Towards Classifying Reactions of SBML-Models

Bachelorarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik

vorgelegt von:	Fabienne Lambusch
Gutachter:	Dr. Christian Rosenke
Zweitgutachter:	Dipl.-Inf. Ron Henkel
Abgabedatum:	07. April 2015

Contents

Abstract

Biological processes are often described by models. With the increasing amount of such biological models, researchers encounter the question how similar these various models are and whether it is possible to group them according to their features. The complexity of the question leads to the problem that only partial solutions exist until now. In addition, manually studying similarities of a large set of complex models is barely feasible. This thesis examines automatic methods to find structural similarities of models in a graph representation with respect to the biological background. Thus, a simple element-wise comparison of the graph structures is assumed to be inappropriate. Instead, an approach is chosen to automatically find the most frequent structures within the models' components. Appropriate patterns were found, which occur in more than a half of the 445 input models. The occurrences of the resulting structures can serve as a reasonable similarity measure for grouping the models that share many common structures. By laying the foundation to group models, also the requisite preconditions are established for an application like browsing through a group of models.

List of Figures

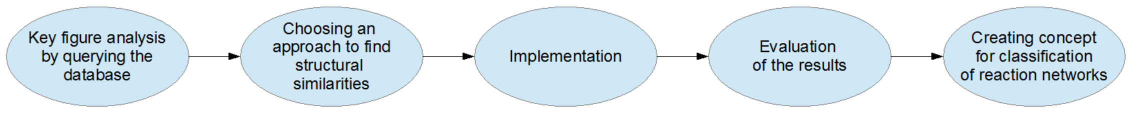
List of Tables

1 Introduction

Biologists use models to describe complex biological systems on an abstract level. With the increasing amount of designed models, researchers are interested in detecting models that are relevant to particular issues. Thereby, one important aspect is the knowledge about the similarity of the occurring components in models and their relations. The increasing amount of data makes the manual analysis of the models' structures barely feasible. As a consequence, appropriate automatic methods for this purpose must be chosen, which can provide the desired information. The analysis of various models in their entirety became possible through the recent research for standardising the model encoding and storing biological models centrally. Public repositories, such as the BioModels Database, provide access to models with their corresponding publications (Li et al., 2010). But a structure analysis of models stored in the BioModels Database has not yet been carried out. For a model database called KEGG, structural similarities of model components have already been examined (Koyutürk et al., 2004; Hattori et al., 2003), but the models differ from those in the BioModels Database. Section 2 describes examples for research in the field of finding similarities in biological models. The aim of this thesis is to elaborate automatic methods to identify structural similarities of models in the BioModels Database. With the obtained results it becomes possible to assign the models to groups according to the similarity of their structure. This process of assignment subsequently is called the classification of models (Section 3.5). A use case of classification is the organisation of models, such that researchers can easily browse through a group of models that is related to their work.
In the field of biology, the description of models can have various shapes. Therefore, standards for encoding models were developed, such as the modelling languages CellML (Lloyd et al., 2004), NeuroML (Gleeson et al., 2010) and SBML (Hucka et al., 2003). To reduce the complexity of finding structural similarities, this thesis only considers SBML encoded models. Li et al. (2010) states, that SBML - the Systems Biology Markup Language - "[...] has so far been the most successful standard model exchange format in this field". SBML is a common XML-based language for encoding models that represent biochemical reaction networks (Hucka et al., 2003). This representation of models illustrates meshing reactions that are responsible for a certain process within complex biological systems. The participants of reactions are called species. For instance, a possible reaction would be the well-known synthesis of hydrogen and oxygen into water. Several reactions together built networks, if they share some species. In Section 3.1 the used models and their representation are explained.
They are stored in a database, which uses graphs to represent its data (Henkel et al., 2014) (Section 3.2). Therefore, entities are presented as vertices and their relationships as edges between them. A graph in this database, according to the above example, would then be a vertex for the reaction that is connected to vertices representing the participating species - hydrogen, oxygen and water. Although SBML-models can have some more entities, in this thesis only the reaction networks are considered for a start. By focusing just on this graph representation, the task of searching structural similarities of reaction networks is transformed into problems of graph theory. To create a reasonable similarity measure for the models' networks represented as graphs, it is essential to regard the network structure as a whole, rather than treating them as a set of vertices and edges. Lakshmi and Meyyappan (2012) state that the simple pairwise comparison of nodes and edges within a network neglects its structure, whereas it is possible to respect the composition of network elements by viewing the graphs as similar, if they share

many common substructures. Consequentially, the problem of detecting structural similarities within the models is defined as a frequent subgraph mining task. Frequent Subgraph Mining (abbrv. FSM) is the problem to find frequent occurring structures within graphs (Keyvanpour and Azizani, 2012). A difficulty is that FSM algorithms require subgraph isomorphism testing - the problem to decide whether a graph is embedded in another (Lakshmi and Meyyappan, 2012). This is known as an NP-complete task (Keyvanpour and Azizani, 2012). Thus, FSM techniques still require heuristics, prior knowledge or another particular strategy to improve the performance. Various FSM algorithms with several priorities have been developed so far and that is why choosing an algorithm appropriate for the use case is an important element of this thesis. Figure 1 shows the outline of this thesis.

Figure 1: Planned workflow


Key figure analysis by querying the database
Choosing an approach to find structural similarities
Implementation
Evaluation of the results
Creating concept for classification of reaction networks

The figure shows how the proceeding for this thesis is outlined

I started with a key figure analysis for a general overview about quantities of the models' components. The results are presented in Section 5.1. The analysis revealed that most reactions have a maximum of up to three participatory species, while most species in turn take part in up to three reactions. Moreover, the major part of models contain less than 30 reactions and species. On this basis, I chose an algorithm to automatically find structural similarities of the models' networks. Section 3.3 explains the different approaches, which were considered to get structure information. The decision was made for the Frequent Subgraph Mining approach. Section 3.4 shows an overview of FSM algorithms and describes the applied algorithm gSpan (Yan and Han, 2002), while Section 4 describes the technical details of the implementation. Tyson and Novák (2010) carry out functional patterns of activation and inhibition referring to the living cell (Section 2.2). Finding functional patterns as described there, would be a great achievement to show the capability and accuracy of such an automatic working procedure for mining biologically precious patterns and thus, could offer the possibility to find so far undiscovered functional structures. In Section 5.2 the results of the mining are described. Section 6 evaluates the results from the computer science point of view and describes tasks for future work.
In conclusion, the FSM algorithm gSpan is applicable to find structural similarities within a set of SBML-models and can ease the work for researchers in this field by automatically performing the mining of patterns. In this thesis valuable patterns were found, whereas a biological interpretation of the results by a domain expert is still necessary. On the basis of the results, very useful applications can be implemented. For example, a function to search structural similar models could be established as well as a recommender system that suggests suitable structures, while a researcher is modelling a process.

2 State Of The Art

The problem of searching similarities between biological models is already examined, but with different focuses. Section 2.1 shows studies of the similarities between models according to semantic annotations. Semantic annotations add external knowledge to model parts by linking them to ontologies (Alm et al., 2014). These ontologies consist of terms defining entities in a domain and semantic links specifying the relations between the entities (Robinson and Bauer, 2011). Thus, the studies for similarities according to semantic annotations do not deal with the structure of model networks, but with the meaning of their components. Nevertheless, they are related to the topic of this thesis by proposing approaches for extracting features that can be used to classify models of the BioModels database.
Section 2.2 describes a paper, in which functional patterns of network structures are discovered and Section 2.3 describes approaches for finding frequent patterns in biological networks. Occurrences of such patterns in model networks can serve as a significant criteria for similarities and thus, their examination is useful for the aim of this thesis. Terms used in the following, such as the network structure, are explained more detailed in Section 3.

2.1 Similarity Of Biological Models According To Semantic Annotations

Schulz et al. (2011) examine a similarity measure for biological models according to their semantic annotations, but ignore the network structure. Annotations give additional information for the models and link them to web resource entries. On this basis various annotations are combined in their work to form a new single ontology. This is mainly because the entries can have the same meaning, but are stored in different resources. Another possibility is that entries describe similar entities - for example one species that is a special case of another - but are not related. By the integration of various annotations, they become comparable by means of semantic text analysis. In the paper two different types of similarity measures for semantic annotations are considered. One is based on feature vectors, where there is one feature vector per model established, which contains entries with the number of occurrences of all regarded annotations. The other group of measure is structure-based and is computed by pairwise comparing annotation elements.
Then, an evaluation of the two approaches for measurements is done by comparing a manual classification of the models with an automatic clustering according to the different measures. Afterwards, the measure based on feature vectors is considered more suitable. On the basis of their results, the authors implemented a platform, where the main application is a search for relevant models in the BioModels database. The searched models are ranked according to the semantic similarity with the input data set. Further applications of the platform are the clustering of models by their similarity of annotations and the visual alignment of their elements - the matching of equivalent elements from two models.
Schulz et al. (2012) developed a heuristic to infer missing annotations in partially annotated biological models. Their method, called "semantic propagation", is implemented for SBML-encoded models. It provides the possibility to propagate feature vectors with entries for each annotation. These can be used for the comparison of the models' similarity. The idea is to establish a similarity measure that is not only based on annotations of single model elements, but on the annotations inferred from the network structure. Additional information can be retrieved for a model element, if it is connected to other elements. Two reactions then can be compared

- even, if their annotations are missing - by evaluating the annotations of their participating species. The propagation of semantic information is done simultaneously for all elements with an initial direct similarity, where the inferred similarity becomes integrated. Applications of the approach are an alignment of partial model annotations as well as a similarity search based on the features propagated. Furthermore, the authors state that, because the focus is on annotations, the revealed similarity measure could be enriched by combination with a similarity measure focusing on the network structure.
Alm et al. (2014) examine four annotation-based methods to extract characteristic features of biological models and evaluate their applicability to determine similarities between models. SBML-encoded models from BioModels Database are stored in a graph database for further studies. Four different model sets are used, whereof annotations from three frequent ontologies are considered. Four methods for the feature extraction are analysed and three of them are found appropriate. These methods are based on techniques of clustering and text classification. The method considered most suitable is a bottom-up clustering based on the semantic annotations of the models, which further utilises the information content of the referenced ontologies. The extracted features can serve as a similarity measure to classify or compare biological models as well as to be used for retrieval tasks.

2.2 Functional Patterns In Cellular Networks

Tyson and Novák (2010) take into account structures of networks, but only deal with models of the living cell and focus on the function of the structures. By using their expert knowledge, they find that the complex networks of the information processing within cellular models can be decomposed into simple patterns, which fulfil a certain function within a cell. For this purpose, they first search for possible structures of such pattern. Then, assigning potential processing roles for the identified motifs is studied. Furthermore, it is analysed whether such resulting pattern serve as significant modules in the information-processing of the cell and act as expected in real biochemical networks. In the paper basic motifs are found that fulfil these expectations. They are contained in regulatory networks of living cells. Nevertheless, the information processing within cells is not yet completely understood and the interaction of various functional modules requires further examination.
This thesis can take advantage of the finding, that complex networks can be decomposed into simple functional patterns. The occurrences of certain patterns in models' networks may serve as a reasonable similarity measure for the network structure of the considered models.

2.3 Detecting Biological Network Patterns

Similar to the approach of the last Section 2.2, the following papers take into account the structure of models' networks by means of occurring patterns. But the focus here is on the frequency of occurrences instead of mainly considering the function. Furthermore, the presented concepts can be processed automatically, while analysing the function of patterns requires knowledge of an domain expert. As mentioned above, it is assumed that occurrences of certain patterns in models' networks may serve as a reasonable similarity measure between network structures.
Wong et al. (2011) discuss the motivation to find frequent occurring patterns within biological networks and ask whether there is a correlation between the functional behaviour of such pat-

terns with their structural topology. The authors present several existing algorithms for this purpose. The algorithms are evaluated by experimental results, classified according to several characteristics, and their advantages as well as disadvantages are discussed.
There are already applications of finding frequent patterns within metabolic pathways. Koyutürk et al. (2004) define metabolic pathways as "chains of reactions linked to each other by chemical compounds (metabolites) through product-substrate relationships." Their paper deals with metabolic pathways in the KEGG database. KEGG is the abbreviation for the Kyoto Encyclopaedia of Genes and Genomes (Fionda and Palopoli, 2011). It mainly contains static models that describe metabolic pathways and thus, the models differ from the once used in this thesis. That is the reason why there is still a need for a structure analysis for models of the BioModels database. Nevertheless, the paper demonstrates a possibility to find structural similarities within biological models, which is also a task of this thesis.
Koyutürk et al. (2004) search for frequent subgraphs within a set of metabolic pathways in the KEGG database, where the pathways are represented as directed graphs with unique node labellings. The authors state, that their approach is also applicable to various other biological networks with only minor modifications at the most. They reduce the computational cost for the algorithm by making use of the sparse nature of metabolic pathways and unique node labelling. Their approach aims to discover common patterns of related enzyme interactions. This thesis takes advantage of frequent subgraph mining as well (Section 3.4), but with a focus on SBML-models of the BioModels database.

3 Methods

The aim of this thesis is to find significant structures of reaction networks within a set of SBML-models, which are stored in a graph database. The next Section 3.1 describes the original source and structure of these models, while Section 3.2 provides information about the storage and possible querying. Different approaches for mining structural information have been considered according to their advantages and disadvantages, which are explained in Section 3.3. Section 3.4 shows an overview about common algorithms of the chosen approach that is called Frequent Subgraph Mining. Furthermore, it describes the applied algorithm gSpan as well as the used implementation. Section 3.5 elucidates classification.

3.1 Data

The used set of models originally comes from BioModels Database release 26[1], whereof only the curated branch - that means the verified models - is extracted. BioModels Database is a public repository that provides access to models with their corresponding publications (Li et al., 2010). There are different standard encodings for models, but in this thesis only SBML-encoded models are considered. SBML is the abbreviation for the Systems Biology Markup Language, which is a common XML-based language for encoding models that represent biochemical reaction networks (Hucka et al., 2003). The reaction networks of SBML-models can be expressed as graph structures as one can see in Figure 2.

[1]ftp://ftp.ebi.ac.uk/pub/databases/biomodels/releases/2013-11-04/

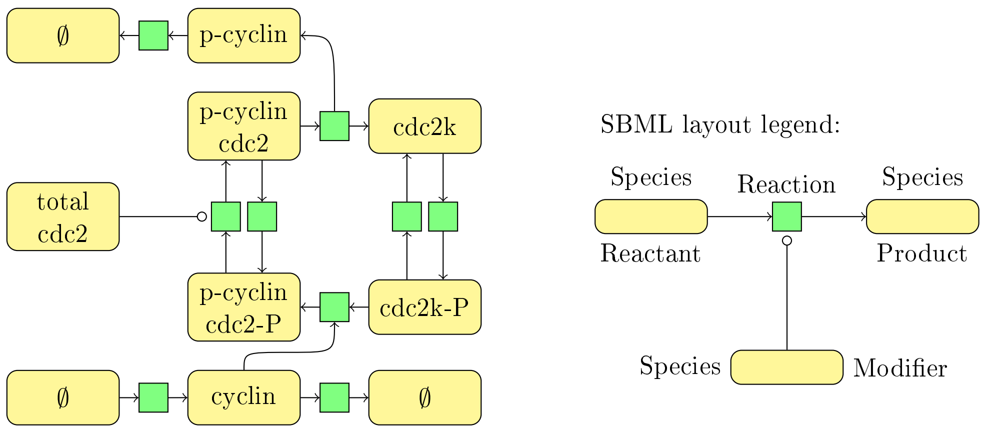


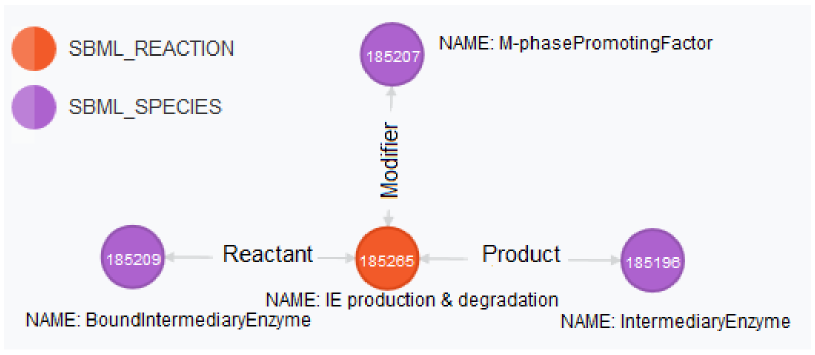
Figure 2: Reaction network of the cell cycle by Tyson.
Figure courtesy Dr. Christian Rosenke
Figure based on BioModels Database `http://www.ebi.ac.uk/biomodels-main/BIOMD0000000005`

The figure contains seven species and nine reactions as well as their relations. Hucka et al. (2003) defines species as "entities such as ions and molecules that participate in reactions" and a reaction as "some transformation, transport or binding process, typically a chemical reaction, that can change one or more chemical species". By using a graph representation, the networks consist of vertices for reactions as well as for species. Species are only related to reactions and reactions in turn are only related to species. These relations equate edges between the vertices in the graph representation. The edges can represent a relation for a reactant, product or modifier. Focusing on the reactions in the figure, an incoming edge is the relation to a species acting as a reactant and an outgoing edge is the relation to a species acting as product.
Modifier enable a reaction, but take not part actively. Edges representing the relation to a modifier are characterised by a cycle at the end. A reaction with its participating species correspond to a biochemical equation. Several reactions together built networks, if they share some species. The figure 2 shows such a network and in the following, a sub-network is described as an example. There is one reaction, where the species "p-cyclin cdc2-P" is a reactant, the species "total cdc2" is a modifier and the species "p-cyclin cdc2" acts as a product. Furthermore, the species "p-cyclin cdc2" participate in another reaction as reactant, where "p-cyclin cdc2-P" acts as a product. As a consequence, the both reactions sharing species are linked by these species and thus, built a network. In this specific case they even built a cyclic network.
In addition to the above mentioned elements, there are further components in SBML-models, which are not considered in this thesis. Hucka et al. (2003) state that a "[...] chemical reaction can be broken down into a number of conceptual elements: reactant species, product species, reactions, stoichiometries, rate laws, and parameters in the rate laws". Appendix 7.1 shows an example for an SBML-structure, but the focus is on elements regarded in this thesis - the species and reactions. Nevertheless, another important aspect is the existence of SBML-rules, which are mathematical expressions that can be used to add, for example, parameters and constraints to model equations. In some SBML-models this rules are utilized to represent parts of the networks. To reduce the complexity of finding structural similarities, the focus in this thesis is on the graph representation of the networks, where this rules are not taken into account.

3.2 Model Storage And Query Language

The biological models extracted from BioModels Database are stored in a NoSQL graph database called MaSyMoS[2], which more precisely is a Neo4j database. Operating data and initial key figures are accumulated by using the query language Cypher.
By now, there exists a variety of different alternatives to the longtime de-facto standard of the relational databases (Partner and Vukotic, 2012). These concepts are known as NoSQL. The name does not mean that the concepts are against the language SQL, but that they are non-relational databases, which are not built on the basis of tables. Furthermore, they generally use non-SQL languages and mechanisms for their interaction (Moniruzzaman and Hossain, 2013). The term NoSQL, which was coined in 1998, rather means 'Not Only SQL' and should indicate that a coexistence of the technologies is possible and the utility depends on the intended application. Graph databases are a certain category of NoSQL databases. They represent their data as networks of interconnected objects. In computer science graphs are extensively used for modelling and analysis (Partner and Vukotic, 2012). An example is the evaluation of network topologies in social networks, chemical compounds or atomic connections. The usage of graph databases is especially of interest when the focus is more on the relationships between data than on the data itself (Moniruzzaman and Hossain, 2013). An advantage of graph databases is the visual depiction as the example in Figure 3 shows. The graph model's structure consists of vertices and directed, typed edges that define the relationships (Hunger, 2014). Because attribute-value-pairs named as properties can be added to both - the vertices and edges - the data model is also called property-graph. The models used in this thesis are stored in a Neo4j database, which is a native open-source graph database. As described in the last Section 3.1, only models with their reactions and participatory species are considered for further examination. The graph representation of networks presented there is similar to their graph structure in the database. The networks consist of vertices for reactions as well as for species. They can be connected by edges representing a relation for a reactant, product or modifier (Henkel et al., 2014). Figure 3 shows an example reaction as it can be retrieved from MaSyMoS similarly. The vertices for reactions and species have an internal identifier and the additional properties ID and NAME.

Figure 3: Reaction example in MaSyMoS

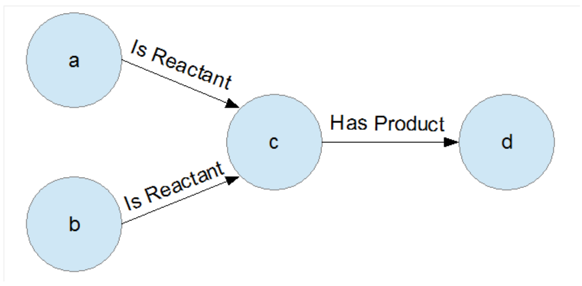


The figure shows an SBML-reaction adopted from the visual depiction in MaSyMoS

[2]https://sems.uni-rostock.de/projects/masymos/

The graph representation eases the use of algorithms of the well-founded graph theory (Partner and Vukotic, 2012). An example is to query data by using the graph traversal, which visits a set of nodes according to existing connections. Neo4j monitors the nodes already passed through, with the result that each node will just be visited once (Partner and Vukotic, 2012). It remains efficient even with increasing depth of the considered graphs.
In this thesis the language Cypher is used to query the Neo4j database MaSyMos. Cypher is a declarative query language, as well as SQL, but has specialized in graph querying (Hunger, 2014). Neo4j also supports other kinds of query languages like SPARQL and the imperative language Gremlin, but Cypher is widely used for Neo4j in particular (Robinson et al., 2013). Possible reasons for that may be the ease of reading and understanding this query language as well as it can be used to precisely describe graphs. One can intuitively query the database for compliance with certain patterns like using a diagram for the description. Figure 4 shows a graph of a reaction with its participating species that is used to ask Neo4j with Cypher for matches with this graph pattern.

Figure 4: Query example for a reaction with its participatory species



Cypher is developed to use it intuitively like drawing a graph.
This figure shows a graph pattern, which can be used to query the database

The figure describes the connection from a to c and b to c, which is synonymous with a species being a reactant in a reaction, and the connection from c to d as synonym for a reaction having a species as its product. For instance, this could be the synthesis of hydrogen and oxygen to water. This is similarly like one would work with Cypher:

```
MATCH (a)-[:Is_Reactant]->(c)-[:Has_Product]->(d), (b)-[:Is_Reactant]->(c)
```

To receive a complete Cypher-query and get the nodes of the reaction and species, one can add the line

```
RETURN a, b, c, d
```

at the end. The query means that one wants to search for all database entries, which match the given pattern and get returned for each match the three species and one reaction that are involved. It is also possible to ask for local matches around a specific starting point, for example, if only participatory species of a reaction with the name water_synthesis are searched, a START-clause can be added at the beginning of the Cypher query above. This clause could appear like

```
START c=node:index-name(name='water_synthesis')
```

For more information about Cypher and possible clauses see Robinson et al. (2013).
By using Cypher as query language, a key figure analysis could be performed to retrieve general information about the reaction networks of the used models. Examples of the utilised queries are listed in Appendix 7.5 and the results are explained in Section 5.1. The analysis revealed that there are several similarities within the models' networks, such as the number of reactions as well as the number of their participating species. Nevertheless, the key figure analysis just gives an overview about existing quantities. To find structural similarities of reaction networks, which make it possible to differentiate between groups of similar models, an appropriate automatic methods must be chosen. The next section describes several approaches, which have been examined for this purpose. The knowledge, gained by querying the database with Cypher, is helpful for a reasonable decision.

3.3 Approaches For Mining Structural Information

To find structural similarities within the networks of biological models, three different approaches have been examined - Clustering, Graph Similarity Search and Frequent Subgraph Mining. Each is examined below and finally, it is decided that frequent subgraph mining best fits the needs.
Clustering is the problem to automatically group a set of objects according to a given similarity criterion (Theodoridis and Koutroumbas, 2008). There are no predefined classes to assign the objects to. The aim is to find a hidden structure and thus, to gain new knowledge about the data. With just a bit prior knowledge the clustering algorithm has to discover similarities and differences of the data to organize groups (Nadler and Smith, 1993). For some algorithms the number of clusters has to be determined and the best partition is found by optimizing a cost function (Theodoridis and Koutroumbas, 2008). Others produce clustering sequences by starting with one cluster and splitting it or starting with a cluster for each object and merging it iteratively. The quality of the results depends on the intention of the human expert, who has to choose a similarity measure as well as a clustering scheme (Nadler and Smith, 1993). Nevertheless, clustering can still produce unexpected artificial groups, which are against human intuition. Another problem is, that not every data set is suitable for clustering, if it does not tend to distinguishable groups (Theodoridis and Koutroumbas, 2008). Therefore, this clustering tendency should be examined before using a clustering algorithm.
For applying clustering methods, fully implemented solutions were searched. Apache Mahout[3] is a scalable machine learning library that offers algorithms, among others, for clustering. The problem is that all provided clustering methods take so called feature vectors as their input, a characteristic vector per object of the data set. These vectors are used to compute a mathematical distance between the objects. Thus, feature vectors are not designed to use the special properties of graph structures (Riesen et al., 2007). It is assumed that generating meaningful feature vectors for reaction networks is very difficult without adequate knowledge about the used model data. That is why Apache Mahout is excluded as a solution. There are also structural clustering approaches based on graph representations, for instance Schellewald (2007) contains papers with such a topic. Foggia et al. (2007) evaluates four different graph-based clustering methods and Elghazel et al. (2007) uses b-colouring for a new greedy graph based partition clustering. Nevertheless, it is supposed that such an approach would possibly not give the desired results according to the above mentioned problems.

[3]http://mahout.apache.org/

Graph Similarity Search is the problem to find, within a set of graphs, the once that are similar to a given input graph (Zhao et al., 2013). Thus, its methods are well applicable for graph databases (Wang et al., 2009). There are two types of graph similarity search, the k-NN search finds the top k similar graphs and the range search finds results within a user-defined similarity score. On this basis a search function for models could be established perspectively, which delivers results with a ranking according to the similarity score. Graph similarity search algorithms mainly consist of three steps - efficient indexing for filtering, distance measure and query processing (Ding et al., 2014). There are various approaches with different priorities. Some paper focus on efficient indexing, for example Wang et al. (2012) suggest a two-level index designed from graph decomposition. Other propose effective filtering without pairwise similarity comparison (Yan et al., 2006). Ding et al. (2014) specialises on large attributed graphs and Wang et al. (2009) developed a kernel-based method for k-NN search in large graph databases.
Furthermore, a similarity measure appropriate for graphs is necessary. Riesen et al. (2007) states, that "[i]n contrast to statistical pattern recognition, where patterns are described by vectors, graphs do not offer a straightforward distance model like the Euclidean distance." Zheng et al. (2014) and Zhao et al. (2013) focus on the graph edit distance for measuring the similarity. The graph edit distance is a flexible measure that is widely used for graph approaches in research areas like classification, clustering and object recognition (Zheng et al., 2014). It measures the dissimilarity of graphs by computing the needed distortions - addition, deletion and substitution - of transforming one graph into another (Riesen et al., 2007). It is possible to assign different weighting to the transforming operations (Wang et al., 2009). This could enable a search function for models with variable focus of the ranked results. Nevertheless, it may not be well applicable as a stand-alone approach for purposes like gaining new knowlede and grouping models, because information can just be retrieved by putting in graphs for similarity testing. Thus, interesting unexpected structures could be ignored. Furthermore, it is questionable how precise the similarity can be measured without adequate prior knowledge. Therefore, it was decided to focus on data mining and to notice graph similarity search as a future work. Then, the performance and correctness of a future search function may be improved with a specialised distance measure based on the gained knowledge.
Frequent Subgraph Mining is a research area of data mining, which is specialised in graph structures (Lakshmi and Meyyappan, 2012). It is the problem to find the structures embedded in graphs, so called subgraphs, that occur frequently (Keyvanpour and Azizani, 2012). The key figure analysis (results in Section 5.1) shows the existence of similarities within major parts of the model's quantities. This outcome suggests, that there could be biologically precious patterns, which occur with a higher frequency. For that reason, the decision was made to apply a frequent subgraph mining algorithm to the set of biological models. Furthermore, using the occurrences of the resulting frequent subgraphs as features for the models may provide a meaningful similarity measure. Such an appropriate similarity measure can improve the grouping of models as well as searching similar models by increasing the search speed, the exactness of the outcome and the scalability (Keyvanpour and Azizani, 2012). In addition, finding functional structures as described in Tyson and Novák (2010) would be a great achievement. Hence, frequent subgraph mining was chosen for closer examination, which follows in the next section.

3.4 Frequent Subgraph Mining Algorithms

Given a set of graphs, frequent subgraph mining (abbrv. FSM) is the problem to find subgraphs within these graphs that pass a given frequency threshold (Keyvanpour and Azizani, 2012). The algorithms mainly consist of two steps. Candidate generation is the step to list all possibly occurring subgraphs and frequency counting determines the number of occurrences of the candidates. The first step bears the risk to produce candidates in a exponential number according to the size of the searched structures. The latter requires subgraph isomorphism testing - the problem to decide whether a graph is embedded in another (Lakshmi and Meyyappan, 2012). This problem is known as an NP-complete task (Keyvanpour and Azizani, 2012). Thus, FSM techniques require heuristics, prior knowledge or another particular strategy to improve the performance. Various FSM algorithms with several priorities for improvement have been developed so far, which is the reason why choosing an algorithm appropriate for the use case is an important element of this thesis. Finally, the decision was made for an algorithm called gSpan (Yan and Han, 2002). Reasons for this decision are, among others, the supply of exact results instead of approximate once, the combination of candidate generation and frequency counting in one procedure and the existence of a pre-implemented version of gSpan. For a better understanding of further characteristics, a brief overview about FSM algorithms is given below and then, gSpan and its used implementation is described.
There are a lot of different Frequent Subgraph Mining algorithms (Keyvanpour and Azizani, 2012). To choose an appropriate one for a certain application, considering important aspects of the methods is necessary. These aspects are especially the type of input graph, the necessity of prior background knowledge, the need for exact or just approximate results as well as for completeness of the resulting pattern set, the available memory and the possibility of user intervention. Table 1 (page 28) shows common Frequent Subgraph Mining algorithms characterised by important features.

FSM algorithms can be differentiated, for example, according to their input type (Keyvanpour and Azizani, 2012). Some algorithms take one large graph and find the frequent subgraphs depending on the frequency within this graph. Other have a graph set as their input and search for structures that occur in at least a certain number of graphs within the set. The aim of this thesis is to find structural similarities in a collection of reaction networks, such that it becomes possible to group the models according to the structures of their networks. For that reason, an FSM algorithm with a graph set as input is assumed to be appropriate, whereby the networks must not be combined to a large graph and considering the frequency according to the number of graphs excludes outliers in the network structures. An example for an outlier in MaSyMoS is one model that has several reactions, where more than 170 modifiers are involved, whereas the most other models have reactions with just one or two modifiers (Section 5.1).
The candidate generation method is another important characteristic. There are mainly four bases mentioned for these methods - join, extension, inductive logic programming and replacing. Candidate generation by join means beginning with small frequent substructures and merging them to bigger structures where frequent ones in turn can be joined. Extension based methods start with frequent nodes and iteratively add one of each possible edges, while infrequent patterns often are pruned immediately and will not be observed for further extension. By using inductive logic programming, abbrv. ILP, first order predicates represent the subgraphs. Keyvanpour and Azizani (2012) state that in the replacing strategy "[...] after detecting the frequent subgraph

in each stage, the detected subgraph is replaced by a node in the main graph and in the next stage, the mining process continues on a new graph obtained from graph replacing." GSpan is an extension based algorithm that takes a graph set as its input and produces all frequent connected subgraphs according to a given frequency threshold (Yan and Han, 2002). Therefore, it uses a unique minimum depth first search code of the graphs, a lexicographic ordering on these codes and, based on that, it builds a search tree. Because it uses the minimum DFS code of graphs as canonical label, two graphs are isomorphic if and only if their code is equal. This fact transforms the task into a sequential pattern mining problem, where already algorithms exist to solve it. Furthermore, gSpan accelerates discovering patterns by combining candidate generation and frequency counting, while efficient pruning is performed. It also avoids false positive pruning. GSpan is for example used by Priyadarshini and Mishra (2010) and its performance is evaluated in comparison to the algorithms MoFa, FFSM and Gaston by Wörlein et al. (2005). The latter have developed a software for this purpose, which is called the Parallel and Sequential Mining Suite (abbrv. ParSeMiS[4]). It is a Java based project and realises the algorithms gSpan, Gaston and Dagma as well as some extensions like CloseGraph.
The above mentioned advantages of gSpan, such as the use of a canonical labelling and the availability of a full implementation, led to the decision of using the gSpan algorithm for the purpose of this thesis.

3.5 Classification

The aim of this is to extract structural features of model networks that can serve as a reasonable similarity measure for the classification of the networks. Classification in the field of pattern recognition is the problem to automatically map objects to predefined classes (Theodoridis and Koutroumbas, 2008). To ensure optimal decisions for the classification, prior knowledge is necessary. A main task for the classification is to extract suitable features (Nadler and Smith, 1993). Assigning objects to classes often depends on the similarity of the objects to class representatives and therefore, choosing an appropriate similarity measure is necessary (Theodoridis and Koutroumbas, 2008). Most algorithms require learning to increase the correctness of the results (Nadler and Smith, 1993). Learning means for example, that an expert has to determine misclassified objects in iterative steps (Sharma and Kaur, 2013). A difficulty of classification is to deal with outliers or ambiguous cases, which potentially require an extra class defined or otherwise they will increase the error rate of the method (Nadler and Smith, 1993). For example, MaSyMoS contains one model that has several reactions, where more than 170 modifiers are involved. The most reactions have just one or two modifiers.
The aforementioned clustering (Section 3.3) can be seen as a special case of classification (Nadler and Smith, 1993), where the patterns have to be organized into reasonable groups without existence of predefined classes (Theodoridis and Koutroumbas, 2008). For further information see (Lambusch, 2015).

[4]https://www2.cs.fau.de/EN/research/zold/ParSeMiS/index.html

4 Implementation

To find frequent occurring patterns within a set of biological models, the "Parallel and Sequential Mining Suite[5]" (abbrv. ParSeMiS) is used, which is implemented in Java and further needs the libraries for antlr and prefuse[6]. Appendix 7.2 shows an adjusted method of ParSeMiS to get it work. By means of Eclipse[7] a runnable jar file is produced. Listing 1 illustrates the proceeding from getting an appropriate input file for working with ParSeMiS to the post-processing for image-files of the patterns found.

Listing 1: Pseudocode for producing results

```
Get reaction networks as json-file
Substitute irrelevant data
Convert json to dot
Split graph within dot-file to unconnected graphs
Execute ParSeMiS version of gSpan
Add appearence properties to found patterns
Split file in separate files for each pattern
Create image file for each pattern
```

First, the relevant network data are retrieved as a json file with the tool curl (Appendix 7.3). Because, there are still various irrelevant data in it, they are substituted by using the tools awk and sed. ParSeMiS is, among other file formats, applicable to files with graphs in a dot-format. Thus, a script is applied to convert the json-file to a dot-file. The script is shown in Appendix 7.4. The dot-file then contains one large graph for all the networks. The tool ccomps is used to split this graph into all unconnected networks. Then, the gSpan version of ParSeMiS can be applied to get a dot-file with the patterns that fulfil a given frequency threshold [8]. For this thesis ParSeMiS ran on an Intel Xeon E5410 Quad-Core Processor with 16GB main memory, 2.33GHz, Debian 8 as operating system and Oracle Java SE 64bit version 1.7.0. In dot-files attributes for the visualisation of the graphs can be added. The tool sed is used to add colours and other attributes to the graph elements. The tools grep and sed are used to get a file with the frequency of each pattern, which is contained as a comment in the output file of ParSeMiS. With the tool csplit the patterns within the dot-file are split into separate files and the name can be chosen according to the frequency of each pattern. These dot-files can directly be converted to image-files with the dot tool.

[5]`https://github.com/timtadh/parsemis`

[6]used versions: antlr 2.7.5, prefuse release 2007.10.21

[7]Version: Kepler Service Release 1, jre 1.8.0 update 40, java SE development kit 1.7.0 update 45

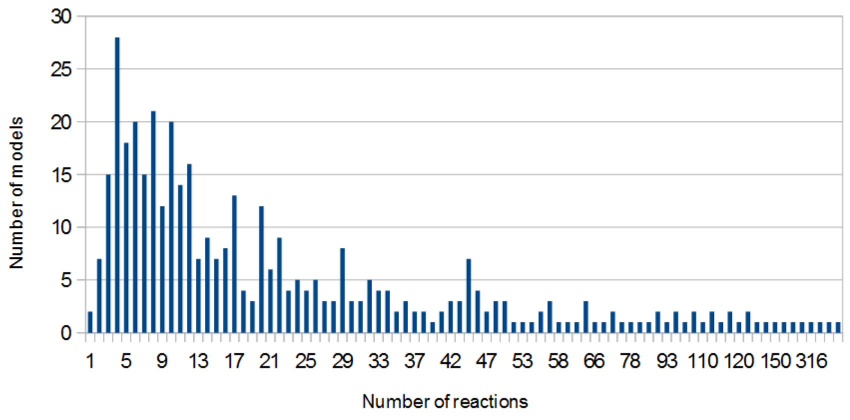
[8]`java -jar ParSeMiS_final.jar -graphFile=testFile.dot -outputFile=fragments.dot -minimumFrequency=250`

5 Results

5.1 Key Figure Analysis

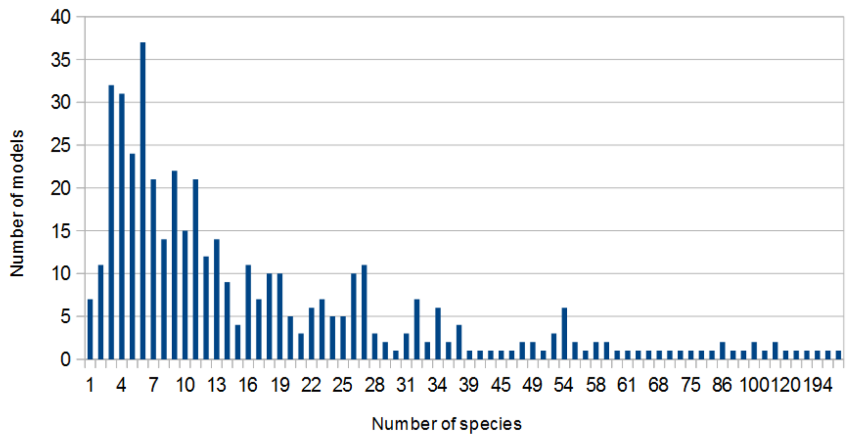
The aim of the key figure analysis is to get quantities of the vertices for models, reactions and species as well as the edges between them. There are 1303 models in the database, whereof 462 are SBML-models. 445 of them have species and 403 have reactions. Each reaction belongs to exactly one SBML-model and there exist 12335 reaction vertices in total. Figure 5 shows, that each of the models has one up to 827 reactions, with an accumulation of models that have three up to twelve reactions. The maximum amount of models, which share the same number of reactions, is 28 models with a count of four reactions. Furthermore, there are a few outliers with more than 100 reactions. The average number of reactions within a model accounts for 31, while the variance is 4451 and the standard deviation is 67.

Figure 5: Models with their number of reactions



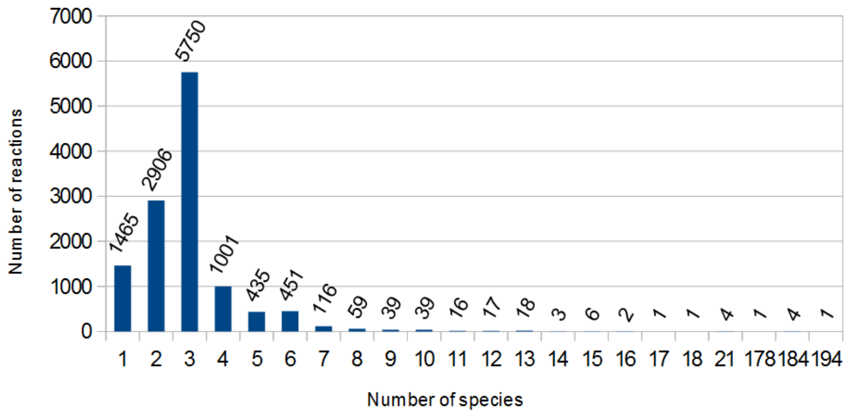
There exist 9512 species vertices in the database. 445 of the SBML-models have one up to 622 participating species (Figure 6). A noticeable accumulation of models can be found from three up to eleven species, while there are just a few models with more than 54 Species. The average number of species within a model amounts to 21. The variance accounts for 88 and the standard deviation accounts for 9.

Figure 6: Models with their number of species



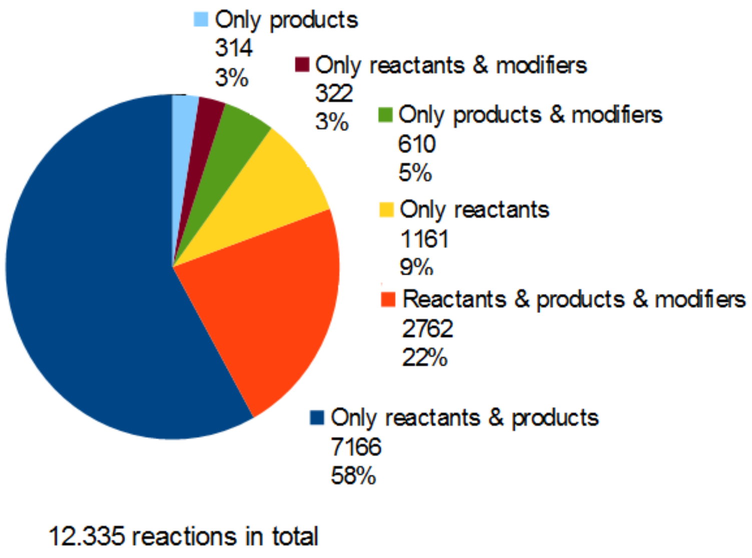
In Figure 7 one can see that a large part of reactions contain three species. In general, the reactions have one up to 194 species. The arithmetic average amounts to three, the variance to six and the standard deviation to two.

Figure 7: Reactions with their number of species



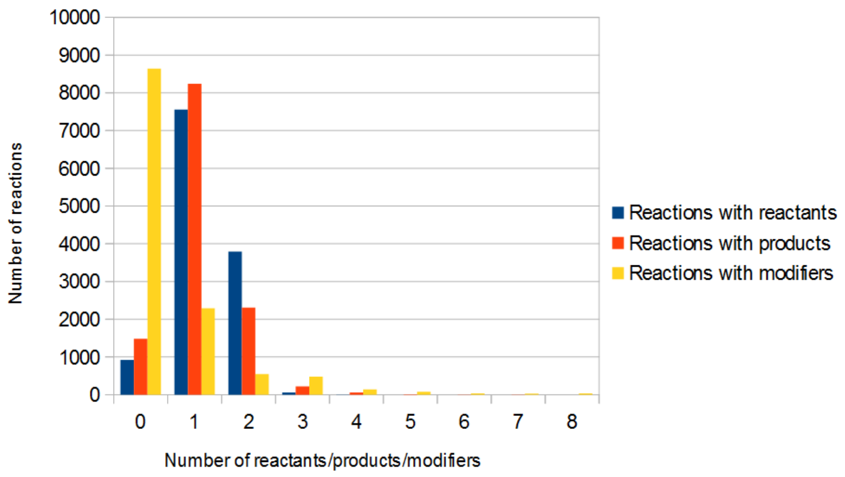
These species can act as reactants, products or modifiers within the reactions. Figure 8 shows how many percent of the reactions have reactants, products or modifier. One can see that more than half of the reactions have only reactants and products. Furthermore, a larger percentage of reactions have reactants, products and modifier. There are no reactions with only modifiers and thus, no ones that have neither reactants nor products. From the entirety of 12335 reactions, 11411 contain reactants, 10852 products and 3694 reactions have modifier.

Figure 8: Percentages of reactions that have reactants, products or modifier



The distribution from Figure 8 is reflected in Figure 9, which shows that most reactions have one or two reactants as well as one or two products, and no or one modifier. The reactions have a maximum of four reactants, 14 products and normally a maximum of 19 modifiers. There is just one model with six reactions that have more than 174 modifiers with a maximum of 181 modifiers.

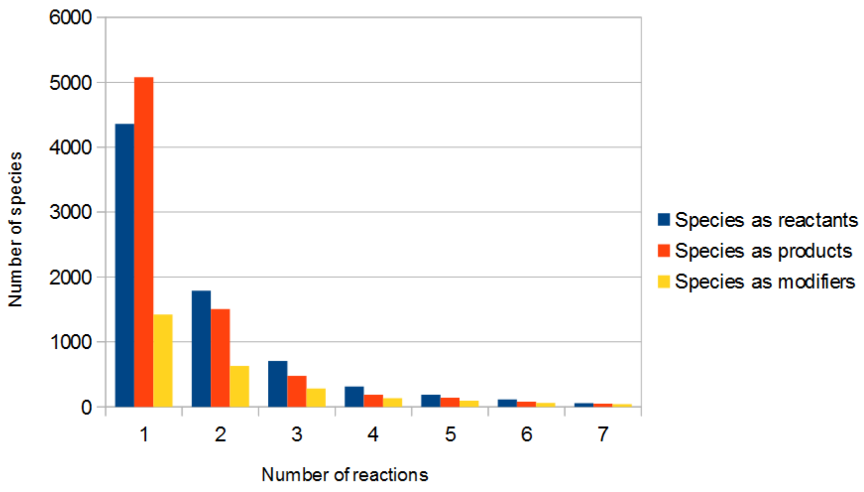
Figure 9: Number of reactants, products or modifiers of the reactions



The figure shows all numbers of reactants and products, but only numbers of modifiers that are contained in more than 20 reactions

Figure 10 presents in how many reactions the species take place as reactant, product or modifier. Most species are contained in just one or two reactions, while the maximum of reactions in which species are contained as reactant is 180, as product 360 and as modifier 359.

Figure 10: Species take place in X reactions as reactants, products or modifier



Species, which take place in more than seven reactions,
are not shown, because there are less than 50 species for each number of reactions

Furthermore, from the entirety of 9512 species 7642 serve as reactants, 7632 as products and 2889 as modifiers. 459 species even serve neither as reactant nor product nor modifier.
These results could be retrieved by using the language Cypher to query the used graph database MaSyMoS. Appendix 7.5 gives examples of the queries used for the key figure analysis.

5.2 Patterns Found

This section presents the results from searching frequent structures within the models. All subgraphs could be found, which occur in at least 55 percent of the 453 input graphs. Just a few of the 53 resulting patterns are portrayed below. The full list is in Appendix 7.6. The patterns that occur in 100 percent of the models are just the two smallest existing structures with one vertex and zero edges - the vertex for species (Figure 11a) and for reaction (Figure 11b).

Figure 11: Smallest patterns found

(a) Species vertex

SBML_SPECIES

(b) Reaction vertex

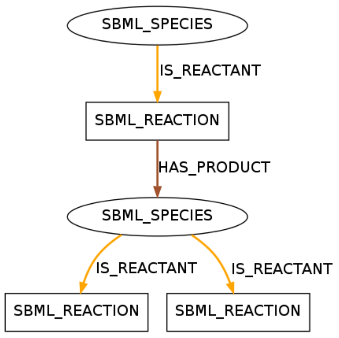
SBML_REACTION

. Patterns found in 453 of 453 graphs

Furthermore, the pattern of a reaction with one product is contained in 452 graphs, while the pattern of a reaction with one reactant can just be found in 435 of the 453 graphs. The most frequent structure with a branch occurs in 357 models and is a species that serves as reactant for two reactions. Thus, it is the most frequent pattern that contains a vertex with two outgoing edges, while the first pattern of a vertex with two incoming edges - a species as product for two reactions - can be found in 325 graphs. The maximum overall degree of vertices is three, whereas the subgraphs have up to two outgoing edges (see for example Figure 12a) or up to two incoming edges (see for example Figure 12b).

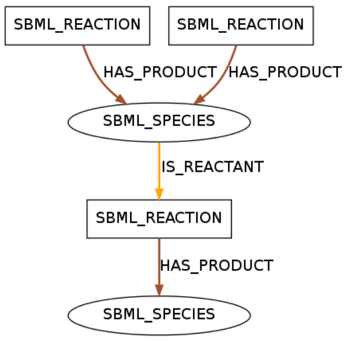
Figure 12: Patterns with degree of vertices up to three

(a) Contains vertex with outgoing degree of two



Pattern found in 314 of 453 graphs

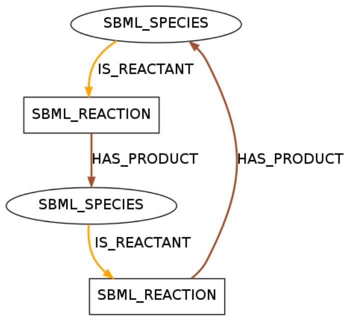
(b) Contains vertex with incoming degree of two



Pattern found in 280 of 453 graphs

Figure 13 shows the one cyclic structure found. It consists of four vertices - two species and two reactions - as well as four edges. All components belong to the cycle. Both species serve as reactant for a distinct reaction and are the product of the other reaction.

Figure 13: Cyclic pattern

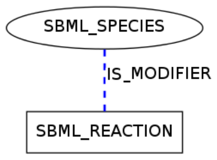


Pattern found in 262 of 453 graphs

With decreasing frequency, the patterns tend to be larger. Mainly structures that contain modifiers are strikingly smaller in comparison to others with similar frequency, as one can see in Figure 14. No patterns could be found, which contain more than one modifier.

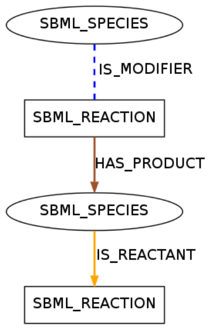
Figure 14: Patterns containing a modifier

(a) Smallest pattern containg a modifier



Pattern found in 297 of 453 graphs

(b) Larger pattern containing a modifier

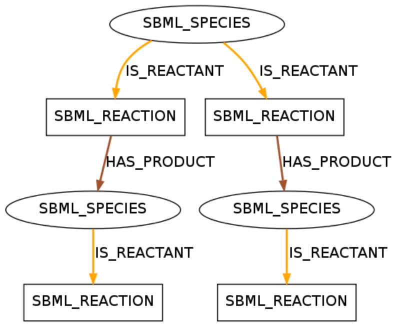


Pattern found in 266 of 453 graphs

The largest structures found have seven vertices and six edges. Examples are shown in Figure 15. Furthermore, four vertices for reactions, four edges for a reactant (Figure 15a) and four edges for a product (Figure 15b) is the maximum amount found within the set of subgraphs.

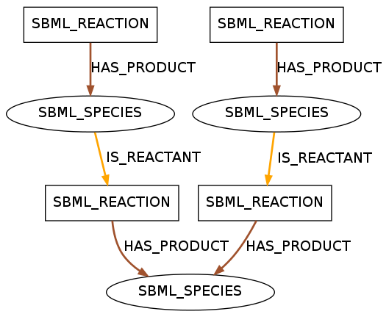
Figure 15: Examples for large branched patterns

(a) Contains four edges for reactants



Pattern found in 254 of 453 graphs

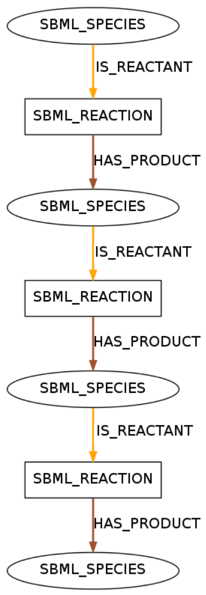
(b) Contains four edges for products



Pattern found in 252 of 453 graphs

Figure 16 shows a structure, which is also one of the largest found patterns without branches. Moreover, it is an example for the maximum count of species vertices, which is four, and the maximum count of vertices generally, which is seven. As well, it contains the longest directed path within the subgraphs, that amounts to six.

Figure 16: Example for a pattern containing the longest path



Pattern found in 269 of 453 graphs

6 Discussion

The chosen combination of key figure analysis and searching the most frequent patterns revealed new knowledge about the network structure of SBML-models. This knowledge can be used to compare structures and thus, to group models according to their reaction networks. Possible solutions will be explained later in this section. Valuable patterns could be found. Their properties correspond to the results of the key figure analysis. According to this analysis, more than a half of the reactions have no modifiers (Figure 8) and the other reactions mostly have only one modifier (Figure 9). These quantities are reflected in the rare occurrences of modifiers within the found patterns. The key figure analysis moreover showed that the models contain on average 31 reactions and 21 species, while most reactions have only up to three participating species and most of the species just take part in one or two reactions. These results demonstrate that the models' networks are sparsely populated. There are one or less branches within the patterns. Only one of the patterns with the lowest frequency contains a species, which acts as reactant for two reactions and another species that acts as product for two reactions. It is assumed that the number of occurring branches is increasing with lower frequency of the patterns. A remarkable fact is that no reaction with several reactants or products is contained in any pattern, although nearly a half of the existing reactions have three participating species. This suggests, that reactions with more than two species involved may be frequent only within a certain type of models, such that these reactions are not spread over a large amount of models. Thus, they could not be found with the used approach. Having more than three participating species then could be a characteristic of a certain group of models. Examining this is a task for future work. A possible approach for that purpose would be to use another frequent subgraph mining approach that respects the number of subgraph occurrences in total instead of considering subgraphs that occur in a certain number of graphs.

There are several patterns found, which may occur - combined to larger structures - with a lower frequency than considered. For example Figure 15a and Figure 15b may yield a large cycle. Although subgraphs are found with up to seven vertices, there is only one cycle with four vertices found. The next larger cycle with six vertices is not contained in the results. This encourages the assumption that cycles are relatively rare with increasing frequency or occur with a larger number of vertices involved. The path lengths increase with the decrease in frequency of the patterns. In this thesis only certain aspects of the models are considered, such that the found patterns just consist of the nodes for species and reactions as well as edges for reactants, products and modifiers. For a complete analysis all additional information should be involved in future studies. Examples are the so called SBML-rules - mathematical expression that are used to add, among others, parameters and constraints to the models (Hucka et al., 2003). The following illustrates a conspicuousness that appeared in this context. To get the input for ParSeMiS, a file with all relevant nodes and their edges is created and then split into connected graphs. Thereby, a file with 453 graphs is produced, while the database just contains 445 SBML models with species or reactions. I assume that the problem of different quantities consists in the fact that some models have unconnected parts in their reaction networks. Their connections may only be described by SBML rules, which remain unconsidered in this thesis. The existence of more graphs in the file should not interfere the mining process, because gSpan mines only frequent connected subgraphs anyway (Yan and Han, 2002). Nevertheless, there exists a problem with the models, which use SBML rules instead of modelling the behaviour by means of connections

between reactions and species. Their structure can not completely be mined without considering the specified rules. For a complete analysis of the data in future studies, the rules have to be involved in the input file or rather nodes and edges have to be created for their expressions in the database.
But indeed, valuable patterns were found, such as a cyclic structure (Figure 13) or a reaction chain with a path length of seven (Figure 16). These patterns can serve as a reasonable similarity measure to classify models according to the structure of their reaction networks. One possibility is to directly use the evident characteristics resulting from the found patterns. For example, it is observed that the path length increases with the decrease in the frequency of patterns. Thus, the existence of shorter or longer paths within a model could serve as an appropriate feature for a classification. Furthermore, it is mentioned above that reactions with more than two species involved seem to be concentrated in some models and may characterise a group of models. Other examples are the occurrences and size of rare cycles as well as the number of branches within the networks. As one can see, there are already various useful features, which can be extracted from the patterns' appearance to provide grouping criteria.
Moreover, the occurrences of the found patterns may provide a great similarity measure between model networks. Lakshmi and Meyyappan (2012) state that the simple pairwise comparison of nodes and edges within a network completely neglects the structure, whereas viewing networks as similar, if they share many common substructures, is a much more adequate similarity measure. For that reason, the number of occurrences of each characteristic pattern found can be examined within each model perspectively. Then, a feature vector for each model can be created, where the entries correspond to the number of occurrences of each pattern. As mentioned in Section 3.3, there are fully implemented classification and clustering tools that take feature vectors as input. For example Apache Mahout[9] provides such solutions. Thus, after creating the feature vectors, these can directly be used to group the models automatically according to their network structure. By organising the models like this, it becomes possible that researchers can easily browse through a group of models that is related to their work. As a consequence, correlations between models can quickly be recognized and may lead to new discoveries in the behaviour of structural similar networks. Furthermore, the reusability of models can be increased by offering the opportunity to easily find relevant network structures. The number of pattern occurrences per model - needed for creating feature vectors - can efficiently be queried from MaSyMoS with the language Cypher. To ease the effort of manually translating each pattern into an appropriate shape for a query, automatic methods for this purpose should be developed in the future.
A biological evaluation and a comparison of the found structures with the motifs of Tyson and Novák (2010) is a task for future work, too. Matches could show the capability and accuracy of an automatic mining procedure such as gSpan as well as so far undiscovered functional structures could be found. Furthermore, gSpan could be executed again only on thematically similar models to discover such functional patterns for certain groups of models. The main future task is to create a search function for models that provides ranked results according to the structural similarity to an input model. With the results of this thesis a great step forward is made. Various similarity criteria are found as well as the requisite preconditions for classification or clustering are established. With groups of structural similar objects, the computational costs of a ranked similarity search can be reduced by scaling down the search space.

[9]http://mahout.apache.org/

Table 1: Comparison of a selection of common FSM-Algorithms, adapted from (Keyvanpour and Azizani, 2012) and (Lakshmi and Meyyappan, 2012), additional information for FSMA algorithm are taken from (Wu and Chen, 2008)

Algorithm	Year	Input type / Topology	Graph representation	Kind of mined subgraphs / Nature of output	Candidate generation	Search method / Frequency counting / Search type	Effective application area	Limitations
SUBDUE	1994	Labelled single large graph	Adjacency matrix	Complete set of frequent, connected subgraphs	Level-wise search	Greedy incomplete search on minimum description code length	Databases without subgraphs with high frequency	Extremely small number of patterns
GREW	1998	Labelled, undirected, single large graph	Sparse graph	Frequent, connected, maximal subgraphs	Iterative merging	Greedy incomplete search on maximal independent set		Misses many interesting patterns
FARMER	1998	Set of graphs, topology not limited	Trie structure	Frequent, connected subgraphs	Level-wise search ILP	Breadth first search on trie datastructure based on background knowledge	Proof of concept of a framework	Inefficient
AGM	2000	Graph DB, topology not limited	Adjacency matrix	induced frequent subgraphs	Vertex extension	Complete breadth first search on canonical labels		
FSG	2001	Set of undirected graphs	Adjacency list	Frequent, connected subgraphs	One edge extension	Complete breadth first search on transaction identifier (TID) lists	Databases with variety of node and edge labels	NP-complete
HISIGRAM	2004	Labelled, undirected, single large graph	Adjacency matrix	Frequent, connected subgraphs	Iterative Merging	Complete breadth first search on maximal independent set	High-scale sparse graph	Inefficient
MOFA	2002	Set of labelled, undirected graphs	Adjacency list	All frequent, connected subgraphs	Rightmost extension	Complete depth first search on DFS lexicographic order	Molecular databases	Generated graphs may not be exactly frequent
gSpan	2002	Set of labelled, undirected graphs	Adjacency list	Frequent, connected subgraphs	Rightmost extension	Complete depth first search on DFS lexicographic order		Not scalable
Gaston	2004	Set of labelled, undirected graphs	Hash table	Frequent, connected, maximal subgraphs	Extension	Complete depth first search on embedding lists		Interesting patterns may be lost
FFSM	2003	Set of labelled, undirected graphs	Adjacency matrix	Frequent, connected subgraphs	Merging and extension	Complete depth first search on suboptimal canonical adjacency matrix tree		NP-complete
FSMA	2008	Set of labelled graphs	Incidence matrix	Frequent, connected subgraphs	Extension	Complete breadth first search on normalized incidence matrix	Graphs that do not share special characteristics	

The table shows a comparison of common FSM-algorithms according to important characteristics, such as the input type, graph representation and nature of output

Bibliography

Alm, R., Waltemath, D., Wolkenhauer, O., and Henkel, R. (2014). Annotation-Based Feature Extraction from Sets of SBML Models. In *Data Integration in the Life Sciences*, pages 81—-95. Springer.

Ding, X., Jia, J., Li, J., Liu, J., and Jin, H. (2014). Top-k Similarity Matching in Large Graphs with Attributes. *Lecture Notes in Computer Science - Database Systems for Advanced Applications*, Volume 842:pp 156–170.

Elghazel, H., Yoshida, T., Deslandres, V., Hacid, M.-S., and Dussauchoy, A. (2007). A New Greedy Algorithm for Improving b-Coloring Clustering. In *Graph-Based Representations in Pattern Recognition*, pages 228—-239. Springer.

Fionda, V. and Palopoli, L. (2011). Biological Network Querying Techniques: Analysis and Comparison. *Journal of Computational Biology*, 18(4):595–625.

Foggia, P., Percannella, G., Sansone, C., and Vento, M. (2007). Assessing the performance of a graph-based clustering algorithm. In *Graph-Based Representations in Pattern Recognition*, pages 215—-227. Springer.

Gleeson, P., Crook, S., Cannon, R. C., Hines, M. L., Billings, G. O., Farinella, M., Morse, T. M., Davison, A. P., Ray, S., Bhalla, U. S., Barnes, S. R., Dimitrova, Y. D., and Silver, R. A. (2010). NeuroML: A language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS Computational Biology*, 6(6):1–19.

Hattori, M., Okuno, Y., Goto, S., and Kanehisa, M. (2003). Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways. *J Am Chem Soc*, 125(39):11853–11865.

Henkel, R., Wolkenhauer, O., and Waltemath, D. (2014). Combining computational models, semantic annotations, and associated simulation experiments in a graph database. *PeerJ*, pages 1–20.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J. H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531.

Hunger, M. (2014). *Neo4j 2.0: Eine Graphdatenbank für alle.*

Keyvanpour, M. R. and Azizani, F. (2012). Classification and Analysis of Frequent Subgraphs Mining Algorithms. *Journal Of Software*, 7.

Koyutürk, M., Grama, A., and Szpankowski, W. (2004). An efficient algorithm for detecting frequent subgraphs in biological networks. In *Bioinformatics*, volume 20.

Lakshmi, K. and Meyyappan, T. (2012). Frequent Subgraph Mining Algorithms - A Survey And Framework For Classification.

Lambusch, F. (2015). *Pattern Recognition in the Context of Computational Models.* Literature study, Universität Rostock.

Li, C., Donizelli, M., Rodriguez, N., Dharuri, H., Endler, L., Chelliah, V., Li, L., He, E., Henry, A., Stefan, M. I., Snoep, J. L., Hucka, M., Le Novère, N., and Laibe, C. (2010). BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC systems biology*, 4:92.

Lloyd, C. M., Halstead, M. D. B., and Nielsen, P. F. (2004). CellML: Its future, present and past. In *Progress in Biophysics and Molecular Biology*, volume 85, pages 433–450.

Moniruzzaman, A. B. M. and Hossain, S. A. (2013). NoSQL Database : New Era of Databases for Big data Analytics- Classification , Characteristics and Comparison.

Nadler, M. and Smith, E. P. (1993). *Pattern Recognition Engineering.*

Partner, J. and Vukotic, A. (2012). Neo4j in Action. chapter 1. Meap edition.

Priyadarshini, S. and Mishra, D. (2010). An approach to graph mining using gspan algorithm. In *2010 International Conference on Computer and Communication Technology, ICCCT-2010*, pages 425–430.

Riesen, K., Neuhaus, M., and Bunke, H. (2007). Bipartite Graph Matching for Computing the Edit Distance of Graphs. In *Graph-Based Representations in Pattern Recognition*, pages 1—-12. Springer.

Robinson, I., Webber, J., Eifrem, E., Breu, F., Guggenbichler, S., and Wollmann, J. (2013). *Graph Databases.*

Robinson, P. N. and Bauer, S. (2011). *Introduction to Bio-ontologies.* Chapman & Hall/CRC mathematical and computational biology series. CRC Press/Taylor & Francis.

Schellewald, C. (2007). *Graph-Based Representations in Pattern Recognition*, volume 4538 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg.

Schulz, M., Klipp, E., and Liebermeister, W. (2012). Propagating semantic information in biochemical network models.

Schulz, M., Krause, F., Le Novère, N., Klipp, E., and Liebermeister, W. (2011). Retrieval, alignment, and clustering of computational models based on semantic annotations. *Molecular systems biology*, 7:512.

Sharma, P. and Kaur, M. (2013). Classification In Pattern Recognition: A Review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4):9.

Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition, Fourth Edition*, volume 11.

Tyson, J. J. and Novák, B. (2010). Functional motifs in biochemical reaction networks. *Annual review of physical chemistry*, 61:219–240.

Wang, X., Ding, X., Tung, A. K. H., Ying, S., and Jin, H. (2012). An efficient graph indexing method. In *Proceedings - International Conference on Data Engineering*, pages 210–221.

Wang, X., Smalter, A., Huan, J., and Lushington, G. H. (2009). G-Hash: Towards Fast Kernel-based Similarity Search in Large Graph Databases. *Advances in database technology : proceedings. International Conference on Extending Database Technology*, 360:472–480.

Wong, E., Baur, B., Quader, S., and Huang, C. H. (2011). Biological network motif detection: Principles and practice. *Briefings in Bioinformatics*, 13(2):202–215.

Wörlein, M., Meinl, T., Fischer, I., and Philippsen, M. (2005). A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM, and Gaston. *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 392–403.

Wu, J. and Chen, L. (2008). Mining Frequent Subgraph by Incidence Matrix Normalization. *Journal Of Computers*, Vol.3 No.1.

Yan, X., Zhu, F., Yu, P. S., and Han, J. (2006). Feature-based similarity search in graph structures.

Yan, X. Y. X. and Han, J. H. J. (2002). gSpan: graph-based substructure pattern mining. *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*

Zhao, X., Xiao, C., Lin, X., Liu, Q., and Zhang, W. (2013). A partition-based approach to structure similarity search. *Proceedings of the VLDB Endowment*, 7(3).

Zheng, W., Zou, L., Lian, X., Wang, D., and Zhao, D. (2014). Efficient Graph Similarity Search Over Large Graph Databases. *IEEE Transactions on Knowledge and Data Engineering.*

7 Appendix

7.1 SBML Example With Focus On Species And Reactions

Listing 2: SBML structure, adopted from (Hucka et al., 2003)

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level1" level="1" version="2">
  <model name="gene_network_model">
        <listOfUnitDefinitions>
          ...
        </listOfUnitDefinitions>
        <listOfCompartments>
          ...
        </listOfCompartments>
        <listOfSpecies>
          <species name="RNAP" compartment="Nuc"
          initialAmount="0.66349" />
          <species name="src" compartment="Nuc"
          initialAmount="1" boundaryCondition="true" />
          ...
        </listOfSpecies>
        <listOfParameters>
          ...
        </listOfParameters>
        <listOfRules>
          ...
        </listOfRules>
        <listOfReactions>
          <reaction name="R1" reversible="false">
                <listOfReactants>
                  <species Reference species="src" />
                </listOfReactants>
                <listOfProducts>
                  <species Reference species="RNAP"/>
                </listOfProducts>
                  ...
          </reaction>
          ...
        </listOfReactions>
  </model>
</sbml>
```

7.2 ParSeMiS Changes

Listing 3: The substitution for the "mine" method in Miner.java of ParSeMiS

```
public static Collection<Fragment> mine(Collection graphs,
Settings settings)
{
        final Statistics stats = settings.stats;

        // start memoryCheck, if necessary
        Thread t = null;
        if (settings.memoryStatistics) {
                t = memoryCheck(stats);
                t.start();
        }
        // search fragments
        if (INFO) {
                stats.searchTime -=
                        System.currentTimeMillis();
                stats.searchTime2 -=
                        LocalEnvironment.currentCPUMillis();
        }

        final Collection<Fragment> expectedFragments =
                settings.algorithm.initialize(graphs,
                settings.factory, settings);

        Collection<Fragment> ret = settings.strategy
                        .search(settings.algorithm);

        ret.addAll(expectedFragments);

        if (INFO) {
                stats.searchTime +=
                        System.currentTimeMillis();
                stats.searchTime2 +=
                        LocalEnvironment.currentCPUMillis();
        }
        // filter fragments, if necessary
        final FragmentFilter filter = LocalEnvironment
                        .env(settings.strategy).filter;
        if (filter != null) {
                if (INFO) {
                        stats.filteringTime -=
                                System.currentTimeMillis();
                }
                ret = filter.filter(ret);
                if (INFO) {
                        stats.filteringTime +=
                                System.currentTimeMillis();
                }
        }

        // stop memoryCheck, if necessary
        if (t != null) {
                t.interrupt();
        }

        return ret;
}
```

7.3 Curl Command To Get Json-File With Networks

Listing 4: PHP script to convert json file format to dot format

```
curl -X POST -d '{"query": "MATCH (r:SBML_REACTION)-[h]->(s:SBML_SPECIES)
RETURN r,s,h", "params": {} }' http://sems.uni-rostock.de:7474/db/data/cypher
-H "Content-Type: application/json" > resultHttp.json
```

7.4 Script For Conversion From Json To Dot

Listing 5: PHP script to convert json file format to dot format

```
<?php

$json = json_decode (file_get_contents ("WithoutHttp.json"));
$returns = $json->data;

echo "graph { \n";
foreach ($returns as $r)
{
        #var_dump ($r);

        echo $r[0]->start . " [label=\"SBML_REACTION\"];" . "\n";
        echo $r[0]->end . " [label=\"SBML_SPECIES\"];" . "\n";
        echo $r[0]->start . " -- " . $r[0]->end . " [label=\"
                                                " . $r[0]->type . "\"];" . "\n";

        #break;
}
echo "} \n";

?>
```

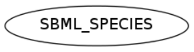
7.5 Examples Of Cypher Queries

Listing 6: Examples of Cypher queries for the key figure analysis

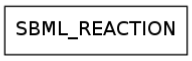
```
Number of models:
MATCH (model:MODEL)
RETURN COUNT(model) AS modelsCount

Number of models with reactions (only SBML-models have reactions):
MATCH (sbmlModel:SBML_MODEL)
WHERE sbmlModel-[:HAS_REACTION]->()
RETURN COUNT(sbmlModel) AS modelsWithReactions

Number of existing reactions:
MATCH (reaction:SBML_REACTION)
RETURN COUNT(reaction) AS reactionsCount

Number of models that have a certain number od reactions:
MATCH (model:MODEL)-[hasReac:HAS_REACTION]->()
WITH model, COUNT(hasReac) as numOfReactions
RETURN COUNT(model) AS numOfModels, numOfReactions

Number of reactions that have a certain number of reactants:
MATCH (reaction:SBML_REACTION)-[hasReac:HAS_REACTANT]->()
WITH reaction, COUNT(hasReac) as numOfReactants
RETURN COUNT(reaction) as numOfReactions, numOfReactants

Number of reactions that have a certain number of modifiers:
MATCH (reaction:SBML_REACTION)-[hasMod:HAS_MODIFIER]->()
WITH reaction, COUNT(hasMod) as numOfModifier
RETURN COUNT(reaction) as numOfReactions, numOfModifier

Number of species that take part in a certain number
of reactions as reactants:
MATCH (species:SBML_SPECIES)-[:IS_REACTANT]->(reaction:SBML_REACTION)
WITH species, COUNT(reaction) AS numReactions
RETURN COUNT(species), numReactions

Number of species that take part in a certain number
of reactions as reactants, products and modifiers:
MATCH (species:SBML_SPECIES)-[:IS_REACTANT]->(reaction:SBML_REACTION)
WHERE species-[:IS_PRODUCT]->reaction
        AND species-[:IS_MODIFIER]->reaction
WITH species, COUNT(reaction) AS numReactions
RETURN COUNT(species), numReactions
```

7.6 Patterns Found

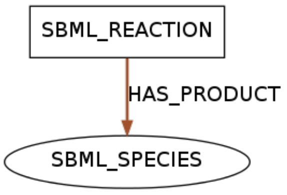
The following presents the full list of results from searching frequent structures within the models. All subgraphs could be found, which occur in at least 250 of the 453 input graphs. The graphs can have vertices for species (oval) and reactions (rectangle). The vertices can be connected by edges representing a relation for a reactant (yellow), product (brown) or modifier (blue). Each pattern is a connected graph. The number of graphs, in which a pattern occurs, is specified for each pattern on its right-hand side.



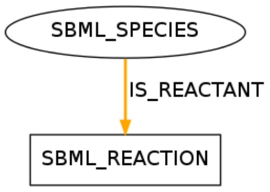
453



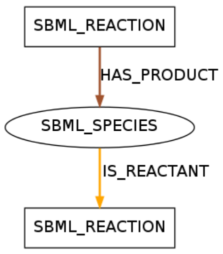
453



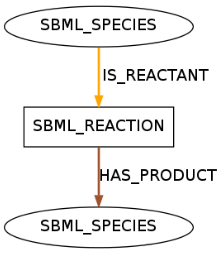
452



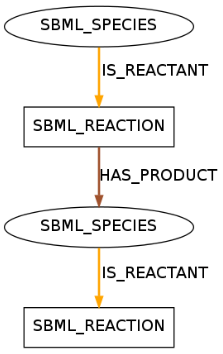
435



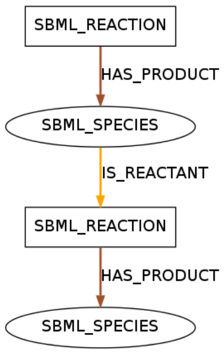
427



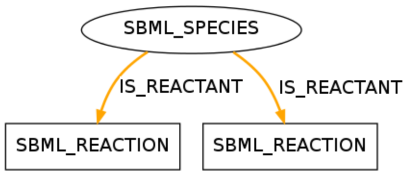
377



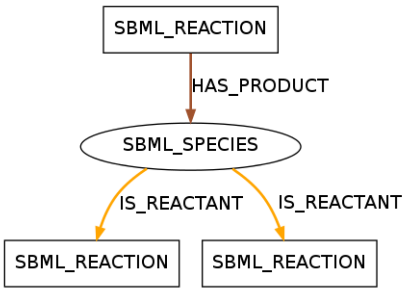
368



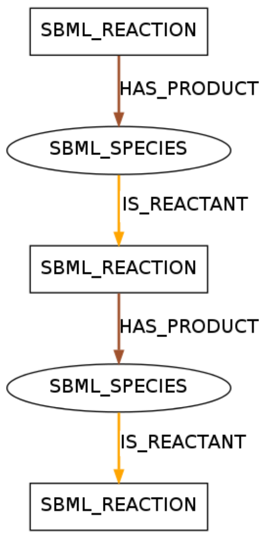
366



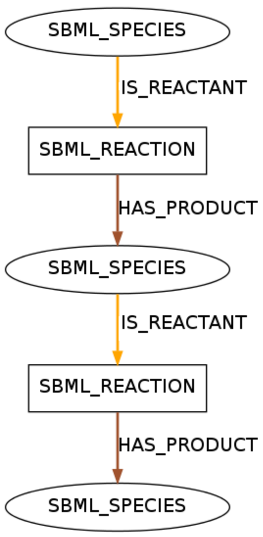
357



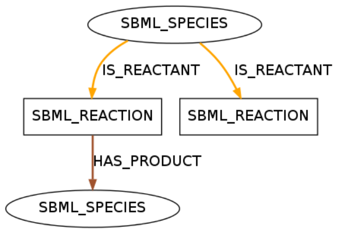
346



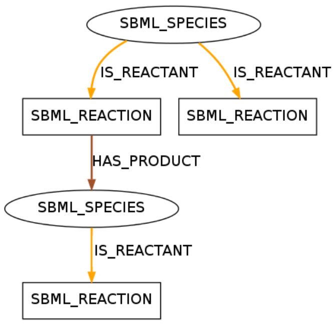
337



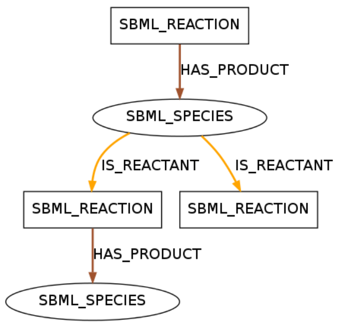
323



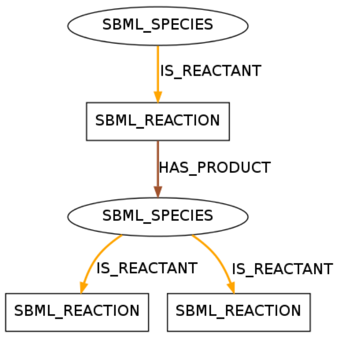
329



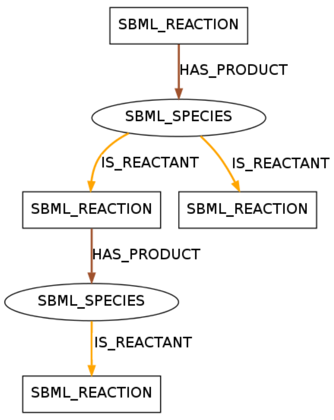
319



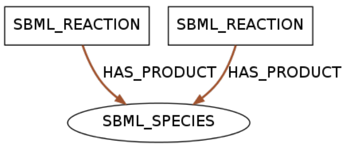
318



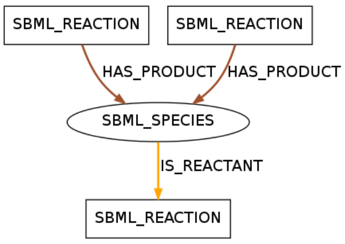
314



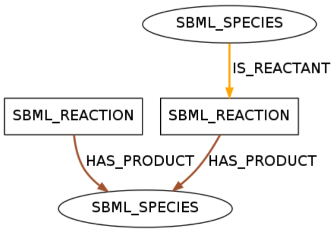
301



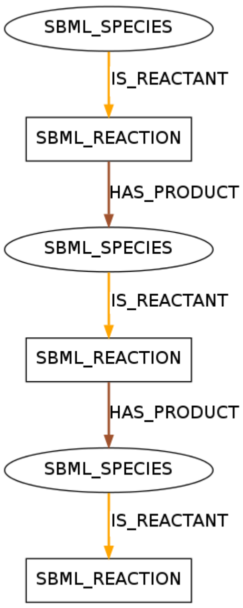
325



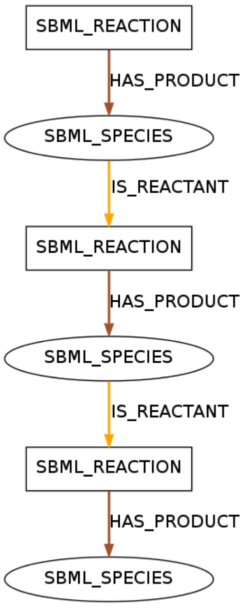
307



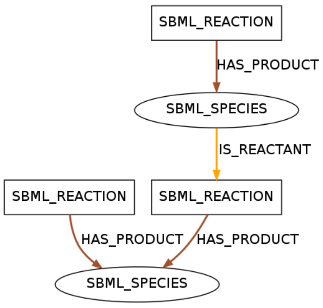
305



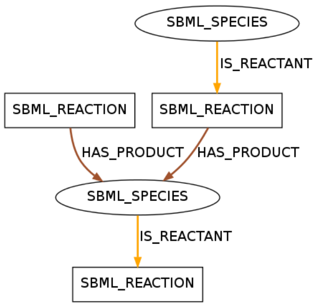
305



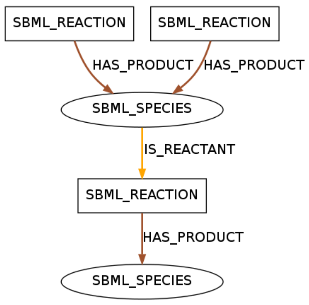
300



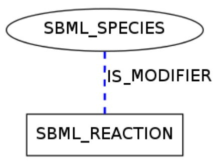
299



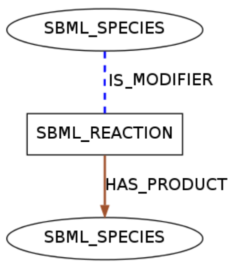
287



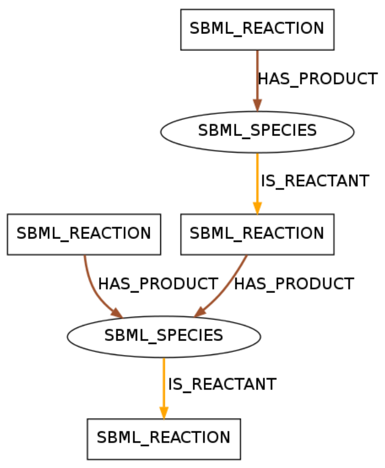
280



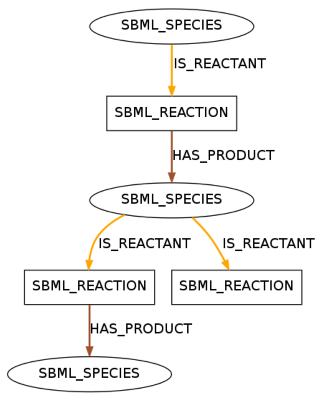
297



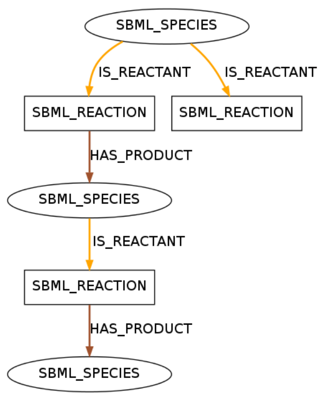
281



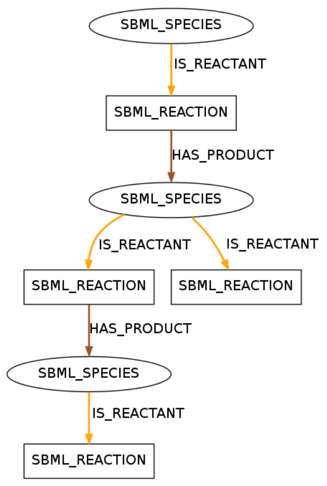
273



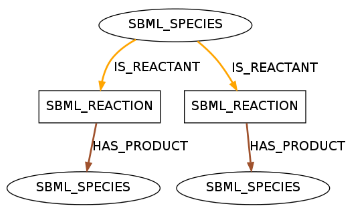
288



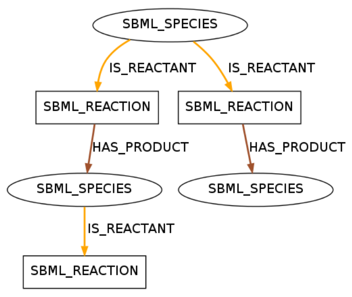
274



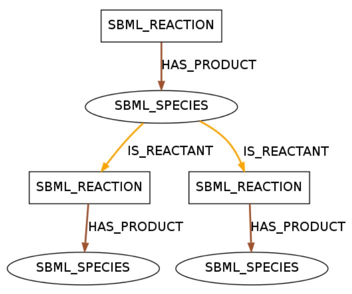
274



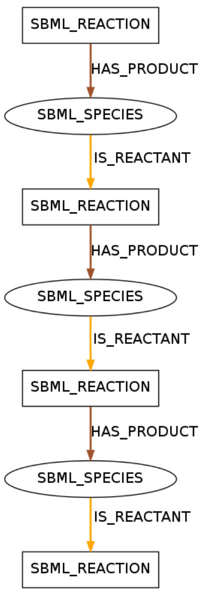
283



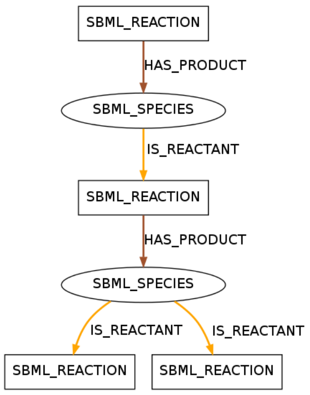
273



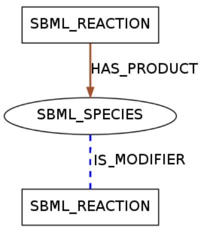
272



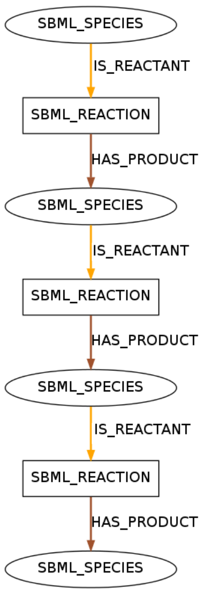
278



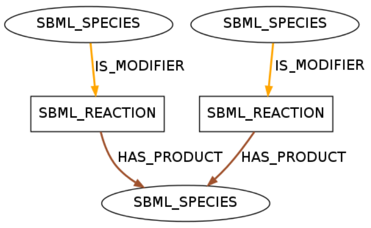
271



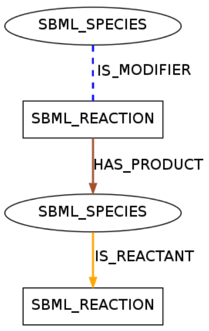
270



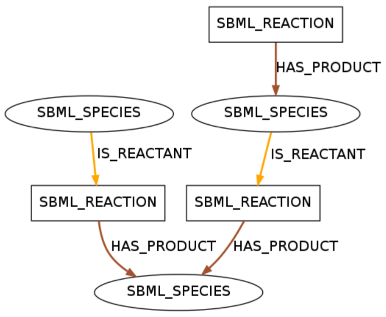
269



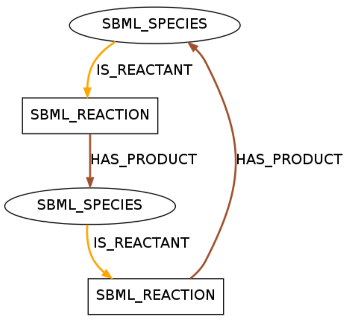
266



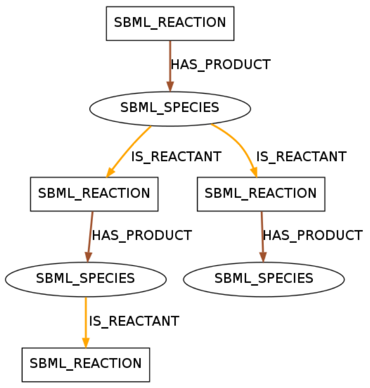
266



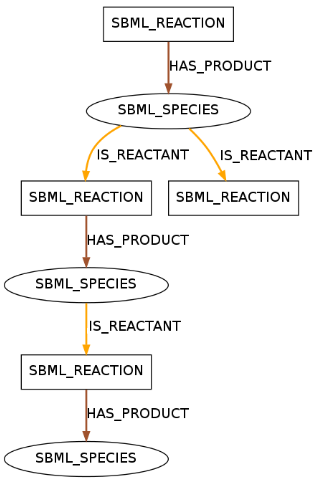
264



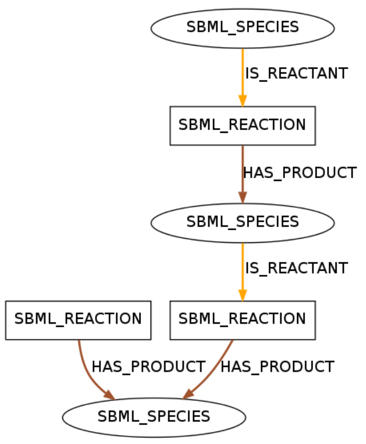
262



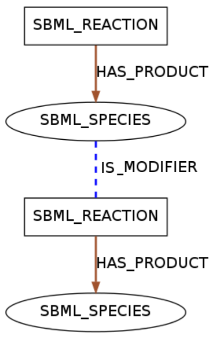
259



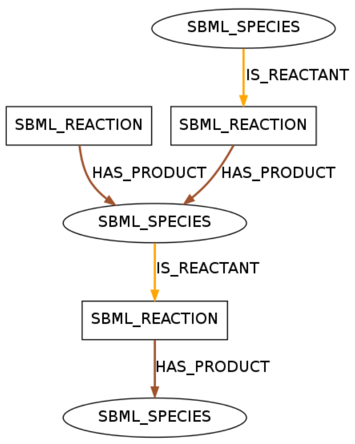
258



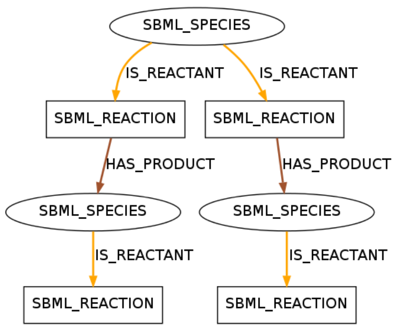
256



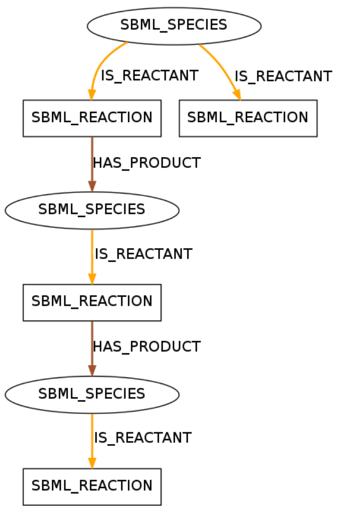
254



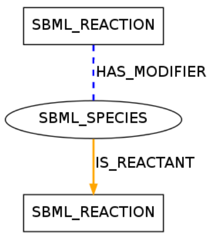
254



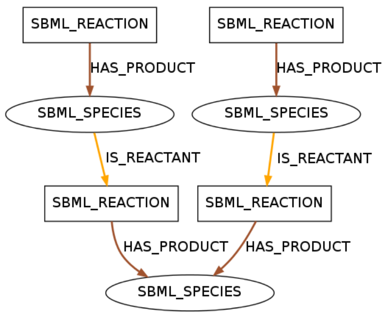
254



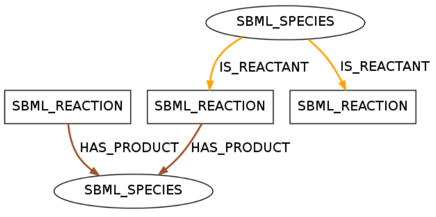
254



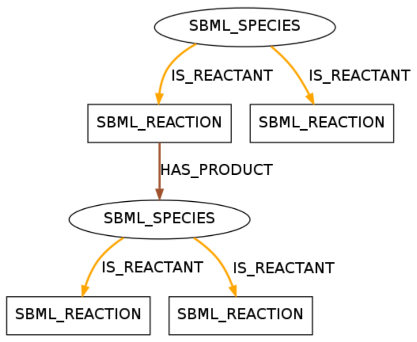
253



252



250


SBML_SPECIES
IS_REACTANT
IS_REACTANT
SBML_REACTION
SBML_REACTION
HAS_PRODUCT
SBML_SPECIES
IS_REACTANT
IS_REACTANT
SBML_REACTION
SBML_REACTION

250

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.
Ich versichere, dass alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche gekennzeichnet sind.
Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt worden und auch nicht veröffentlicht worden.

Rostock, den 07.04.2015