \in G$, if and only if $\psi(v) = \psi(w)$ for some *suitability function* ψ.

I. e., a template embedding is an embedding that has to fulfill some given constraints. In connection with hydraulic graphs a typical constraint is the following: "A node v of the template can only be mapped onto a node w in the hydraulic graph, if v and w are of the same type." Figure 27 shows a template and a graph, and Figure 28 again shows this template along with its embedding.

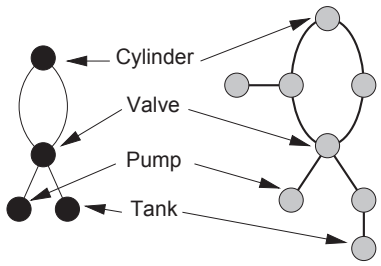

Figure 27: A template and a graph.

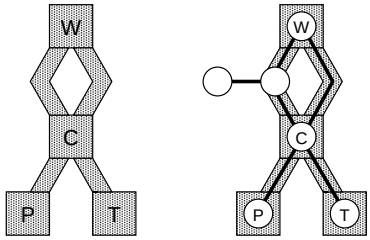

Figure 28: A template along with an embedding in a graph.

Figure 28 shows that the hydraulic graph must be compressed (preprocessed) to make an embedding possible: Dead branches or cycles complicate or may inhibit an embedding.

It becomes clear that template embedding is of another nature than the production rules of a graph grammar. The latter have been used to find small subgraphs, so to speak, "local embeddings". Template embeddings, on the other hand, aim at the entire graph. Note that by each template embedding in the comprised graph exactly one hydraulic axis is detected.

5.2 Developing a Suitability Function for Hydraulic Axes

In the following we develop step by step a method for the identification of hydraulic axes by means of template embedding.

We start with the task to embed a graph $G_1 = \langle V_1, E_1 \rangle$ in a graph $G_2 = \langle V_2, E_2 \rangle$ without imposing any embedding constraints. To achieve a valid embedding, it must be ensured that all nodes and edges of G_1 are correctly mapped on G_2. While a node mapping can be determined immediately, the edge mapping requires some particular search effort. Hence the necessary steps for a primitive embedding are as follows.

1. *Connected Components.* Determination of the connected components of G_2. Note that all images $\phi(v)$ of the nodes in V_1 must lie in the same connected component of G_2. The runtime complexity for this step is $O(|E_2|)$.
2. *Node Mapping.* Within this step merely the node lists of G_1 and G_2 must be traveled until all nodes in V_1 are assigned. The runtime complexity without mapping constraints is $O(|V_1|)$.
3. *Path Search.* A path search has to be performed for each pair of nodes that is connected by an edge in G_1. The search can be realized by a depth-first search or a breadth-first search. The runtime complexity is $O(|V_2| \cdot \log(|V_2|) + |E_2|)$ if a minimum-spanning-tree algorithm along with a Fibonacci-heap is employed (cf. [4]).

Figure 29 shows a primitive embedding.

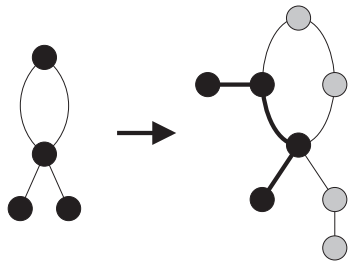

Figure 29: A primitive embedding.

Within a next step we claim that all nodes of G_1 must be mapped onto nodes of the same degree in G_2. Such a mapping (the step 2) can also be realized in $O(\max\{|V_1|, |V_2|\})$, if the nodes of G_2 are sorted relating their degree. Figure 30 shows an example for such an embedding.

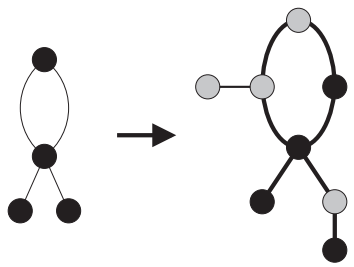

Figure 30: Embedding that is compliant with the node degree constraint.

The node degree constraint does not imply that actually all edges of an image node in G_2 are used when mapping the edges. Thus we further constrain the embedding by

claiming this restriction. Now the paths between the nodes cannot be determined by the minimum-spanning-tree algorithm—we will use the shortest-path algorithm of Dijkstra instead instead. We force the shortest-path algorithm to use all edges of a start node v by temporarily removing the first edge incident to v after each shortest-path run. The total runtime complexity then is $O(|V_2| \cdot \log(|V_2|) + |E_2| + k^2 \cdot (|E_2| \cdot \log(|V_2|)))$, $k \leq |V_1|$. Figure 31 shows a possible embedding.

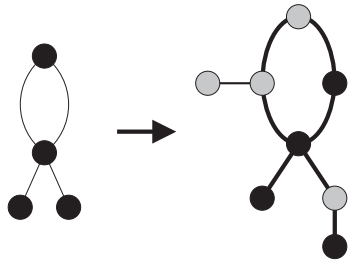

Figure 31: Embedding that is compliant with the edge mapping constraint.

In a last step we further tighten up the node embedding restriction: A node $v \in V_1$ is only allowed to be mapped onto a node $w \in V_2$, if v and w are of the same component type. The consideration of this restriction does not worsen the runtime complexity; it can be checked along with the node degree constraint. Figure 32 shows how the component type restriction for nodes comes to effect.

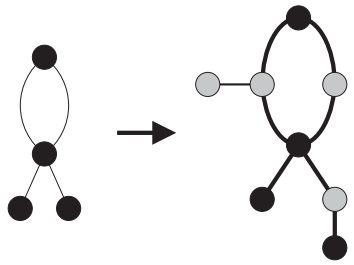

Figure 32: Embedding that is compliant with the component type restriction.

6 Couplings Between Hydraulic Axes

This section describes how the coupling level between hydraulic axes can be determined. Basically, methods similar to those for the identification of hydraulic axes are employed.

6.1 Coupling Between Two Adjacent Hydraulic Axes

The coupling levels as introduced in subsection 2.4 have very different properties, and different efforts are needed for their identification. The following paragraphs investigate this effort, whereas it is assumed that exactly two axes are given.

Couplings of Level 0 and Level 1

The identification of these coupling types is based on the concept of connected components. Axes "connected" by coupling level 0 lie in different connected components in the original graph.

Axes connected by coupling level of 1 lie in different connected components after all control edges have been deleted, which can be achieved within $O(|E|)$ steps.

Couplings of Level 2 and Level 3

Given a coupling of level 2, i.e. a parallel coupling, both the supply element and the tank is used for all axes, whereas each axis's control element has a direct access to them. When given a coupling of level 3 only the control element of one axis has a direct to access to the supply element and the tank. Figure 33 illustrates the latter case.

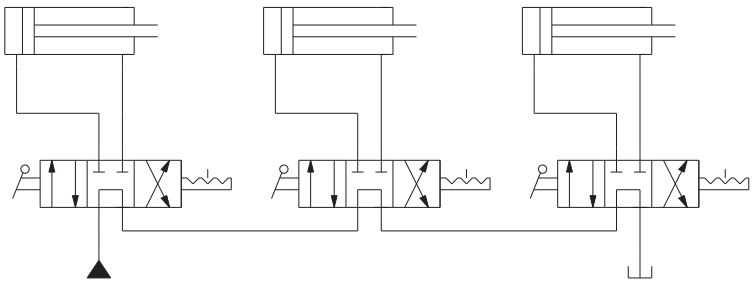

Figure 33: Three hydraulic axes coupled in series..

To identify both cases we follow up the paths that connect the working elements with the same pump and tank respectively. In the case of a level 2 coupling each such path contains only control elements of one axis; in the case of a level 3 coupling all but one path does contain control elements of at least two axis. Since we don't need the shortest paths between the components, the search effort can be estimated with $Q(|E|)$.

Level 4 Coupling

The identification of this coupling level is similar to the identification of identical hydraulic axes. In both cases (two axes are identical and two axes coupled by level 4) there is only one control element. In the latter case one of the axes does contain an additional behavior-modifying component, such as a hydraulic resistor (cf. Figure 34).

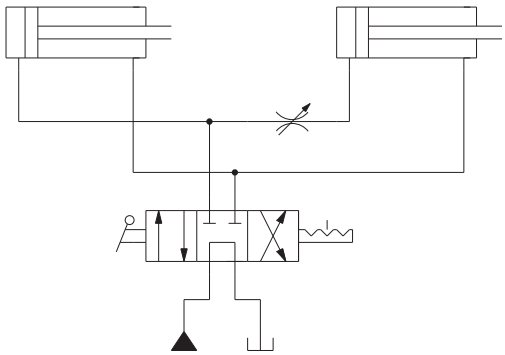

Figure 34: Sequentially coupled axes..

We compare all paths from the working elements to the supply element til we find a shared component. Up to that point no control element must be encountered, otherwise the axes are not coupled sequentially. If no control element has been encountered, the elements of the path are checked if they contain at least one behavior-modifying component (cf. Figure 35). If so, the axes are sequentially coupled.

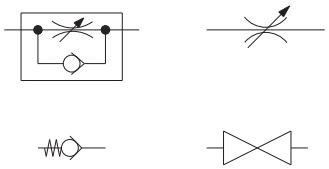

Figure 35: Examples for behavior-modifying components..

6.2 Transitivity of Couplings

Given a circuit with n axes, normally the couplings between *all* axes need to be determined. Using a naive approach for this job, the above search effort is carried out $\binom{n}{2} \in O(n^2)$ times. If, for example, a circuit contains a lot of axes of only one coupling type, a linear number of comparisons is sufficient.

In this connection some kind of transitivity property for coupling types would be useful. Figure 36 shows a circuit where three axes are coupled, and having classified two of the three couplings, the third coupling type can be inferred.

Figure 37 shows that, in generality, such a transitivity cannot exist.

However, given three axes and information on two coupling types, we are able to restrict the third coupling to a subset of all types: Let the known coupling types be a and b, $a, b \in \{0, \ldots, 4\}$. Then for the third coupling c the following holds: $c \geq \min\{a, b\}$. Stated another way, a weaker coupling cannot be possible since the axes are coupled indirectly via the third axis. This property can be exploited to reduce the complexity of the coupling type determination.

6.3 Hierarchy of Couplings

Hydraulic axes in complex circuits are often organized hierarchically (cf. Figure 38).

The identification of the hierarchical organization of the axes will also save comparison effort for the coupling type determination. The leafs of the coupling tree correspond to the

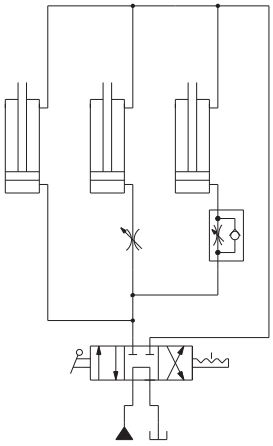

Figure 36: Equally coupled axes..

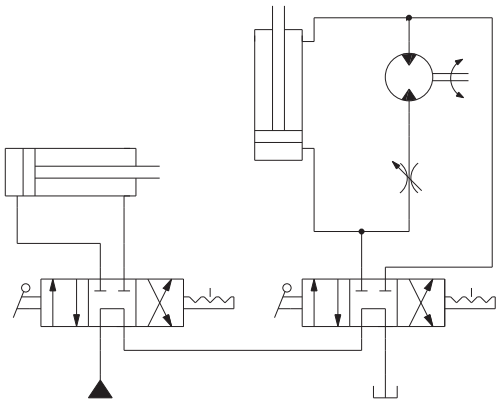

Figure 37: Sequentially coupled axes..

actual axes, inner nodes correspond to so-called meta axes (cf. Figure 39). The coupling between two meta axes defines the coupling between all those axes that do not stem from the same meta axis.

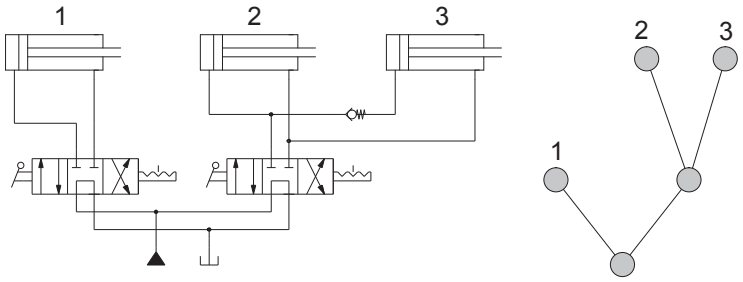

Figure 38: Circuit with abstracted coupling tree..

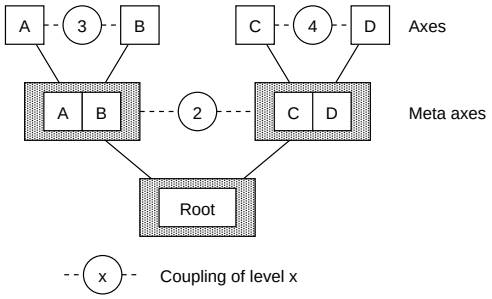

Figure 39: Abstracted coupling tree with meta axes..

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge. Massachusetts, 1990.

[2] D. Jungnickel. *Graphen, Netzwerke und Algorithmen*. BI Wissenschaftsverlag, Wien, 1990.

[3] D. Kimelman. Reduction of Visual Complexity in Dynamic Graphs. Technical report, IBM Thomas J. Watson Research Center, 1994.

[4] J. McHugh. *Algorithmic Graph Theory*. Prentice Hall, 1990.

[5] M. Nagl. *Graph-Grammatiken: Theorie, Anwendungen, Implementierung*. Vieweg Verlag, 1979.

[6] A. L. Rosenberg. Issues in the Study of Graph Embedding. Technical report, IBM Thomas J. Watson Research Center, 1994.

[7] H. J. Schneider. On Categorical Graph Grammars Integrating Structural Transformations and Operations on Labels. *Theoretical Computer Science*, 109:257–274, 1993.

[8] R. Sedgewick. *Algorithms in C++*. Addison-Wesley Publishing Company, 1992.

[9] B. Stein. Optimized Design of Fluidic Drives—Objectives and Concepts. Technical Report tr-ri-97-189, University of Paderborn, Department of Mathematics and Computer Science, Aug. 1996.

[10] B. Stein and E. Vier. Computer-aided Control Systems Design for Hydraulic Drives. *Proc. CACSD 97, Gent*, Apr. 1997.

[11] E. Vier. Rechnergestützter Entwurf geregelter fluidischer Antriebe. Technical Report MSRT 3/96, Gerhard-Mercator-Universität - GH Duisburg, MSRT, 1996.

[12] K. Wagner and R. Bodendiek. *Graphentheorie: Zahlen, Gruppen, Einbettungen von Graphen und Geschichte der Graphentheorie*, volume 3. BI Wissenschaftsverlag, 1993.