FAKULTÄT FÜR !NFORMATIK

Data Models, Graph Analysis, and Information Retrieval from Biological Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Software Engineering & Internet Computing

ausgeführt von

Raul Fechete
Matrikelnummer 0225871

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Ao.Univ.-Prof. Dr.techn. Rudolf Freund
Univ.Doz. Dr. Bernd Mayer

Wien, 20.02.2009

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Technical University of Vienna

Data Models, Graph Analysis, and Information Retrieval from Biological Data

A Master Thesis Submitted to the
Faculty of Computer Sciences
in Candidacy for the Degree of
Diplom Ingenieur (Dipl.-Ing.)

by
Raul Fechete

supervised by
Ao.Univ.-Prof. Dr.techn. Rudolf Freund
Univ.Doz. Dr. Bernd Mayer

Vienna
February 20, 2009

Abstract

With the dawn of the *omics revolution* a new paradigm was born in molecular biology: the extraction of causal dependencies from descriptive data, building the foundation of a new discipline - computational systems biology. With it, a shift of perspectives took place, pushing the research focus away from data gathering, and towards information mining and assessment.

This thesis introduces a possible approach to information extraction from a biological context. By combining different omics data types, we construct a molecular dependency network which we then use as basis for our novel information mining strategy.

A well-structured graphical representation of functional dependencies between genes, mRNAs and proteins can significantly improve the understanding of the cell functioning and, by extension, optimize hypothesis generation for laboratory experiments, with the ultimate goal of disease-associated biomarker and drug target discovery.

We present a three-stage method involving data warehousing, scientific information consolidation, as well as analysis and visualization. Finally, we investigate the method by running tests using specific datasets from B-cell lymphoma and renal transplant ischemia reperfusion injury, and elaborate on the obtained results.

Zusammenfassung

Mit dem Anbruch des Omics-Zeitalters wird in der Molekularbiologie ein neues Paradigma geboren, die Extraktion von kausalen Zusammenhängen aus deskriptiven Daten, das, in weiterer Folge, als Grundkonzept für ein neues Forschungsgebiet dient - die Computational Systems Biology. Zur selben Zeit findet ein Perspektivenwechsel statt, der den Forschungsschwerpunkt von der Datensammlung auf die Datenauswertung verschiebt.

Diese Diplomarbeit stellt einen neuen Ansatz zur Wissensgenerierung vor. Durch die Kombination verschiedener Omics Datentypen bauen wir ein molekulares Abhängigkeitsnetzwerk auf, das als Basis für unser neues Wissensableitungsverfahren dient.

Eine gut strukturierte grafische Darstellung der Abhängigkeiten zwischen Genen, Transkripten und Proteinen kann das Verständniss der zugrunde liegenden Prozessen wesentlich verbessern und, in weiterer Folge, die Hypothesengenerierung für Laborexperimenten optimieren.

Wir stellen ein dreistufiges Verfahren vor, das aus Data Warehousing, Informationskonsolidierung und -analyse besteht. Abschließend untersuchen wir unsere Methode mit Datensätzen aus der B-Zell Lymphom- und ischämischer Reperfusionsschadensforschung.

Acknowledgments

I would like to thank Dr. Bernd Mayer for making this thesis possible with both his expert advice and financial support.

My full gratitude also goes to all my emergentec colleagues for their invaluable help and the great working environment.

Contents

List of Figures

List of Tables

Listings

List of Algorithms

Chapter 1

INTRODUCTION

1.1 General Background

This first chapter aims at getting the reader acquainted with the basic concepts behind *systems biology* and *omics*. In order to better outline the goals of this thesis, we will also provide the reader with a brief overview of cell protein biosynthesis as well as of the concept of *biomarkers*. Finally, we will state the goals and scope of the thesis and shortly describe the methods employed therein.

1.1.1 Systems Biology / Omics

While the first attempts at studying life under all its aspects followed a reductionist approach, systems biology was born as a life science only after the academic community realized that a biological system is in fact more than just the sum of its components.

As a result, the notion of *emergence* [1] was introduced in this context to describe the properties of a biological entity that only become obvious when viewing the latter as an integrated whole. One of the main goals of systems biology thereby, is to analyze the emergent properties by pursuing an integrative approach [2, 3].

In [4], Goel et al. provide a qualified comparison between the *standard* and the - later introduced - *computational* systems biology: while the former was born as a paradigm shift from the reductionist approach, the latter arose to tackle the problem of the ever increasing amount of available data and provide new analysis means including, among others, information discovery through data and literature mining, and experimental testing of hypotheses generated through mathematical and computational modeling.

We can identify two types of models: *qualitative* and *quantitative*. While the former describe a cellular status quo based on experimental results, the latter, like the E-Cell [5], aim at allowing precise, whole cell simulations at the molecular level [6]. Due to the extreme complexity of the targeted systems, however, they remain mostly in the area of toy systems. For more information on qualitative models, please see [7].

The computational systems biology attempts to create working models of en-

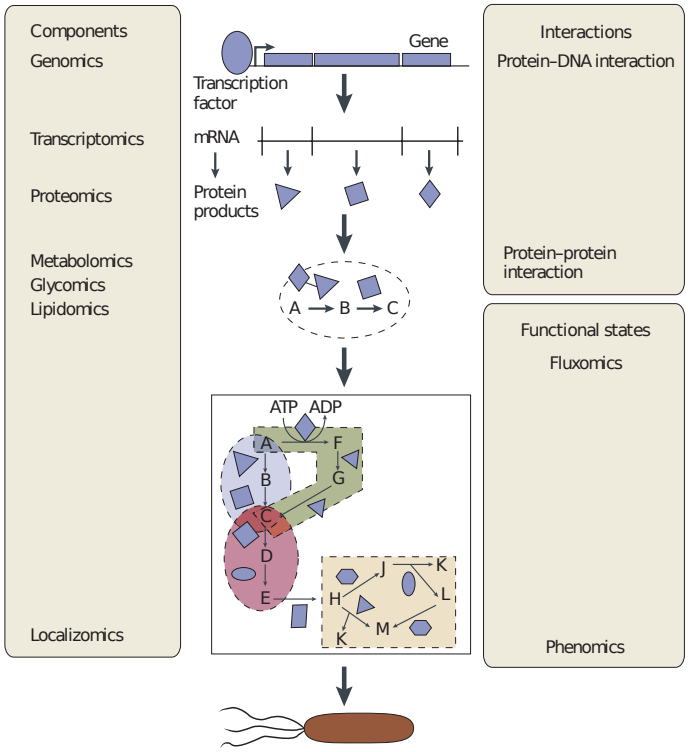


Figure 1.1: The omics types [11]

tire biological systems [8, 9] by, for example, processing omics data acquired through three areas of expertise [10]:

1. high-throughput interrogation technologies,

2. increasingly comprehensive databases of biomolecules and their interactions,

3. computational predictions of molecular function and interaction.

We have, however, yet to provide the reader with a definition for *omics*.

The term itself is a generic name for an entire class of experiments focusing on different aspects of cellular biology. Figure 1.1 provides an overview of the main omics data types, as presented in [11] by Joyce and Palsson.

The work described in this thesis is settled mainly in five of the listed fields. The basic category is *genomics*, concentrating on the study of the human genome and the identification of genes. The analysis of the gene expression is handled by *transcriptomics*. *Proteomics* is focused on determining the protein levels in the cell, while *localizomics* provides information on their subcellular location. *Interactomics* aims at studying the interactions between DNA, RNA, proteins and other molecules and the resulting consequences.

For a better understanding of this area of study, we will attempt to follow the path of the genetic information next, beginning from the deoxyribonucleic acid (DNA) and reaching as far as its manifestation as cell phenotype.

1.1.2 The Cell

When we say *cell*, we almost always refer to the fundamental unity of life. The cell can be prokaryotic or eukaryotic. It can be as small as one micrometer in diameter or as large as one meter in length. It can be epithelial, muscular, nervous, etc. It can have a fixed position inside a tissue or it can be mobile, flowing with the bloodstream. It can be benign or malign (if part of a tumor). The list could go on endlessly, emphasizing the variety of the entity we call *cell*.

A careful reader will notice that all properties previously mentioned are part of the cell's *phenotype*. Since both the structural and functional blueprints of the cell are encoded in the DNA, the following question arises: how does descriptive information stored in the genome come to manifest itself as an observable property?

From DNA to Phenotype

In spite of a certain residual risk of oversimplifying the issue, it is, however, still safe to assume that the majority of the phenotype traits emerges from the interaction networks forming between genes, transcripts, proteins and other molecules present in the cell.

The road from DNA to proteins, on the other hand, as presented for the first time 1958 in [12] by Francis Crick, is the central dogma of the molecular biology. It describes the information transfer from DNA to RNA, and from RNA to proteins [13, 14]. We will have a closer look at this transformation process next.

The basic building blocks of the deoxyribonucleic acid are the four nucleotides: adenine (A), guanine (G), cytosine (C) and thymine (T). A single DNA strand is a nucleotide polymer built along a sugar-phosphate backbone. The DNA molecule consists of two antiparallel DNA strands held together by hydrogen bonds, rotating around an imaginary axis and forming the well-known double helix. The genetic information is encoded in the nucleotide sequence.

The protein synthesis takes place in two phases. The first one is the *transcription* and the second one is the *translation*.

During transcription, the *RNA-polymerase* [16] enzyme binds to certain sections of the DNA molecule and begins moving downstream ($3' \rightarrow 5'$), successively unwinding the two polynucleotide chains and producing a complementary copy of the main strand. In some cases, the binding of the polymerase is facilitated by proteins called *transcription factors*.

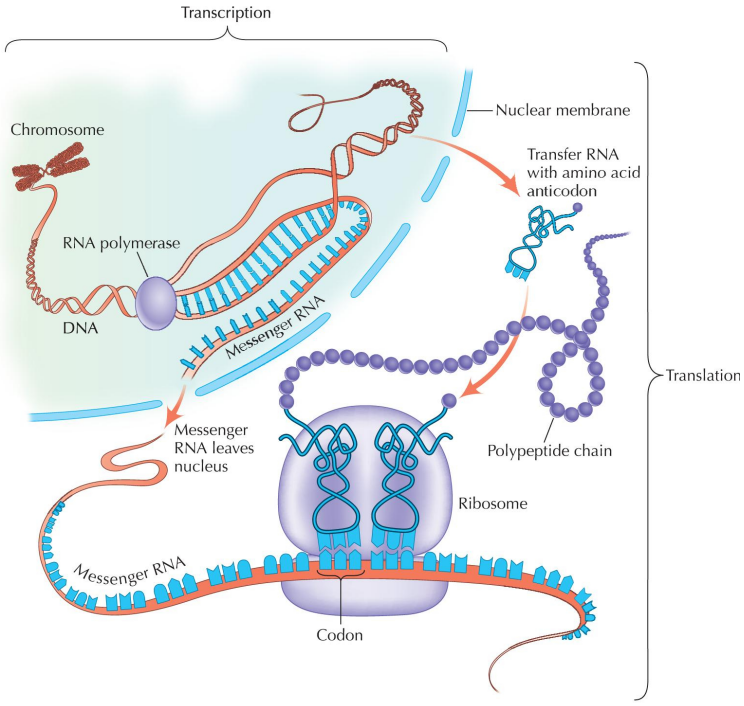


Figure 1.2: The protein biosynthesis [15]

The resulting *primary transcript* is then spliced and reassembled yielding the final *messenger RNA* (*mRNA*). During the reassembly process, sections of the transcribed mRNA are being removed (the *introns*) while the remaining ones (the *exons*) are being joined back to a shorter RNA strand.

In the translation phase, amino acids are being linked together in the ribosome, in an orderly fashion, according to the mRNA transcript, thereby producing the polypeptide chain of the final protein.

Having introduced the players we can now put them into context. We learned that systems biology strives to fully understand the processes taking place in the cell and that the information necessary to achieve this goal is won largely through different omics techniques. We have also seen that the cell phenotype and functionality emerge, among others, from the interactions between various cell molecules, like RNA, proteins, and so on.

The interactions themselves, however, are not isolated but aggregate to entire functional pathways responsible for cell proliferation, cell death, etc. Often, a molecule is involved in more than one process, stripping its mere identification in the cell of any additional information on the underlying mechanisms. In some cases, however, measuring an under- respectively over-expression of a molecule is considered to be clear evidence for a certain cellular process. This type of actors are referred to as *biomarkers*.

Cellular processes can also be disease-associated, with their corresponding biomarkers playing a special role in the clinical diagnosis of the illness. In the next section we will have a closer look at this type of molecules.

Disease-Associated Biomarkers

Before proceeding with an informal definition of a *disease-associated biomarker* we will provide the reader with a brief description of a *gene expression experiment*.

In the previous section we have seen that during transcription and translation mRNA and proteins are being synthesized. For a better understanding of the cell's functionality, it is also of high informational value, how much of each certain mRNA type is available in the cell at a certain moment. Gene expression experiments are quantitative methods developed specifically to assess this kind of information.

Since a direct quantification of the protein levels poses certain technical difficulties, the experiments are limited to measuring the transcript concentrations, which are considered to be in direct correlation with the former due to the mRNA $\rightarrow$ protein translation.

Because of the extreme complexity of the biological processes taking place in the cell, it is usually difficult to postulate a simple rule without likewise accepting the risk of finding inconsistencies in a number of cases. Nevertheless, abstracting beyond and in spite of possible biological exceptions or experiment-induced noise, we assume that healthy cells of a given type, without being exposed to different artificial external stimuli, exhibit similar gene expression patterns regardless of the sample source.

Based on the previous considerations, gene expression experiments are used to determine statistically significant differences in the concentration values of proteins between samples with different background, like, for example, diseased and healthy tissues.

Diseases can often be traced at the gene expression level, with the deviating mRNA concentrations being either a cause or a result of the illness. In an ideal environment, cells of the same population type, affected by the same disease would also exhibit similar gene expression patterns. This is, however, rarely the case in real life. Different experiment conditions, strong experiment noise, or simply, poor understanding of the cellular processes, all work against a clearly defined expression pattern-disease association.

In light of these considerations, a *biomarker* associated with a certain disease is a molecule present in the cell, a deviation in the concentration of which has been generally accepted as an indicator for the presence of that disease, on the basis of a verified biological explanation. Examples for biomarkers can be found in [17] for chronic kidney disease.

As we will see in the next section, biomarkers and specifically their prediction is also one of the central topics handled in this thesis.

1.2 Thesis Overview

This section attempts to provide the reader with an overview of the thesis and the motivation behind it, as well as a quick glance at the employed techniques.

1.2.1 Prologue: Descriptive Data vs. Causal Dependencies

During the last decade, the science community has witnessed a growing informational avalanche originating in the plethora of available techniques for experimental biological data retrieval. The trend was boosted once again in 2001 by the publication of the sequenced human genome [18] and has been increasing ever since with results acquired through high throughput approaches like the aforementioned gene expression assays, protein-protein or protein-DNA interactions analysis and mass spectrometry experiments coupled with two-dimensional gel electrophoresis [19].

One major disadvantage of the gathered data is the fact that it is purely descriptive. While tackling the scientific question of *what*, it provides no answer to the functional *why*. Both are fundamental for the understanding of the cell's functionality, but the leading role is still taken up by the need to comprehend the causal dependencies linking the actors (proteins, genes, etc.). Let us consider the specific case of a disease study at molecular level. A gene expression experiment may provide us with a list of differentially regulated genes as potential therapeutical targets, but without a thorough understanding of the functional connections between them, no drugs can be developed.

This brings us to the goals of this thesis.

1.2.2 Scope and Goals

The main goal of this thesis is to improve methodologies for information gain from existing biological data. We can identify several areas of interest.

The major focus is dedicated to the deduction of causal dependencies at the molecular level. Given the descriptive data obtained through various omics experiments as input, the developed application shall provide assistance mechanisms for predicting functional connections between genes, mRNA and proteins. In this context, the goal of the system is the extraction of emergent biological information from different, interlinked omics fields.

A specific use case of the system in the medical field and a subsequent goal is the identification of disease-associated genes and biomarkers. The newly acquired

knowledge is then to be used in the prediction of therapeutical targets for medical drug development.

By combining data from different omics experiments, our software constructs molecular networks and visualizes them as graphs. A well-structured graphical representation of a molecular interaction network can significantly improve the understanding of the cell functioning and, by extension, optimize hypothesis generation for laboratory experiments.

1.2.3 Method Outline

We will now attempt to provide the reader with an overview of the methods employed to achieve the stated goals.

From a technical point of view, the constructed system represents a comprehensive framework for biological data integration and analysis. We can identify three main sections.

Data warehousing

The first stage of the analysis workflow comprises the data integration process. At the base of our system lies an extensive data warehouse, currently incorporating sections from over ten databases provided by renowned international biological research organizations, like the European Bioinformatics Institute (IPI, Gene Ontology, IntAct, UniProt), the National Center for Biotechnology Information (GenBank, RefSeq), the Ontario Cancer Institute and University of Toronto (OPHID), the Kyoto Encyclopedia of Genes and Genomes (KEGG) and several others.

To prevent any artificial biasing during the import phase, the data is stored with the corresponding identifiers of the source databases, organized around a global accession mapping provided by the International Protein Index of the European Bioinformatics Institute.

The ease of accessibility and maintenance of the system is ensured by the employment of an object-relational data mapping backed up by an associated object persistence layer, in our case, Hibernate.

Scientific data consolidation

During the second phase, we consolidate the gathered information in four functional categories, and calculate, for each individual pair of proteins, a dependency score based on the data available for the two entities in each respective class. A fifth category, the *transcription factors* has also been modeled, but will only be used in the next version of the system. Please refer to section 4.3 for further details.

We differentiate between the following informational classes:

- *Protein interactions*

  In this section we gather information about experimentally confirmed or computer predicted molecular interactions between the two actors.

- *Functional pathways and biological ontologies*

  This category contains information on the two corresponding genes' dependency on the functional level, i.e., if the two entities both occur on the same pathways or in the same ontology terms, etc.

- *Subcellular location*

  The subcellular location section consolidates information on the presence of the two proteins in the various cell components, like the nucleus, the golgi apparatus, etc.

- *Gene expression sheets*

  This section holds information on the expression of the two mRNAs across different tissue types.

- *Transcription factors*

  In this category we gather information about relationships between transcription factors and the genes the expression of which they regulate.

The four values are then used as input parameters for a dependency function asserting the relation strength between the two actors in the continuous interval $[-1, 2]$, from weakest (-1) to strongest (2).

Data analysis and visualization

In the third phase, we construct a complete dependency graph using the biological entities (genes, proteins, etc.) as nodes and the relation strength values obtained in the previous step as edge weights.

The new data representation form allows an intuitive display of the emergent information. One possible analysis scenario, based on the rationale that nodes bound by heavier edges share stronger functional dependencies, would be to apply a layouting algorithm on the resulting graph that clusters nodes linked by heavier edges closer together. In that way, we can provide visual feedback on proteins and genes with a higher probability of working together in the cell.

Independently from the previous example, we can introduce the notion of *neighborhood* of a biological entity, as a set of nodes and their interlinking edges selected on the basis of a given criteria. The concept of neighborhood is also

the main tool used to achieve the goals stated in the previous section. Let us consider a specific example. Given a list of genes, possibly determined in a gene expression experiment, the typical use case would be to analyze their interactions in the context of the calculated dependency graph. To achieve this, we define three types of neighborhood selection criteria, each with its own rationale:

1. *Single maximum value paths*

   Given a set of biological entities, i.e., a set of nodes in the graph, a trivial approach to determining the dependency networks spanning between them is to calculate the closure over the maximum value paths linking them.

   Let e_{ij} be an edge between two arbitrary nodes i and j and $\omega(e_{ij})$ its weight. Let $P(x, y)$ be the set of all possible paths between two given vertices x and y and $\omega(p)$ the weight of an arbitrary path p from $P(x, y)$, defined as follows:

   $$\omega(p) = \sum_{e_{ij} \in p} \omega(e_{ij})$$

   $$P(x, y) = \{\ p \mid p = (e_{xi_1}, e_{i_1 i_2}, .., e_{i_{n-1} i_n}, e_{i_n y})\}$$

   We define the maximum value path p_{max} between the given vertices x and y as a path for which the following constraint holds:

   $$p_{max} \in P(x, y) \wedge \omega(p_{max}) \geq \omega(p),\ \forall p \in P(x, y)$$

   Depending on the ratio of selected versus maximum possible edges in the analyzed subgraph, we employ either the *Dijkstra* [20] or the *Floyd Warshall* [21] algorithm for the calculation of the paths. To be able to apply a shortest path algorithm to a maximum value path problem, we invert the edge weights, i.e., we use $2 - \omega(e_{ij})$ instead of the original relation strength.

2. *Variant multiple maximum value paths*

   Since several different maximum value paths may exist between two vertices, but both algorithms mentioned in the previous section can only identify one, we must develop a technique that is able to handle multiple maximum value paths as well.

   Taking the issue one step further, it would be of great informational value, from a biological point of view, to not only determine all the maximum value paths but also the paths with an only marginally inferior score.

If we regard the maximum value path as an interlinked set of edges respecting a weight constraint with a lower bound of $\delta \cdot \omega(p_{max})$ for a δ of 1, we can define the set of paths we are looking for as:

$$P_\delta = \{\ p \mid \omega(p) \geq \delta \cdot \omega(p_{max}),\ p \in P(x,y)\ \},\ \delta \in [0,1]$$

In section 5.2.2 we will present an efficient algorithm for determining the δ-variant maximum value paths based on Dijkstra's algorithm and a reversed bounded depth first search (RB-DFS).

3. *Heuristic spanning trees*

   The path-techniques used to build the neighborhood of the selected vertices in the previous sections are well suited for the analysis of the highest ranked functional connections between the actors. Having made this statement, we must also note the fact, that, on the other hand, both techniques ignore certain biological aspects of the network. The nodes situated on the δ-variant paths between two vertices x and y can provide us with an explanation for the dependency between the two, should such a dependency indeed exist. While this is often the case, the data retrieved by these means may also prove misleading if the two entities have no dependency at all.

   To counter this effect, let us take a different approach into consideration. Starting from the assumption that the calculated dependency graph is - as far as the input data allows it - a correct mathematical representation of the reality, it could prove beneficial from a biological point of view to let the neighborhoods grow naturally, beginning from the set of input vertices and *evolving* along the heaviest adjacent edges.

   This new approach is basically the biological visualization of the maximum spanning tree concept from computer science. Having dropped the path-related connectivity constraint of the previous two approaches, this new neighborhood evolution technique can yield various neighborhood topologies and connectivities between the selected actors. We can identify two main behavioral aspects:

   - Hub proteins will develop star topologies, having selected several heavy adjacent edges, while others, like members of signaling pathways, may evolve into elongated components with a low average node degree.
   - While entities with no functional interaction will remain separated in disjoint connected components, the trees of related entities will eventually grow together, merging into a larger connected component.

The latter aspect imposes a certain difficulty on the problem solution since standard maximum spanning tree approaches, like the well-known *Kruskal* and *Prim* [22] algorithms only compute one maximum spanning tree per graph. In our case, however, we need a separate tree for each of the selected vertices.

In section 5.2.3 we will introduce a heuristic approach to this issue based on the aforementioned Kruskal and Prim algorithms.

Chapter 5 is dedicated entirely to the implementation of the three neighborhood expansion methods and the analysis of the B-cell lymphoma and IRI data sets in the context of the dependency graph. It will also present results obtained by each of the introduced techniques.

Chapter 2

BIOLOGICAL DATA AND STANDARDS

The second chapter aims to provide the reader with an overview of the data that has been integrated in our framework, including the corresponding source databases. In the end, we will describe three of the most important file formats used for biological pathway data representation.

2.1 Biological Data

In the introduction to this thesis, we briefly outlined the biological information we use to build the molecular dependency graph: protein interactions, functional pathways, ontologies, subcellular location, gene expression and transcription factors. In this section we will elaborate on the respective data types, their source and the mechanisms employed to link them together.

2.1.1 Identity Mapping

Since the dawn of modern biology, the vast amount of gathered experimental data have forced researchers into developing various informational infrastructures. Considering that many of the ad-hoc solutions evolved in parallel and/or with no coordination between the developers, it is not surprising that different databases emerged, each with its own type of information, constraints, identifiers, etc. A side-effect of this development was that data was often stored redundantly, inconsistently, or was simply omitted altogether.

With the rise of computational systems biology, the need occurred to aggregate different data sources and thereby allow a more comprehensive knowledge mining. In the context of this thesis, when we speak of information aggregation, we refer to the process of merging data from different databases for each individual biological entity (transcript, gene, protein, etc.).

The first obstacle on the path to achieving this goal is set by the different identifier associations. While, in the beginning, each database used its own identifiers, nowadays, the accessions of several leading research institutions have established themselves as de facto standards for this purpose. In spite of this development, it is still often the case that the same biological entity is referred to with different names in different databases, e.g., BCL2 and ENSG00000171791 both denote the same apoptosis regulator.

Molecular level	Accession	Database	Institution
Gene	Identifier	ENSEMBL	EMBL-EBI
	Identifier	Entrez	NCBI
	Symbol	Entrez	NCBI
Protein	Identifier	IPI	EMBL-EBI
	Identifier	UniProt SwissProt	EMBL-EBI
	Identifier	UniProt TrEMBL	EMBL-EBI
	Identifier	RefSeq Protein	NCBI

Table 2.1: Important biological entity accessions

Table 2.1 provides an overview of the most important accession types, as required by our project. Please note that although each database assigns accessions to all its stored biological entities (transcripts, genes, proteins, etc.) we are only interested in the latter two.

On the gene level, we are interested in the identifiers provided by the ENSEMBL [23] database of the European Molecular Biology Laboratory (EMBL) - European Bioinformatics Institute (EBI) [24] and the identifiers and symbols provided by the Entrez [25] database of the National Center for Biotechnology Information (NCBI).

On the protein level, we use the identifiers of the International Protein Index (IPI) [26], the UniProt SwissProt [27, 28], and the UniProt TrEMBL [28], all three provided by the EMBL-EBI. Further, we use the identifiers of the RefSeq Protein [29] database provided by the NCBI.

All the information we integrate in our system is available for at least one of the accession types previously mentioned. Table 2.2 provides an overview of the imported databases, their maintaining organizations, the accessions they use and the type of data they provide. We will present the databases themselves in more detail in the course of the next sections. For now, this information is intended only to better illustrate the problems one encounters when attempting to consolidate data from different sources.

In order to tackle the problem of multiple identifiers per entity, we introduce the concept of *biological hyperstructure*. A hyperstructure is the virtual representation of a physical entity. It holds all identifiers available for that entity grouped by molecular level and accession type. Figure 2.1 depicts a typical hyperstructure as implemented in the current version of our system.

The outer bubble represents the border of the structure. The first classification is done by the molecular level. In the current version, we distinguish only between the gene and the protein levels. This first differentiation provides us with a twofold flexibility:

Source	Institution	Accession	Data type
IntAct	EMBL-EBI	UniProt id.	Protein interactions
OPHID	OCI-UT	UniProt id.	
BioGrid	SLRI & UT	NCBI gene sym.	
KEGG	Kanehisa Labs.	NCBI gene id.	Functional pathways
PANTHER	SRI Intl.	UniProt id.	
Gene Ontology	GOC	NCBI gene id.	Biological ontologies
SwissProt	UniProt	UniProt id.	Subcellular location
Plasma P.P.	HUPO	IPI id.	
BodyMap	NCBI	NCBI gene id.	Gene expression
Internal	EmergenTec	ENSEMBL gene id.	Transcription factors

Table 2.2: The data source accessions

1. Flexibility of *usage*: when working with hyperstructures we can transparently switch between the gene and protein layers, being able to dynamically access the associated information.

2. Flexibility of *expansion*: we can also transparently add new layers to the structure, for example, the transcript layer, without having to modify the existing data.

Inside each layer we further differentiate between the various identifier types. The gene accessions are split into NCBI gene symbols respectively identifiers and ENSEMBL gene identifiers, while the proteins are split into IPI, RefSeq Protein, SwissProt and TrEMBL identifiers.

The dark spots in figure 2.1 depict single accessions. The NCBI identifiers and symbols always come in pairs - hence the one-on-one associations depicted by the arrows. Our hyperstructure provides means to facilitate the retrieval of this information at runtime.

The accessions outside the main bubble symbolize identifiers that have been replaced by newer ones and that are therefore no longer in use. These deprecated names are not associated with a hyperstructure, but instead link to the corresponding, currently valid accession. The decision to do so is based on the rationale that a hyperstructure should always be up-to-date. On the other hand, the reason for allowing each accession to manage its own history and not discarding the deprecated names altogether, is to facilitate the integration of data that is only being provided for older accessions.

The main purpose of the hyperstructure is to provide a mapping between the identifiers of a biological entity, which will then act as basis for the information consolidation. The information blocks depicted in figure 2.1 symbolize the data,

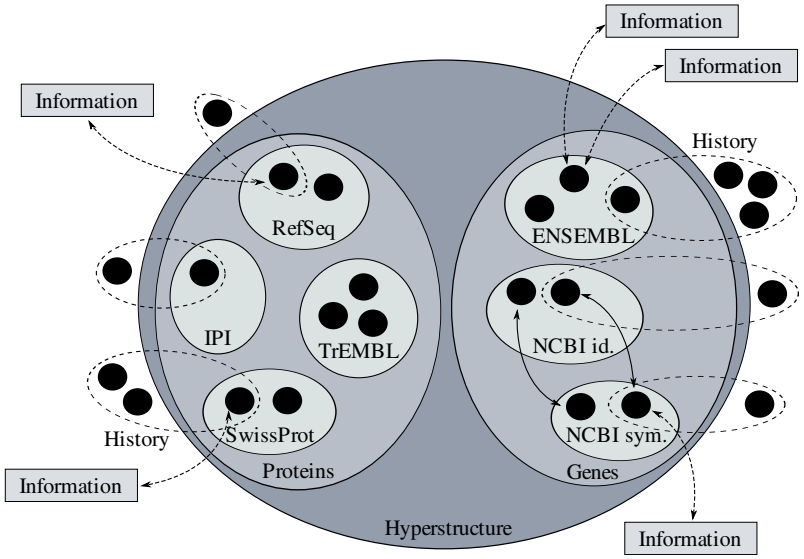


Figure 2.1: A typical biological hyperstructure

M. level	Accession	Primary	History
Gene	ENSEMBL id.	ENSG00000171791	-
	NCBI id.	596	-
	NCBI sym.	BCL2	-
Protein	IPI id.	IPI00020961	IPI00105268
	SwissProt id.	P10415	P10416, Q13842, Q16197
	TrEMBL id.	A9QXG9, Q96PA0	-
	RefSeq id.	NP_000624	-

Table 2.3: A BCL2-associated hyperstructure example

like protein interactions, etc., that is linked to a certain accession in a hyperstructure. If certain information is provided only for a deprecated accession, our system will automatically link it to the currently valid one.

In the context of this thesis, a hyperstructure is tied to a particular protein sequence. This status quo has the side-effect that a protein name is present in only one hyperstructure, whereas gene names may be linked to several structures, provided they have different splice variants.

Table 2.3 shows an excerpt from a biological hyperstructure associated with the aforementioned apoptosis regulator BCL2.

Having introduced the biological hyperstructures, we can now put them in the context of the dependency graph. Since the information is consolidated on a per-hyperstructure basis, it only stands to reason that we also use them as graph

nodes.

In the next sections we will have a closer look at the different information types we aggregate, as well as their source databases.

2.1.2 Protein Interactions

By playing a key role in many of the cell's biological functions, protein-protein interactions are especially important for our molecular dependency graph. Proteins can pass on signals from the cell exterior to the interior through a process called signal transduction. They can transport other proteins from and into the nucleus or they can modify them, e.g., through phosphorylation. Proteins can facilitate cell division or can drive it into apoptosis, for example.

Due to the substantial diversity of the proteins' structure, size and chemical properties, several methods for interaction detection have been developed. While some aim to demonstrate the interactions in a specific experimental setup, other have a probabilistic approach, attempting to computationally predict them based on the aforementioned protein properties.

Two of the most important experimental methods for protein interaction determination are *yeast-two-hybrid* [30] and *co-immunoprecipitation* [31]. In the former, each of the two examined proteins is attached to one half of a transcription factor, for example, GAL4 in Saccharomyces cerevisiae. If the actors bind each other, the transcription factor will be reconstructed, which will further result in an expression of the reporter gene. In the latter technique, an antibody targeting a certain protein is used to *pull* the protein out of a solution together with its associated complex. The result is then analyzed, i.e., the pulled-down proteins are identified where possible. All members of the complex which have been detected at significant levels are considered to have an interaction with the originally targeted protein. Further experimental verification is, however, necessary to remove the possibility of indirect binding.

We imported protein-protein interactions from the following databases:

- The *IntAct* [32] database of the EMBL-EBI contains information obtained entirely through literature mining. The interactions are manually annotated to a high level of detail, including, among others, the experimental confirmations and interacting domains.

- The *Online Predicted Human Interaction Database (OPHID)* [33] of the Ontario Cancer Institute from the University of Toronto is a collection of annotated interactions provided by databases such as BIND [34], HPRD [35], MIPS [36], MINT [37] and DIP [38], as well as predicted interactions made from Saccharomyces cerevisiae, Caenorhabditis elegans, Drosophila melanogaster and Mus musculus.

- The *BioGrid* [39] database of the Samuel Lunenfeld Research Institute and the University of Toronto contains interactions derived from both high-throughput and conventional focused studies, providing full annotation support for thirteen major model organism species.

2.1.3 Functional Pathways

The interactions we described in the previous section are rarely isolated events. Usually, an interaction between two or more molecules will subsequently trigger a second one, which, in turn, will trigger a third one, etc. thereby building entire event cascades. Such sequences serve, in most cases, a well defined purpose and are referred to as *functional pathways*. The apoptosis pathways, inducing programmed cell death, are a well-known example.

Considering that pathways directly represent functional dependencies, they are a core component of our dependency graph. The documented pathways, however, are only a fragment of all existing dependencies in the cell and we will, therefore, try to extend this knowledge by taking emergent information into consideration, obtained through the integration of other data sources, like protein-protein interactions, ontologies, subcellular location, and so on.

Sources for functional pathways are:

- The *Kyoto Encyclopedia of Genes and Genomes (KEGG)* [40] of Kanehisa Laboratories provides, among others, the PATHWAY database containing a collection of manually drawn pathway maps representing the knowledge status quo on the molecular interaction and reaction networks for metabolism, genetic information processing, environmental information processing, cellular processes and human diseases.

- The *Protein ANalysis THrough Evolutionary Relationships (PANTHER)* [41] classification system of SRI International is a collection of protein families subdivided into functionally related subfamilies, using human expertise. The subfamilies model the divergence of specific functions within protein families, allowing more accurate association with function (ontology terms and pathways).

2.1.4 Biological Ontologies

The goal of the *Gene Ontology* [42] classification, created and maintained by the GO Consortium, is to produce a structured, precisely defined, common, controlled vocabulary for describing the roles of genes and gene products in any organism.

The Gene Ontology is structured in three distinct organizational units:

- The *biological process* refers to a biological objective to which the gene or gene product contributes, e.g., signal transduction.

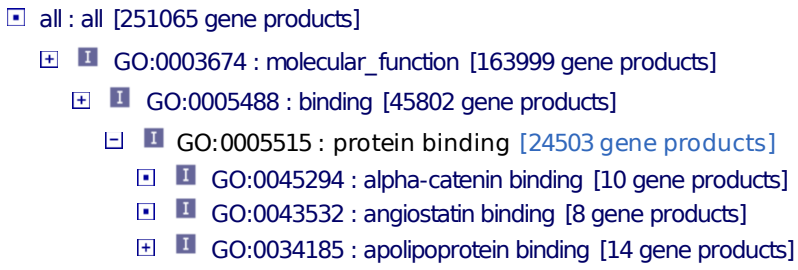


Figure 2.2: An excerpt from the Gene Ontology

- The *molecular function* refers to the biochemical activity of a gene product, e.g., enzyme.
- The *cellular component* describes the subcellular location where a gene product is active, e.g., ribosome.

Figure 2.2 provides an example from GO, depicting the first sub-terms of the molecular function category.

Similar to the pathways, the ontology provides us with information on functional dependencies between molecules, and is therefore an important factor in the calculation of the graph edge weights.

2.1.5 Subcellular Location

In the previous sections, we argued that the existence of a direct interaction between two gene products also indicates, with a high probability, the existence of a functional dependency between these two. This reasoning strategy proves flawed, however, if the two molecules only become active in different subcellular components.

To counter this effect, we will evaluate the interactions in the context of the actors' subcellular location when calculating the edge weights. In order to do so, we need location information on all graph nodes. We obtained such data from the following sources:

- The *SwissProt* [28] database provides high quality, manually curated, both experimentally obtained and computed protein-related information. Among others, the database also provides information on the subcellular location of the described proteins.
- *WoLF PSORT* [43] is an algorithm for protein subcellular location prediction. Based on special properties of the analyzed protein's amino acid sequence, it computes a ten-valued output vector, indicating the probability of the protein's presence in ten different subcellular components.

- The *Plasma Proteome Project (PPP)* [44] of the Human Proteome Organisation (HUPO) provides, among others, a list of proteins the presence of which has been detected in plasma.

2.1.6 Gene Expression Profiles

Aside from the subcellular location, an important role in the dependency prediction is also played by the genes' expression within various tissue types. The rationale is that molecules sharing a functional dependency have to be expressed in the cell at the same time. Furthermore, since in most cases cells aggregate forming tissues, it stands to reason that the same actors have to be expressed across an entire cell population originating from the same tissue.

Information on genes' expression across 32 human tissue types can be obtained from the Gene Expression Omnibus dataset GSE7905 [45].

2.1.7 Transcription Factors

In the introduction to this thesis, during the description of the protein biosynthesis, we briefly mentioned the transcription factors. By facilitating the binding of the RNA-polymerase to the DNA, the transcription factors are directly responsible for a gene's expression, a relation which is basically another form of functional dependency.

The detailed description of a mechanism for in-silico prediction of transcription factor binding site in the human genome can be found in [46]. The database containing information obtained by the technique previously mentioned is generated in-house.

2.2 Pathway Representation Standards

While certain biological data types such as gene expression and subcellular location are relatively easy to handle, types like pathways and, generally, quantitative models share an intrinsic complexity which makes them much more difficult to store and transfer across different application platforms.

Furthermore, with the ever increasing amount of available biological data, the need for a standardized information representation format became more and more obvious. Such a unified mechanism would facilitate the data transfer between applications in bioinformatics and prevent data loss through obsolete software.

During the years, several representation standards for biological pathway data have been developed and, in the course of the following sections, we will have a closer look at three of them. A good overview of the same three standards can also be found in [47].

2.2.1 SBML

In [48], Hucka et al. provide a well-structured overview of the rationale behind the *Systems Biology Markup Language (SBML)*. The first reason named is the need to work with complementary resources from different software tools. Each program has its own strengths and weaknesses and optimal results are achieved by employing each tool for what it is best suited for. Having a data representation standard supported by all programs would spare time and eliminate the risk of errors induced by information re-encoding. Further, the use of a common language would facilitate the reuse of models published in peer-reviewed journals and last but not least, it would prevent models from becoming stranded once the systems they have been created in become obsolete.

SBML has been developed to address the previous issues, but instead of trying to define a universal language for all quantitative models, the authors concentrated on building a common intermediate format, a *lingua franca* as they call it, enabling the transfer of the most essential model aspects.

Defined as an XML-based machine readable format, SBML is being developed in levels, with each extra level providing means for the representation of additional information. Currently, two levels are available, with the third already being in development.

An overview of Level 1 can be found in [49] with a brief description of the features available in Level 2 in [50]. The first level breaks down a chemical reaction in six conceptual elements: *compartments* are finite volume containers, *species* are entities that take part in a reaction, *reactions* describe transformations, transports or binding processes, all with their associated rate laws, *parameters* are quantities with associated symbolic names, *unit definitions* are used, for example, to set abbreviations for default unit combinations, and *rules* are mathematical expressions used, for example, to establish constraints between quantities.

Some of the most important differences between Level 1 and 2 include among others: in Level 2, *mathematical formulae* have been changed from plain text to MathML, the possibility of adding *metadata* and defining new *mathematical functions* has been implemented, and support for *delay functions* and *discrete events* has been included.

The elements described previously are the basic building blocks of an SBML file. Their definition was deliberately kept generic, in order to allow a better coverage of the various systems biology models, and provide the applications using them with a greater interpretation flexibility.

Put in a nutshell, SBML promotes a horizontal information representation architecture for biological models, providing, among others, means for the description of actors, reactions and associated stochastic parameters.

SBML is supported by a wide array of programs, including: Cellerator [51],

```
<model name="Example">
<listOfCompartments>
  <compartment name="Mithocondrial Matrix" id="MM"/>
</listOfCompartments>
<listOfSpecies>
  <species name="Succinate"  compartment="MM" id="Succinate" />
  <species name="Fumarate" compartment="MM" id="Fumarate" />
  <species name="Succinate dehydrogenase"
           compartment="MM" id="Succdeh" />
</listOfSpecies>
<listOfReactions>
  <reaction name="Succinate dehydrogenas catalysis" id="R1">
    <listOfReactants>
      <speciesReference species="Succinate" />
    </listOfReactants>
    <listOfProducts>
      <speciesReference species="Fumarate" />
    </listOfProducts>
    <listOfModifiers>
      <modifierSpeciesReference species="Succdeh" />
      <modifierSpeciesReference species="S4" />
    </listOfModifiers>
  </reaction>
</listOfReactions>
</model>
```

Figure 2.3: SBML example [47]

```
<entry>
<interactorList>
 <proteinInteractor id="Succinate>
    <names>
      <shortLabel>Succinate</shortLabel>
      <fullName>Succinate</fullName>
    </names>
 </proteinInteractor>
 ....
</interactorList>
<interactionList>
  <interaction>
    <names>
      <shortLabel> Succinate dehydrogenas catalysis </shortLabel>
      <fullName>Interaction between ....</fullName>
    </names>
    <participantList>
     <proteinParticipant>
       <proteinInteractorRef ref="Succinate"/> <role>neutral</role>
      </proteinParticipant>
     <proteinParticipant>
       <proteinInteractorRef ref="Fumarate"/><role>neutral</role>
     </proteinParticipant>
     <proteinParticipant>
       <proteinInteractorRef ref="Succdeh"/><role>neutral</role>
     </proteinParticipant>
    </participantList>
  </interaction>
</interactionList>
```

Figure 2.4: PSI-MI example [47]

DBsolve [52], E-CELL [53], Gepasi [54], Jarnac [55], NetBuilder [56], StochSim [57] and VirtualCell [58]. An open source implementation of the SBML format is provided by the libSBML library [59].

Currently, some of the most important databases that provide their data in SBML format are KEGG [40], EcoCyc [60] and Reactome [61].

Figure 2.3 displays a short excerpt from an SBML file.

2.2.2 BioPAX

An approach completely different from SBML, is the *Biological Pathway Exchange (BioPAX)* [62] format. Although also defined in levels, the latter promotes a vertical, ontology-based architecture.

The BioPAX representation format is based on a class hierarchy with *entity* as its top structure and *physical entity*, *interaction* and *pathway* as its immediate children. These are then split into further subclasses with the main relation types between them being *is-a* for inheritance and *has* for aggregation. All structures used in BioPAX are declared in this inheritance tree, and, as mentioned previously, pathways are represented as ontologies.

To better visualize the concept, let us have a quick look at a section of the apoptosis pathway. This ontology has a top level node called *apoptosis*. This node has several children, namely the *intrinsic* and *extrinsic* mechanisms, along with nodes describing the overall regulation of the apoptosis pathway. Moving deeper into the extrinsic branch, we encounter two children: *death receptor signaling* and *caspase 8 forming from pro-caspase 8*. The latter is then split into *activation of pro-caspase 8* and *formation of caspase 8 dimer*. The process is repeated for all

branches of the ontology down to the basic molecular interactions.

BioPAX is defined in W3C consortium's Web Ontology Language (OWL, sic) and thereby offers extensive automated reasoning capabilities based on the source language's strong semantic orientation.

Notably, the advantages of a semantic representation scheme come at a certain processing performance price. While powerful, the language itself creates a high computational overhead. The comparably large file size also hinders from providing the reader with a specific example of a BioPAX model.

Presently two BioPAX levels are available, with the third still in development. The most renowned databases providing their pathways in this format are KEGG [40] and Reactome [61].

2.2.3 PSI-MI

Another biological data representation standard promoting a horizontal definition hierarchy is the *Proteomics Standards Initiative - Molecular Interaction (PSI-MI)* [63] XML format of the Human Proteome Organization (HUPO). Designed initially only for the representation of protein-protein interactions, it can also be used for the representation of pathways.

The PSI-MI format offers a certain trade-off between ease of use and language strength. A structural unit that can be interpreted as a pathway - an *entry* in the format's vocabulary - provides means for the declaration of an interactor respectively an interaction list. Certain attributes, like the interaction type, can be chosen from an external controlled vocabulary. The vocabularies themselves can be decided upon by the user. Moreover, it is possible to use attribute lists for storing information that is not directly supported by the standard's definition.

Some of the databases that currently offer their information in PSI-MI format are: IntAct [63], BIND [34], MINT [37], HPRD [35] and DIP [38].

Figure 2.4 displays a short excerpt from a PSI-MI file.

2.2.4 Summary

In this section, we provided the reader with a quick overview of the most important biological pathway data representation formats. Each of the three standards is too complex to be described properly in only a few sentences, but for the sake of completion, we will attempt to summarize their features in relation to one another.

SBML is best suited for quantitative models, where a high degree of interoperability between different application platforms is required. Although difficult to use due to its high complexity, SBML is presently one of the best supported formats by databases and applications.

BioPAX is the format with the strongest expression level and is best suited for automated reasoning due to its intrinsic semantic definitions. The high complexity and computational overhead make it less attractive for common use.

PSI-MI is, by comparison, the simplest format with the lowest computational overhead. The ease of use, however, comes at the price of lower expression power.

Chapter 3

MOLECULAR NETWORKS

In this chapter we will elaborate on the concept of molecular networks, which is also the main paradigm in our dependency graph. After providing a quick overview of an existing system, we will introduce our own approach to building a human specific protein network.

3.1 Overview

In the first chapter of this thesis we described the differences between the reductionist and the integrative research approaches to systems biology. In the second one we elaborated on the different biological information types. We can now combine this knowledge to describe a new research paradigm: the molecular networks.

The molecular networks are the next logical step away from the reductionist research principle, based on the rationale that all entities (DNA, RNA, proteins, etc.) present in the cell interact, exhibiting an emergent behavior grounded on their coexistence.

Molecular networks can be constructed on the basis of different biological information types. Figure 3.1, for example, depicts a BCL2-associated protein interaction network as generated by the STRING database and web tool (please refer to the next section for details).

The main benefit of representing biological information in the form of molecular networks can be summarized as providing means to facilitate the access to the emergent properties of its members.

3.2 An Existing Approach: STRING

Several approaches to creating molecular networks exist today, with STRING being one the most significant projects to date.

STRING (Search Tool for the Retrieval of Interacting Genes/Proteins) aims to collect, predict and unify most types of protein–protein interactions, including direct and indirect associations [64]. The data is obtained by high-throughput experiments, database and literature mining, and predictions based on genomic context analysis [65].

Aside from direct interaction databases including (but not limited to) DIP, MINT, HPRD and IntAct, STRING also imports curated biological pathway

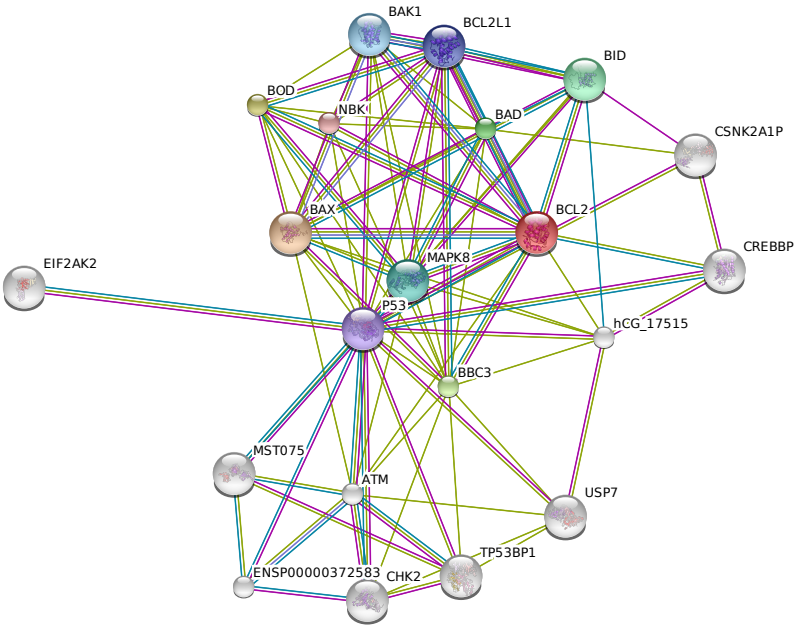


Figure 3.1: An interaction network example for the BCL2 apoptosis regulator as generated by the STRING database and web tool

knowledge from KEGG and Reactome.

Additionally, several algorithms are used on genomic data to predict further protein interactions. The most important ones are listed below [66]:

- Within prokaryotic organisms, the close proximity of genes on the chromosome is a good indicator for functional linkage.
- Genes that join to encode a single fusion protein are considered to share a dependency.
- Genes sharing similar phylogenetic profiles are also considered to contribute to similar functional processes.
- A similar transcriptional response across a variety of conditions is an indicator for functional linkage between genes.

Two further resources are taken into consideration: text mining and interaction analogy in different organisms. The first one includes, among others, the parsing of all abstracts available in the NCBI's PubMed library [67]. The second one implies the transfer of interactions detected between proteins of an organism A to an organism B, if the same proteins are also available in the latter.

All protein pairs are assigned scores describing their information quality in respect to their source, type, experimental verification, etc. Considering that information on a certain association can be obtained from different sources and

that the respective scores may vary, a *combined score* must be calculated. The mathematical formula employed for this purpose is:

$$S = 1 - \prod_i (1 - S_i)$$

with S_i symbolizing the individual scores. The molecular network is defined with the proteins as nodes and their associated interactions as edges. The edge weight function is S.

One disadvantage of the STRING approach is that it only uses one protein per gene, i.e., the one with the longest nucleotide sequence, not taking splice variants into consideration. This also explains why many of the vertices in the network depicted in figure 3.1 are annotated using gene names, e.g., NCBI gene symbols, rather than protein identifiers.

3.3 Building a Human Specific Protein Network

Similarly to STRING we build our molecular network by assigning edge weights based on the information we have consolidated in our database. A major difference to the project described previously, however, is that we construct a complete graph rather than compiling only a partial one.

Also, we do not discard the information on splice variants, but, through the use of hyperstructures, are able to associate each piece of data with the corresponding biological entity.

Last, but not least, our system includes subcellular and extracellular (plasma) location information, as well as knowledge from the PANTHER database.

The first step of the network construction process consists of splitting the information in four category types and calculating for all molecules a pairwise partial dependency score. The generator functions used are listed in the following.

1. f_{INT} - Interaction

   This function consolidates and evaluates all direct and indirect interaction information available in our database for the two proteins given as input parameters. This information includes the databases IntAct, OPHID and BioGrid as well as the interaction lists extracted from the KEGG and the PANTHER pathways.

   The function returns a value of 1 should there be enough evidence for at least one interaction between the two proteins, and 0 otherwise.

2. f_{GEX} - Gene Expression

This function evaluates the correlation of the expression of the two genes given as input parameters across different tissue types. For this purpose, we use the Pearson correlation coefficient, defined as follows:

$$f_{GEX} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{X_i - \mu_X}{\sigma_X}\right)\left(\frac{Y_i - \mu_Y}{\sigma_Y}\right)$$

where X and Y are the two expression vectors, and μ and σ the corresponding mean value and standard deviation.

3. f_{FAT} - Functional Annotation Terms

   The functional annotation value describes the two input gene's dependency as derived from the Gene Ontology Terms, i.e., how many terms the two entities have in common. Furthermore, the function evaluates the number of pathways (KEGG and PANTHER) the two molecules jointly populate.

   For this purpose we calculate the Dice coefficient, defined as follows:

$$f_{FAT} = \frac{2|X \cap Y|}{|X| + |Y|}$$

   where X and Y denote the sets of terms and paths associated with each of the two input parameters.

4. f_{LOC} - Location

   This function evaluates the expression of the two proteins specified as input parameters in the various subcellular components, and in plasma.

   The information taken into consideration originates from the SwissProt database, the Wolf PSORT algorithm and the Human Plasma Proteome Project. Since the first two use different classifications, with UniProt being finer grained, we have decided to map everything onto the PSORT location list. We therefore have a ten-valued consolidated location array for each protein.

   The function is defined as follows:

$$f_{LOC} = 1 - \frac{\sum_{i=1}^{10}|X_i - Y_i|}{\sum_{i=1}^{10}|X_i + Y_i|}$$

where X and Y denote the ten-dimensional location vectors of each protein.

The four introduced functions are united by a fifth one, used to calculate the consolidated edge weight. This latter function is defined as follows:

$$f = f_{INT} + \left(2 \cdot \frac{f_{GEX} + f_{FAT} + f_{LOC}}{3}\right) - 1$$

The partial functions f_{INT}, f_{GEX}, f_{FAT}, and f_{LOC} have the codomain $[0, 1]$ and the return values of the consolidated function therefore lie within $[-1, 2]$, with -1 representing a very low, and 2 indicating a very high probability of a functional dependency between the arguments.

As stated earlier, we calculate the edge weights between all possible pairs of nodes, which leads to a number of edges equal to the square number of proteins represented in the network. Considering that our network, in the current version, holds information on approximately $24 \cdot 10^3$ genes coding for roughly $71 \cdot 10^3$ splice variants, we obtain consolidated edge weights for $5 \cdot 10^9$ edges. Our main approach to tackling this vast amount of generated data is to introduce a *cutoff*, i.e., we discard all edges with a weight below a certain cutoff-value. For more information on the obtained results, please see chapter 5.

3.4 Graph Characterization

Having introduced the concept of molecular networks, we will now elaborate on the associated analysis principles.

A classical approach is to interpret the molecular network as a graph. The direction and edge weight properties are problem specific. Shifting the interpretation of the network into mathematics opens up an entire array of analysis possibilities.

A concept inherent to mathematics that can now be applied to molecular networks are the graph characterization values or graph invariants. A good overview of common graph invariants can be found in [68]. The most important ones, i.e., the ones with the highest relevance in molecular biology, and, specifically, in our project, are listed in the following.

- *Clustering coefficient* - This parameter is defined for a node u and provides information on how strongly connected u's neighborhood is, i.e., how many edges exist between u's neighbors, compared to the maximum number of possible edges between the same vertices. In this context, the neighborhood of u is defined as the set of u-adjacent nodes in the graph.

- *Connectivity* - Similarly to the previous value, the connectivity parameter describes how well connected the entire graph is, i.e., how many edges exist between the vertices of the graph, compared to the total number of possible edges.

- *Index of aggregation (IoA)* - The IoA value plays a special role in the analysis of subgraphs. Considering a case in which we are searching for a well defined set of nodes, the IoA for a certain subgraph provides information on how many of the input nodes have been found in the regarded subgraph.

- *Betweenness* - This parameter is defined for a given input node u and describes the node's importance for the cohesion of the graph, i.e., how many of all existing paths in the graph contain the node u.

- *Closeness centrality* - This value is defined for an input node u and describes how well the given node is linked to all other members of the graph, i.e., the shorter the paths from u to the rest of the nodes are, the higher the closeness centrality value is.

One of the most important properties of cellular networks is that they are *scale free*. While the node degrees of random networks follow a Poisson distribution, the molecular networks are characterized by a power-law degree distribution and are highly non-uniform: most of the vertices have a low degree with only a few hub-nodes.

According to [69], the scale freedom property of biological networks is based on their evolutionary origin. The authors identify two specific reasons. First, biological networks are the result of a *growth process* in which new nodes becomes attached to existing ones. Unless all nodes attach to and form a single strand - which is highly improbable -, local star topologies will occur. Second, new nodes tend to prefer connections to existing ones that already have many links. This process is known as *preferential attachment*.

Further network properties emerge from their scale freedom:

- The *small world effect* is implicitly conditioned by the special degree distribution of a scale free network. Having most vertices connected to hub nodes leads to relatively short paths between any two vertices of the graph.

- Due to the same special degree distribution, such networks have a high *robustness*, since most nodes can be removed with the graph still remaining connected.

- *Modules* can often be identified in biological networks, each with a specific function.

- *Motifs*, i.e., subgraphs - with a certain structure - which occur significantly more often than others in the network, are also a specific trait of biological networks. A typical example is the feed-forward motif, in which the product of gene X regulates both genes Y and Z, while Y also regulates gene Z [70].

In this chapter we have provided the reader with a quick overview of the molecular network concept and have introduced two different specific approaches to generating protein dependency graphs. During the course of the next chapter, we will further describe the construction logic behind our molecular dependency network.

Chapter 4

THE DATABASE

This chapter provides the reader with an in-depth description of our biological information database. We will outline the general concepts employed and describe our specific domain model. Finally, we will briefly describe our database maintenance application.

4.1 Rationale

In the previous chapters we introduced a series of biological information databases. As this type of storage framework is ideal for the housing and retrieval of large data volumes - while upholding consistency constraints [71] -, its use has also become common practice in systems biology.

Our project requires data from different sources, and we chose to tackle this issue by building a data warehouse. While doing so, we unify the structure of the imported data, which, in turn, optimizes the ease of access and eliminates the risks of redundancy and inconsistency.

In contrast to other similar systems, advanced database management features like distributed architecture or optimized access concurrency, do not play key roles in our infrastructure.

Let us consider some of the available options. The ORACLE [72] database, for example, although powerful and - if properly tuned - fast, does not account for its extreme configuration complexity. Also powerful, but much easier to deploy is PostgreSQL [73]. The comparably low speed of the latter, however, could prove to be a major disadvantage in the long run. An overall better choice, in our case, would be the well-known MySQL [74] database.

The Database Management System (DBMS), however, is just one variable in the envisioned system. Furthermore, binding our analysis framework to a certain database would be a poor software design decision, even if our project were not situated in the research field. Since this is very well the case, we have to remain flexible on this end.

To be able to pursue this issue further, we must first say a few words about the programming language our system has been developed in. In order to facilitate our project's deployment in a heterogeneous network, we have decided to use Java, presently at version 1.6, as the base programming language. A pure Java approach was not possible due to the usage of mandatory libraries, for example

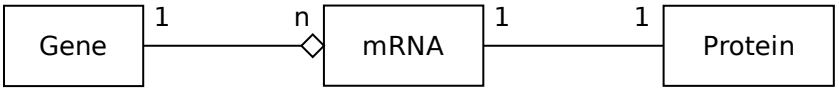


Figure 4.1: A trivial class diagram example

```
class Gene {
  String id;
  List<RNA> mRNAs;
}

class Protein {
  String id;
  RNA mRNA;
}

class RNA {
  String id;
  Gene gene;
  Protein protein;
}
```

Listing 4.1: Java code for classes in figure 4.1

written in C++, like the libSBML for the parsing of the PANTHER pathway files.

Using Java also has other implications on the project's architecture: while all data processed in the program is organized in objects, the classical approach to database information storage is represented by the entity-relationship model. Two options are available at this point: using direct SQL statements and explicitly de-/marshaling the data, or employing an *object-relational (OR) mapping*. We have decided in favor of the latter.

Put in a nutshell, an OR mapping system is able to transparently translate between an object-oriented model, like the one used by the Java programming language, and the standard entity-relationship model employed by most databases.

To better illustrate the concept of object-relational mapping, let us have a closer look at a specific example related to protein biosynthesis. Supposedly we wish to persist information describing which genes are transcribed into which mRNA molecules, and then further, translated into proteins. For now, we will omit the possibility of trans-splicing.

Figure 4.1 formalizes the requirements described previously as a class diagram embodying the following semantics: an mRNA molecule is transcribed from exactly one gene, and is translated to exactly one protein, whereas one gene can be transcribed to a variable number of mRNAs and a protein is always translated from exactly one transcript.

Listing 4.1 contains the Java source code associated with the class diagram in figure 4.1. The reader will notice, however, that the transition between the two representation forms is not bijective. While, for example, the class diagram dic-

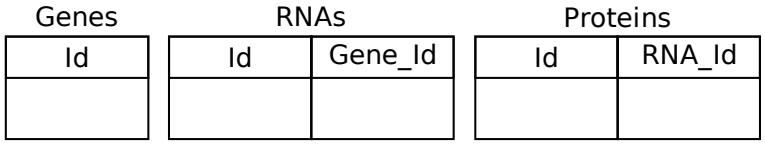


Figure 4.2: Relational table structure for classes in figure 4.1

tates that an mRNA and a protein instance share a bidirectional association, the source code enforces no such constraint. It is, theoretically, possible to instantiate an mRNA object and point it to a protein, without also mapping the relationship backwards.

This is possible because the grammar of the Java programming language is not powerful enough to express logical data dependencies. This is not necessarily a Java flaw, but is rather a characteristic of the imperative programming paradigm and has been explicitly addressed in the declarative programming approach, for example, in the Prolog-derived languages.

Figure 4.2 depicts the database table structure associated with our model. The reader will notice, that this new transformation is also non-bijective. While the relational table structure provides a better formal description of the class diagram than the Java code, it is neither unique, nor does it match the diagram's semantics perfectly. The gene-mRNA one-to-many relationship depicted in the class diagram is modeled in the database by saving the transcribed gene's id with the corresponding mRNA. This allows only one gene per transcript, without limiting the number of mRNAs per gene. Less optimal, however is the description of the mRNA-protein one-to-one relationship: by saving the RNA's id together with the protein, we implicitly allow only one RNA per protein, but it is - erroneously - possible to associate more proteins to the same transcript. We will regulate this by defining SQL constraints.

The transition from Java to SQL is done by the object-relational mapping system, that is also responsible for generating the database structure altogether. The process can be controlled by using *annotations*, i.e., certain syntactic constructs that instruct the OR system how to handle the transformation, e.g., which tables to create, how to manage associations, etc. We will describe some of the most important annotation types in the next section.

Please note that the previous example is trivial. The procedure becomes more complicated when attempting to reconstruct complex relationships like inheritance, many-to-many associations, etc.

The main advantage of such mapping systems is that they eliminate the need for the user, i.e., in this context the programmer, to work with the back-end databases and write direct queries for the underlying tables.

Let us consider a specific use case in which a list of transcripts corresponding

to a set of given criteria is required. The user can post the query to the OR mapping system, and the result will be a list of RNA-class instances upholding the specified constraints. To obtain this list, the mapping system automatically translates the criteria into SQL, executes the query, and finally, marshals the results obtained from the database into objects.

By hiding the database, it also becomes possible to exchange it altogether if the requirements to the underlying DBMS change, without having to modify the application code. This characteristic of the OR mapping systems obsoletes the problem of a definitive DBMS binding and provides us with exactly the flexibility we mentioned earlier in this chapter.

The following section provides the reader with a quick overview of the specific object-relational mapping system we employ.

4.2 Hibernate

Hibernate [75] is presently a state-of-the-art OR mapping system for Java. Aside from implementing Sun's Java Persistence API [76] (JPA), it provides extensive mechanisms for session and entity management, as well as object querying with full-text search.

Considering the framework's complexity, the Hibernate workflow is relatively straight-forward. An overview is listed in the following.

1. The first step consists of defining the domain model, e.g., by drawing the class diagrams.

2. During the second step, the domain model is implemented in Java.

3. The Java code is then annotated with either JPA (our choice) or XML. The annotations instruct Hibernate how to perform the mapping, e.g., which classes should be persisted, which constraints should be applied, etc.

4. During the fourth step Hibernate can be instructed to extract the relational model from the class definitions and automatically create the database.

5. Once the database exists, Hibernate can be used to create, query, modify and delete instances of the annotated classes. The object state will be persisted automatically.

In the course of the next sections we will describe the domain model employed in our project, underlining both the considerations based on requirements imposed by the biological background of the stored data and the ones fueled by the need to avoid traps and pitfalls typical of object-relational mapping technologies.

Before proceeding, however, we must first provide the reader with a quick introduction to the mapping principles.

4.2.1 Basic Techniques

A *mapping* is, in this context, a logical correspondence between a structure in the object-oriented paradigm and a structure in the entity-relationship paradigm, on the basis of which a bijective transformation between the two states can be defined.

The simplest association possible is that of a class to a database table. Assuming that all properties of the class are basic Java types, like `char`, `long`, `int`, etc., (`String` is, by exception, also considered a basic type), it is possible to create a database table for the regarded class, in which each column holds one class property. Each instance of the class created in the program and persisted by Hibernate will be saved as a table entry. The inverse transformation reads a row from the same table, and marshals all values back into the corresponding class variables, recreating the original object.

Since Java classes contain properties of not only basic types, but also references to other instances, Hibernate provides mechanisms to address this issue as well. The simplest reference type is a single pointer to another object, like `protein` in the RNA class, as depicted in listing 4.1, previously in this chapter. This relation is implemented in the database by using a foreign key column in the table associated with the class containing the reference. This pointer $\leftrightarrow$ foreign key pattern is the foundation of more complex techniques, like associations with different cardinality. We will present such techniques in the next section.

4.2.2 Advanced Techniques

The more advanced Hibernate techniques are related to the following topics:

- *Collections*

  A common concept associated with the object-oriented paradigm is that of collections, aggregating entities on the basis of common function or structure. When transposing collections into the relational model, their original semantics play an important role in the design of the database tables. We distinguish between:

  - *One-to-many*: This type of relationship describes an asymmetric association, in which the entity of the *one* side can be linked to an arbitrary number of elements of the *many* side, while the opposite is not true. A typical example is the gene-RNA association in figure 4.1.

    This type of relationship is implemented in the database by saving the primary key of the *one* side together with each element of the *many* side, thereby implicitly ensuring data consistency.

A major Hibernate feature is its ability to reconstruct bidirectional associations by using data from only one side, e.g., in our gene-RNA example, Hibernate will automatically reconstruct the RNA list in a gene object by inspecting only the foreign (gene) keys used in the RNA table. By employing this technique, the OR mapping system automatically prevents inconsistencies.

- *Many-to-many*: Contrary to the former, the latter type imposes no constraints on the association cardinality. However, like in the previous case, the relationships must be bidirectional.

  Let us extend our gene-RNA example with the possibility of trans-splicing, i.e., the primary transcripts of several genes are combined to form the final mRNA. In this case, the mRNA is not only associated with one gene, but, to several, forming a logical many(genes)-to-many(RNAs) relationship.

  In the database, Hibernate uses a separate join-table to store this type of relationship. This new table consists usually of two columns, one for each primary key of the two *many* sides, and holds an ordered Cartesian product of all the keys involved in the association.

  When restoring persisted objects that contain many-to-many associations, Hibernate walks the join-table and rebuilds each collection from the entries found therein.

- *Inheritance*

  Another concept inherent to the object-oriented paradigm is class inheritance. Hibernate provides means for transparent inheritance handling, i.e., genericity. Per default, when querying persisted objects with an OR mapping system, the user must specify the exact targeted class. In case of inheritance, however, it is possible to query an abstract super-class, and receive specific sub-class instances that can be subsequently casted to their correct subtypes.

  Several options for inheritance mapping exist [75]. The most important ones, each with its own advantages and drawbacks, are listed in the following.

  - *Table per concrete class with unions*: When using this technique Hibernate creates tables only for the non-abstract members of the inheritance hierarchy. Each table holds all properties of the implemented class, i.e., tables associated with classes sharing a common ancestor in the inheritance tree have similar columns.

This characteristic and the need to perform unions each time a superclass is queried are the two main drawbacks of this strategy. By eliminating the need for joins, however, this type of mapping performs very well in case of broad, but otherwise shallow inheritance hierarchies with many class instances.

- *Table per hierarchy*: This technique creates a single table for all classes of the inheritance tree. Fields that do not exist in certain instances remain `NULL`. Hibernate uses a discriminator field to decide which row contains which class type.

  The high data overhead is the major disadvantage of this strategy, while the performance is maximized since no joins or unions are necessary.

- *Table per subclass*: This is a normalized database form in which all inheritance relationships are represented as foreign key associations. Each type (regardless if abstract or not) is persisted in its own table, with each field in the inheritance hierarchy occurring only once in its source entity. By splitting a class instance across several normalized tables, this mechanism reduces the risk of data inconsistency.

  When reconstructing an object, Hibernate walks the inheritance hierarchy top-down, inspecting each corresponding table and adding new data to the rebuilt object as necessary.

  This strategy eliminates the need for unions, but dramatically increases the computational overhead generated by joins.

- *Performance*

  The overall performance is a central issue when working with Hibernate. Although powerful, the framework can easily stall an application if used improperly. Best results are often achieved by accepting certain trade-offs. In some cases, for example, adopting a lower normal form [77], like embedding *child* entries in the *parent* table, can provide a powerful speed-boost at the cost of a higher data inconsistency risk.

  Another mechanism for safeguarding information consistency is provided by the cascading semantics mechanism. Cascading semantics are used, for example, in strong aggregations, i.e., where a set of child elements can only exist as long as their parent entity does. If the latter is removed, the former must be deleted as well. Although optimal for managing dependencies, the cascading semantics can significantly degrade performance when running redundant checks on large data volumes.

  In Java applications objects usually build connected networks. If Hibernate were to follow all pointers present in an object, i.e., all foreign keys in a table

row, a query for a single object would result in the OR mapping system trying to load the entire database at once. This issue is addressed by a mechanism called *lazy loading*. Hibernate does not load objects completely, but instead retrieves only the mandatory sections, and provides proxies for the rest. Should the user attempt to access a proxy, Hibernate would search the database and provide the information on demand.

Basic types and, generally, one-to-one associations are loaded from the beginning (*eager loading*). In the domain model implemented in listing 4.1, the protein associated with an RNA instance would, for example, be part of this category, unless explicitly excluded by the programmer.

Collections, on the other hand, regardless of their cardinality, are subject to lazy loading. Again, tuning the loading policy, may improve or degrade performance.

Having seen how one-to-many associations are implemented in the database and having introduced the lazy loading mechanism, we can now describe a special technique for increasing Hibernate performance when inserting a large amount of data.

Since one-to-many collections are built from the single foreign keys in the *many* side objects, the optimal way to save such associations is to set only the *many* side pointer. The *one* side collection does not have to be modified. Both changes are formally equal, but since collections are loaded lazily, we can skip their associated queries in this case altogether. In our specific gene-RNA example, it would be sufficient to set the gene pointer of the RNA object. When loading the associated gene, its mRNA list will be updated automatically.

This principle applies only to one-to-many relationships. Many-to-many associations, however, not only occur less often than the former, but can also be implemented by using two symmetrical one-to-many relationships.

The concepts and techniques presented in this section play key roles in the design our application. In the course of the next sections we will apply this knowledge to create our domain model, while bearing in mind that decisions related to the class hierarchies have a high impact on the corresponding database entities.

4.3 Data Model

Our domain model is divided in four distinct sections:

- The *core* holds the identifier mapping.

- The *context* describes the information we use to generate the dependency graph.
- The *content* describes additional biological information that is not used directly during the graph generation.
- The *biomart* holds the consolidated information extracted from the context section as well as the reference graph itself.

We have decided in favor of this classification for several reasons. The core part is stand-alone and can be used by itself as an identifier mapping infrastructure. The context section is a raw data repository used only during the computation of the biomart. The content information is an optional raw data repository that has no influence on the generation of the dependency graph. The biomart provides means for consolidating information from the other sections and is intended to store different versions of the dependency graph.

4.3.1 Core Identities

The domain model for the core section of the database is depicted as class diagram in figure 4.3.

In section 2.1.1 we elaborated on the identity mapping issue and we introduced the biological hyperstructure concept, uniting all identifiers associated with a specific biological entity.

The various names and identifiers are described in an inheritance tree rooted in a generic biological identity structure. The latter stores information on the name and description of the molecule. If the name is deprecated, the structure additionally points to the presently valid identity. Due to the inheritance mechanism, all attributes are passed down to all subclasses.

The generic biological identities are split further into genes and proteins. This mechanism allows us to transparently add new entity types, like mRNA, etc.

We have three types of gene identifiers: symbols and identifiers from NCBI as well as identifiers from ENSEMBL. A distinctive feature of the symbols is that each one keeps additional track of its own set of alias names.

All proteins can have an associated sequence, and the main group is split into three subcategories based on their origin: IPI, NCBI and UniProt. The NCBI is further divided into manually curated (NP) and computationally predicted (XP) identities, whereas UniProt is separated into SwissProt and TrEMBL.

This architecture allows a flexible handling of the mapping and provides the user with the ability to either treat all entities equally, by using the super-class, or specifically, by down-casting them to their corresponding types.

Due to the large number of identifiers available in the database we decided to map the inheritance tree using the *table per concrete class with unions* technique,

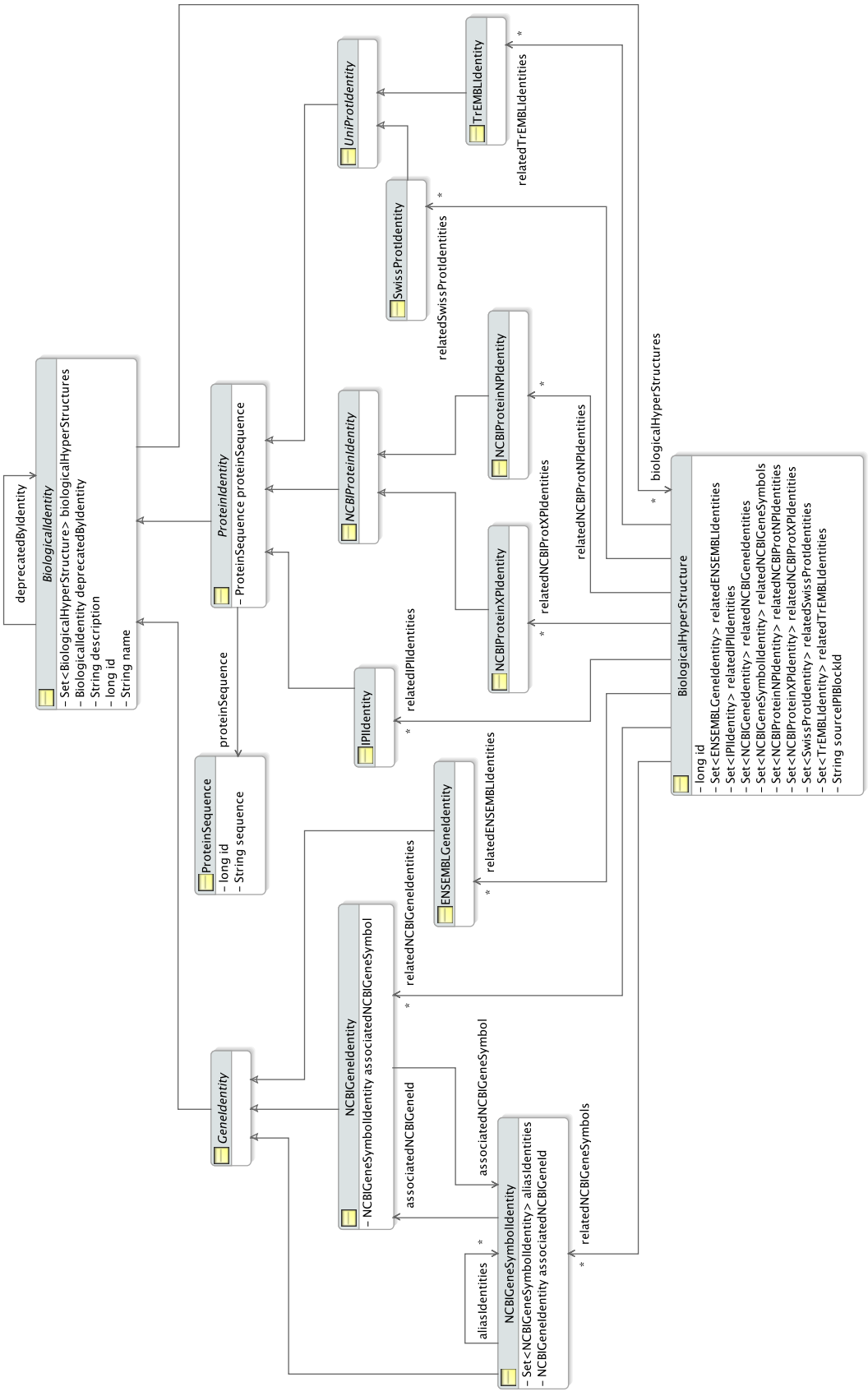


Figure 4.3: The core identities

which allows us to increase the query performance if we use specific classes instead of generic ones, e.g., the IPI identity is used instead of the generic biological identity.

4.3.2 Context Information

All biological information types introduced in sections 2.1.2 to 2.1.7 are considered to be part of the context data group. We will now provide the reader with a description of the corresponding domain models.

Protein Interactions

The domain model for the interaction information is depicted in figure 4.4.

From a strictly relational point of view, an interaction is basically a two-dimensional vector of biological identities. We differentiate between imported and extracted interactions. The former are obtained from OPHID, BioGrid and IntAct, whereas the latter are generated from the KEGG and the PANTHER pathways.

While the extracted interactions are annotated directly with their type and subtype, the imported ones keep track of their experimental confirmations, like experiment type, confirmation count, etc.

The experiments themselves are divided in concordance to their origin into the IntAct and BioGrid categories. We must save both, since a one-on-one mapping is not possible. By using a restricted vocabulary, the IntAct experiment list is considerably better suited for automated processing. It also allows an annotation of the interactions with direction and directness attributes.

Functional Pathways

The domain model associated with the functional pathways is depicted in figure 4.5.

As defined in our class model, a pathway consists of complexes and reactions, which are defined in a common inheritance tree rooted in the abstract pathway component structure.

A complex can be either basic or extended, with the difference that an extended one can be built of an entire group of basic complexes. A basic one, on the other hand, is either a homo-polymerization of a single biological identity or a family composed of several distinct identities. Again, all subtypes are either KEGG or PANTHER specific.

A reaction is annotated with its type (e.g., binding, compound, etc.), subtype (e.g., glycosylation, phosphorylation, etc.), component interaction type (e.g.,

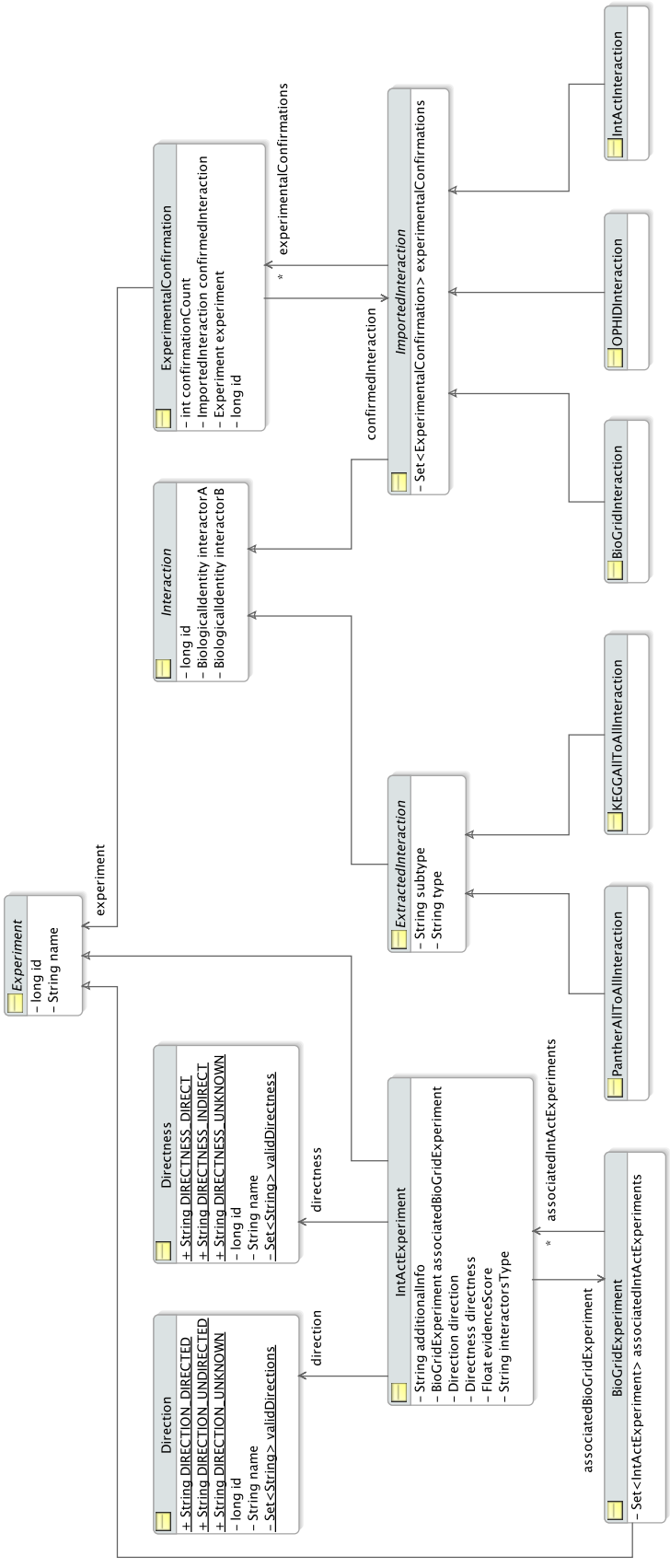


Figure 4.4: The interaction information

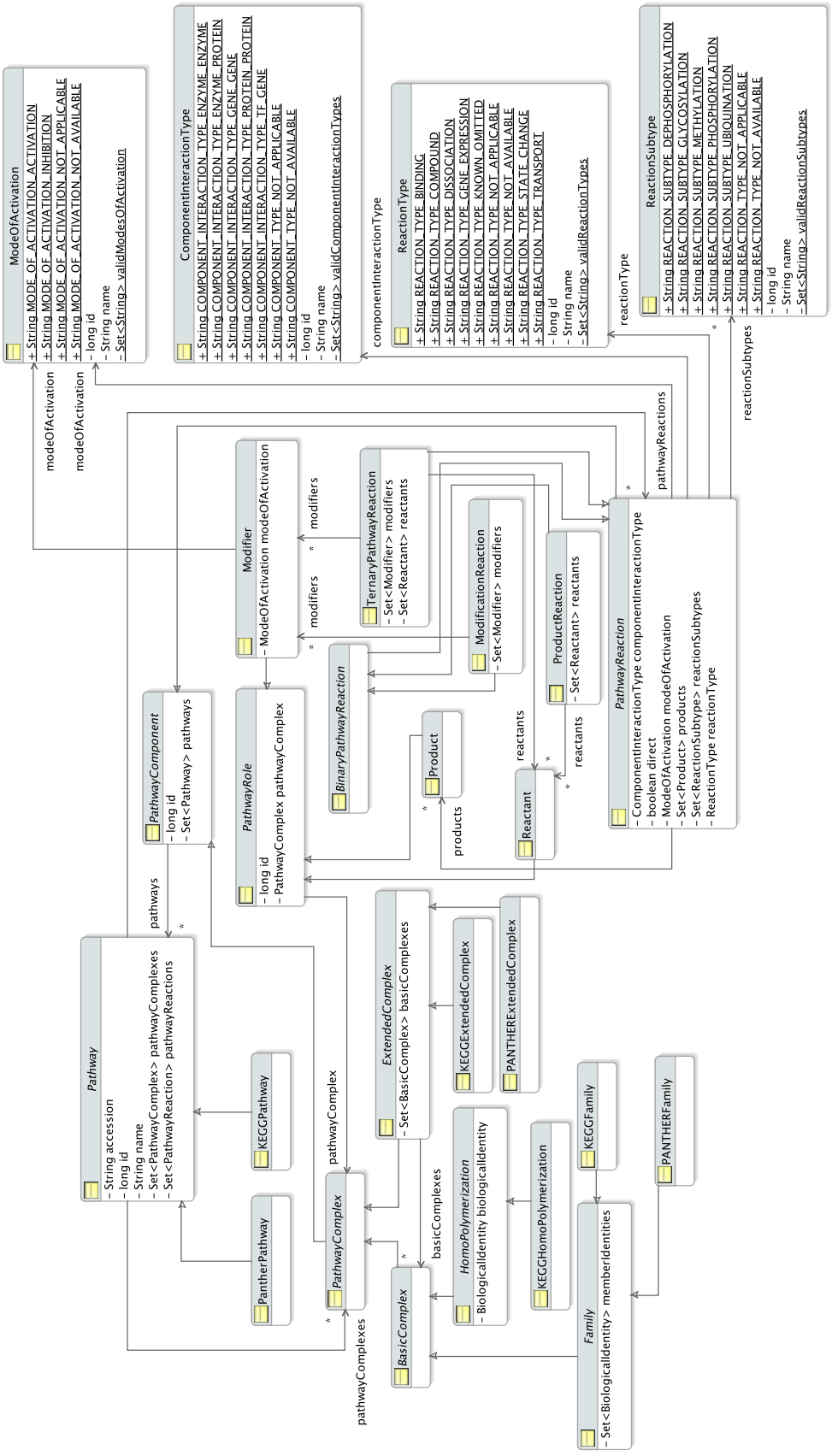


Figure 4.5: The pathway information

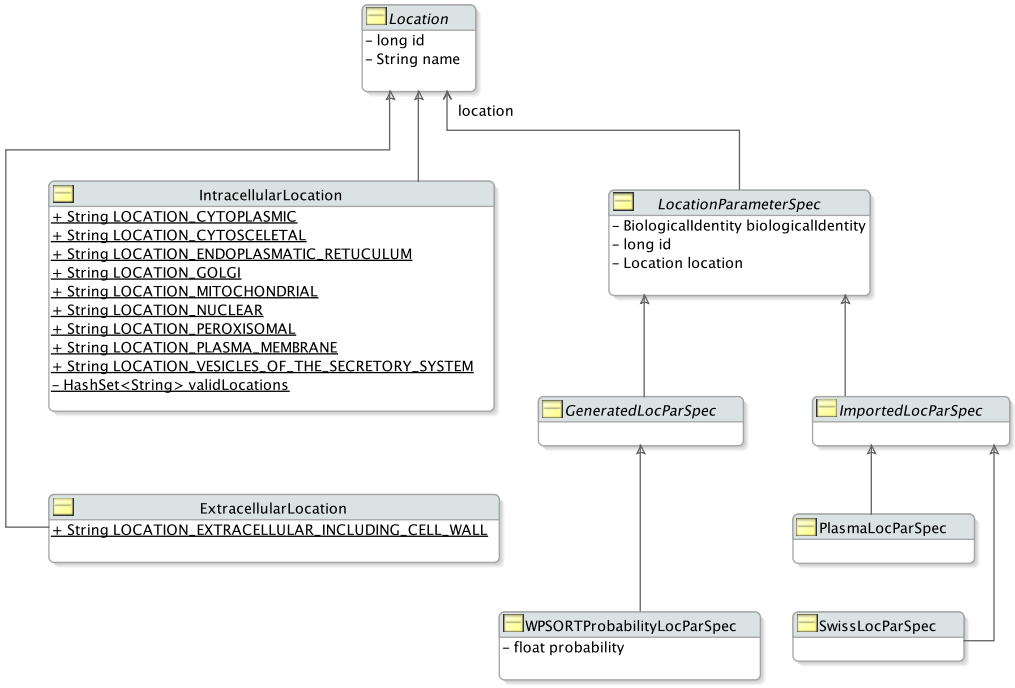


Figure 4.6: The location information

gene-gene, protein-protein, etc.) and mode of activation (e.g., activation, inhibition, etc.).

Complexes can play various roles in reactions: products, reactants and/or modifiers. Each reaction has at least one set of resulting products.

Depending on the number of interactors, we distinguish between binary and ternary reactions. Additionally to the inherited products list, the binary reaction type can also hold a list of reactants or modifiers, depending on its classification in its original database, while the ternary reaction has all three interactor types.

Subcellular Location

Figure 4.6 depicts the domain model associated with the subcellular location information.

We classify the location information for a biological identity in accordance with the data source and the associated subcellular compartment.

The data is either generated or imported. We generate a location probability vector by running the Wolf PSORT algorithm on the protein sequence of the biological identity. At the same time, we import location information from the SwissProt database and the Human Plasma Proteome Project.

The specific locations are separated into the intra- and extracellular categories.

Gene Expression Sheets

The domain model associated with the gene expression information is depicted in figure 4.7.

We group all expression information in datasets. Each dataset holds information obtained from a series of samples, whereas each sample originates from a certain tissue (e.g., bone marrow, retina, etc.), in a certain developmental stage (e.g., fetal).

Each biological identity has an associated expression value for a given sample in a given dataset.

Biological Ontologies

Figure 4.8 depicts the domain model associated with the biological ontologies. In contrast, for example, to the pathways, a simplified model suffices here.

An ontology is defined in our context by its name, top terms and complete term list. Moreover, we differentiate between imported and generated ontologies. The former include raw data, like an excerpt from the Gene Ontology, whereas the latter include preprocessed information resulting from various computations.

Each term has a name, a list of associated biological identities, and it keeps track of both its parent and child terms.

Transcription Factors

The domain model for the transcription factor information is depicted in figure 4.9. Again, a simplified structure suffices.

The information is predicted pairwise for each biological identity-transcription factor association. Since several binding sites may exist per identity, we only save information related to the site with the maximum score. Furthermore, due to binding domain similarities in the transcription factors, it may be necessary to save more than one factor per identity and maximum score site.

Together with the binding site scoring information, we also save a brief description of the prediction method, like the number of upstream kilo-bases taken into consideration.

4.3.3 Biomart

We argued previously that the context section of the database is basically a raw data repository. Our goal is to create a molecular dependency network based on this information. Considering the large number of actors (presently approximately $7 \cdot 10^4$ proteins) and the quadratic amount of edges that need to be computed (approximately $5 \cdot 10^9$) it only stands to reason that we need to preprocess this

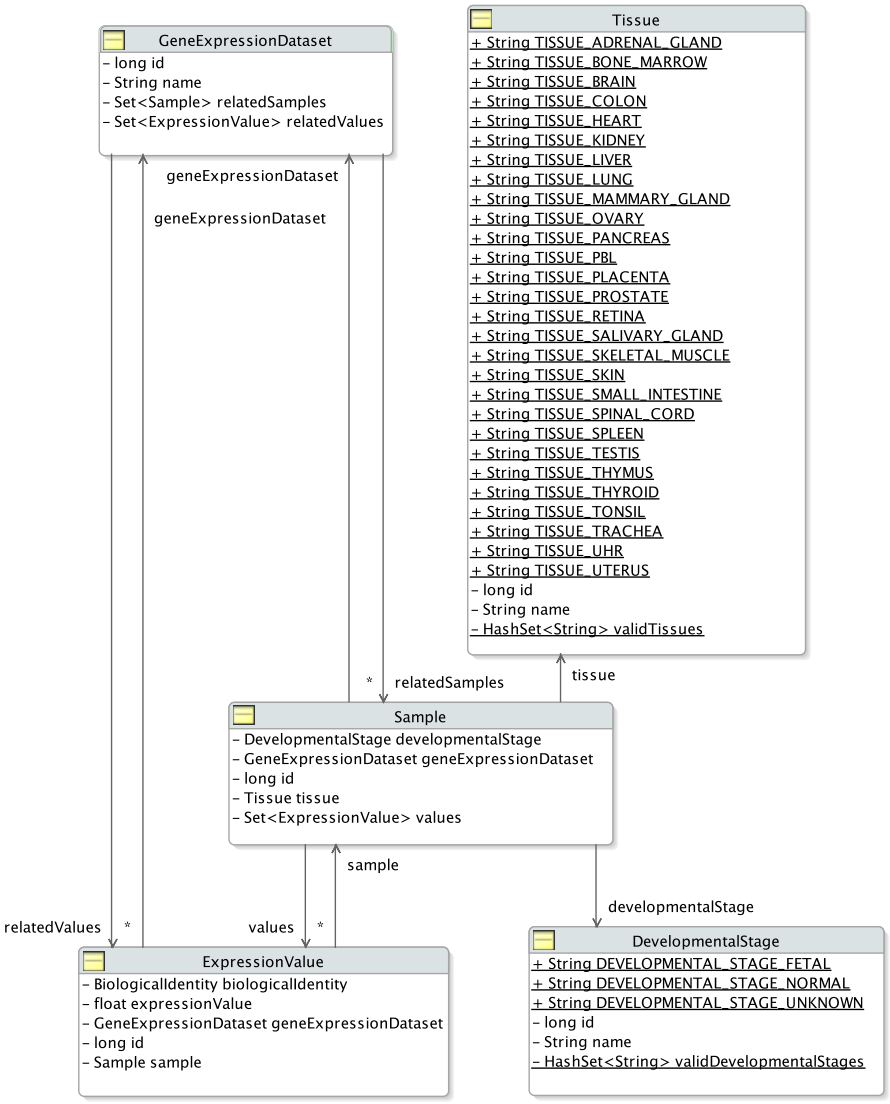


Figure 4.7: The gene expression information

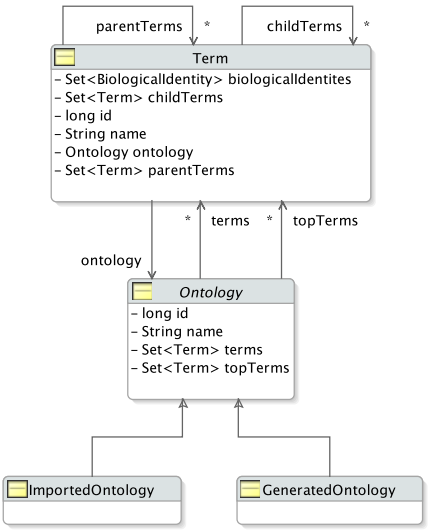


Figure 4.8: The ontology information

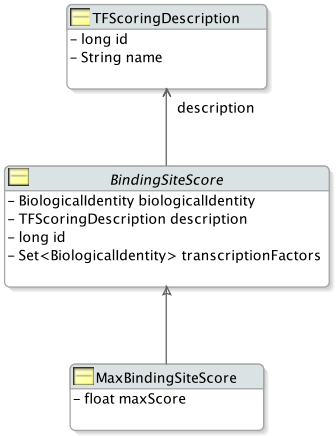


Figure 4.9: The transcription factor information

information and store it adequately in order to increase the query speed when generating the graph. The biomart was designed bearing exactly these considerations in mind.

Consolidated Information

This database section holds the preprocessed information as an early step in the generation of the dependency network. Figure 4.10 depicts the association domain model.

As opposed to the context part, the information is no longer identity-centered but hyperstructure-centered. This way, we unite both the protein and the gene information for each specific biological identity.

For each hyperstructure we save five types of information: localization, interactions, gene expression, transcription factors and functional annotation terms. While the first four consist of rearranged context information with regard to computational performance, the fifth one is algorithmically generated from the biological ontologies and the available pathway information.

We manage the consolidated objects in an inheritance tree in order to preserve information consistency. Moreover, this provides us with the necessary means to introduce additional information later without having to modify the existing structures.

Dependency Graph

Figure 4.11 depicts the domain model for our dependency graph.

Considering the large number of edges that need to be stored, we kept this section of the object-oriented, and, by extension, the entity-relational model simple, to avoid unnecessary computational overhead.

The graph class bears the timestamp and the description of the generation, as well as a list of available nodes.

Each node describes the biological hyperstructure it represents and the edges it is either the source or the target of, whereas each edge provides information on the parameters used for computing its weight.

Hibernate's ability to rebuild one-to-many associations from the pointers of the many-side is especially useful in this particular case. We have here two one(node)-to-many(edges) relationships, which we can persist into the database by setting only the source and target pointers of the edge structures. This way we can execute millions of bulk insert statements into the edge table without having to query any other section of the database.

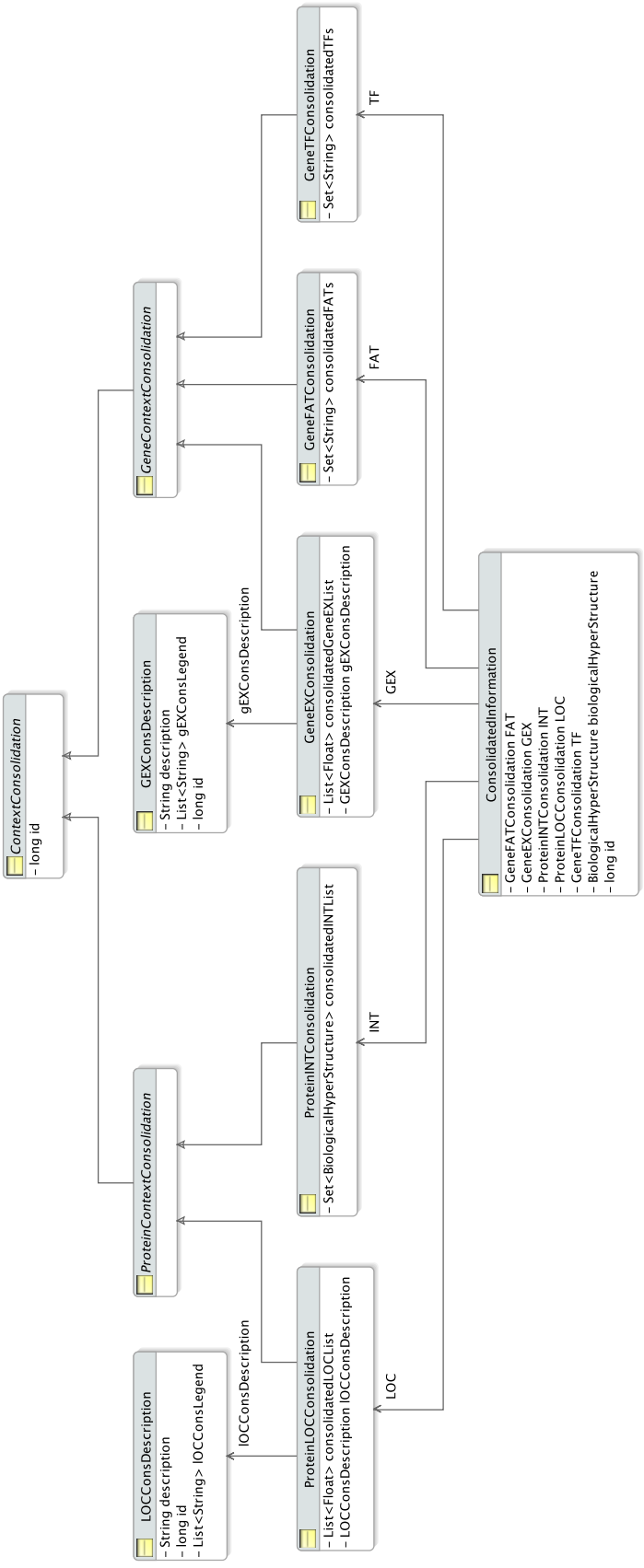


Figure 4.10: The consolidated information

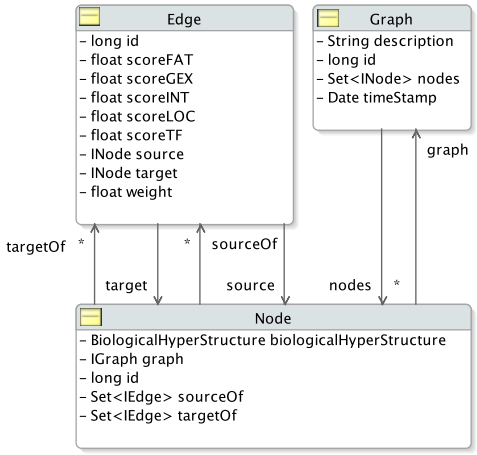


Figure 4.11: The dependency graph

4.4 Database Maintenance

Due both to the considerable size of the final data warehouse and the requirement to continuously update various sections of the database during the research process, we have decided to implement a software tool specifically for this purpose.

Apport is a modular utility able to independently execute different tasks on the database. Figure 4.12 depicts a screenshot of the current application front-end.

Each module to be registered with Apport must expose a clearly defined interface. Dependencies between modules are specified using execution policies, e.g., the data import modules are registered with a priority-based policy, defining the IPI data set, i.e., the core, both as being the first to be run and also playing a critical role for all other imports.

Presently six types of modules are in use respectively planned:

- *Downloader* modules are responsible for data retrieval from the Internet.
- *Importer* modules build the raw data repositories: core, context and content.
- *Preprocessor* modules generate additional information, by, for example, calculating subcellular location probabilities of proteins using Wolf PSORT.
- Biomart *generator* modules consolidate the raw data.
- Graph *calculator* modules compute the dependency network.
- *Extractor* modules pull data from the database.
- *Analyzer* modules perform various computations on the dependency graph, like maximum value paths and spanning trees analysis, etc.

Figure 4.12: A screenshot from the database maintenance program

During the course of this chapter we presented our biological information database, both from a technological point of view as well as from an organizational perspective.

In the next chapter we will provide the reader with specific application examples of our research approach and discuss the obtained results.

Chapter 5

DATA ANALYSIS IN BIOLOGICAL CONTEXT

In the course of this chapter we will outline the rationale for our three biological neighborhood selection methods. We will then apply them on two specific case studies, and discuss the obtained results.

5.1 Biological Neighborhood Selection: Rationale

A standard approach to studying diseases on the molecular level is to determine the genes exhibiting a significant differential regulation in the affected tissue. Hypothesis generation is then conducted by means of literature research.

As present-day knowledge of the cell functioning is limited, this scientific approach often generates false positives. Furthermore, differential gene expression (DGE) experiments provide biased information on the disease itself, by describing a discrete status quo rather than a time-dependent process, which may lead to the erroneous detection of disease effects rather than cause. Furthermore, extrinsic factors like experiment noise may also negatively influence the observation and introduce false results.

In the previous chapters we described a procedure to build a molecular dependency network based on publicly available biological information. We will regard this graph as the *reference* description of a healthy cell.

An improved approach to hypothesis generation, is to highlight the significantly differentially expressed genes on the reference graph, thereby implicitly obtaining information on the biological context.

Before proceeding, we must first make a note on the reference graph used in this chapter. The dependency network described so far in the course of this thesis is a novel, hyperstructure-centered implementation, with its granularity dictated by the protein sequence. As the analysis framework for this graph is not yet completely implemented, we will use the original gene-centered implementation for the analysis performed in this chapter. Both networks are constructed identically, with the only difference consisting in their level of granularity. The gene-centered reference graph has roughly $18 \cdot 10^3$ nodes and approximately $16 \cdot 10^7$ edges.

Since we are working with a complete graph, highlighting nodes without any additional node/edge selection criteria would make little sense. A trivial approach to this issue is to use a *cutoff*. While doing so, we use a fixed edge weight - in the

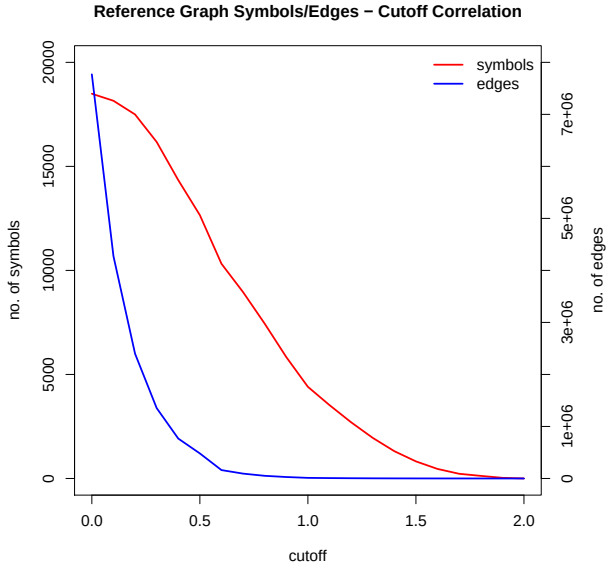


Figure 5.1: Correlation between the number of symbols respectively the number of edges in the reference graph and the cutoff values

interval $[-1, 2]$ - to select a subgraph, all edges of which have a weight above the specified value. All edges below this number are dropped. Additionally, all nodes that have a degree of zero, as a result of the previous measure, are removed as well. The resulting subgraph is used for further analysis.

Choosing the right cutoff is of critical importance. While a high cutoff describes strong dependencies, either obtained from all data sources or demonstrated in experimental setups with high confidence, it tends to bias the information at the same time by emphasizing entities that have often been investigated and therefore a lot of data on the subject is available.

Similarly, using a low cutoff has its own drawbacks. While a low threshold may provide access to correct information obtained through the combination of only some of the different input data types, the exponential increase of the subgraph size threatens to dilute the very same information we are targeting.

To better emphasize this issue, we have provided figure 5.1. Its curves depict the number of symbols (red) and the number of edges (blue) in respect to the cutoff value.

We have only covered the $[0, 2]$ weight interval due to the exponential growth of the number of edges towards lower cutoff values. At a threshold of zero, the subgraph holds approximately $8 \cdot 10^6$ edges, which already prevents almost any statistically relevant statement regarding the underlying data. Also, at the same threshold we have already reached every node in the original graph.

Once we have decided upon the analyzed cutoff, we can visualize the subgraph,

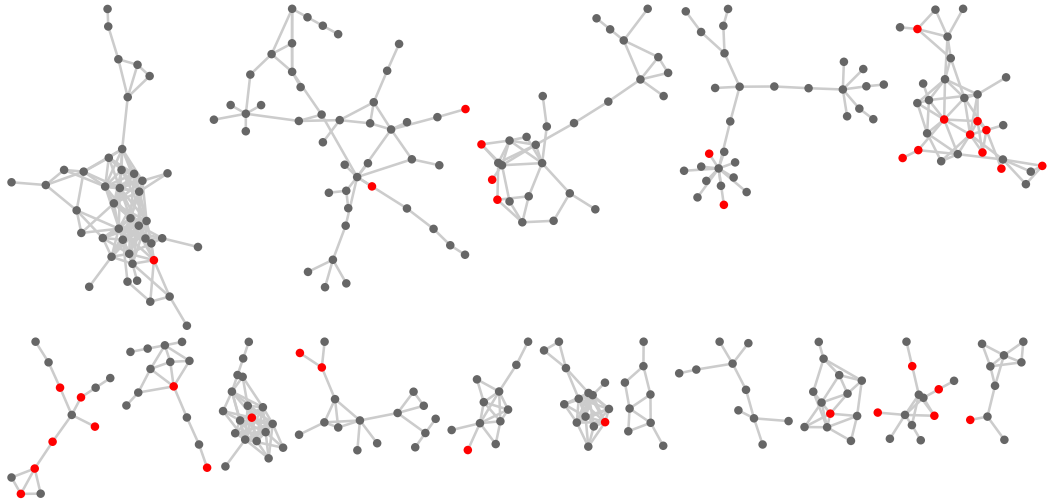


Figure 5.2: A section of the reference graph at a 1.4 cutoff with members of the consolidated BCL dataset highlighted (red)

e.g., with the Cytoscape [78] software. Figure 5.2 depicts a section of the reference graph at a 1.4 cutoff, drawn with the aforementioned program. The red nodes are members of the consolidated B-cell lymphoma dataset, i.e., a list of genes that have been determined as significantly differentially regulated in cancerous B-cells. We will elaborate on this issue in section 5.3.

This approach works, however, only for high cutoffs. For lower cutoffs, additional node and edge selection methods are needed. Since we are obviously interested only in nodes and edges that share some type of relation to the initial entity set, we will refer to them as the latter's *biological neighborhood*.

When provided with an experimentally determined DGE list, we naturally assume that certain functional dependencies exist between its members. The goal of the reference graph is to improve the hypothesis generation regarding such dependencies.

Since strong dependencies translate into high edge weights in our network, it only stands to reason that we mine large graphs, i.e., at lower cutoffs, using maximum value paths (MVP). This way we attempt to, at least partially, reconstruct dependency sub-networks as they exist in the real world. Figure 5.3 depicts a possible outcome of a maximum value path approach to the neighborhood selection for the same consolidated B-cell lymphoma dataset, in a 1.3 cutoff. Contrary to the cutoff-only approach, the MVP-computed subgraphs are much smaller in size since many of the paths overlap.

The concept of *maximum value* paths supports a bijective transformation into *shortest* paths (SP). The latter has been extensively addressed by computer science, especially as a routing problem. We will also employ SP-specific algorithms

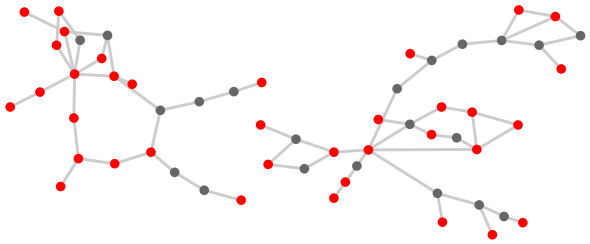


Figure 5.3: MVP-computed subgraph for the consolidated BCL dataset in a 1.3 cutoff

to calculate the maximum value paths.

Contrary to computer science, however, in systems biology we need a slight shift of perspective. Routing problems inherently require only one shortest path between two points A and B. The dependency graph is calculated with a specific mathematical function on the basis of different data sources, which may result in several high-value paths between the same genes. An MVP-based analysis approach would be incomplete, if we ignored these.

Calculating all-to-all maximum value paths, as we do in the previous approach, implicitly infers the existence of dependencies between all involved pairs of genes. This is not necessarily the case since cellular processes may be disjoint with regard to the sets of involved actors. Calculating all-to-all MVPs automatically induces a certain error in the computation by *poisoning* the resulting dataset with non-members.

We described our approach to this issue in the introductory chapter of this thesis: heuristic spanning trees (HST). We consider each node of the input DGE set as being the root of its own spanning tree and add edges (and nodes) beginning from the edge with the highest weight to the lowest. This concept is similar to the computer science's maximum spanning trees (MST), except that the additional constraints we use during their computation prevent them from evolving into MSTs.

The notion of neighborhood inherently implies locality. By not restricting the number of nodes we take during the HST computation, the algorithm would always return the complete graph, and by not constraining the HSTs' depth, we would always obtain the same strongly connected components we see when using the cutoff-only filtering method.

In some configurations, trees may grow together forming cycles. The reasons we refer to them as HSTs are because spanning trees are the computer science concept closest to the graphs we are computing and because we employ modified versions of ST-specific algorithms for their calculation.

In the next section we will have a look at the algorithms employed for the

calculation of the biological neighborhood from a computer science point of view.

5.2 Neighborhood Selection Algorithms

In the previous section we presented the rationale behind the neighborhood selection methods. We will now describe the algorithms themselves, and the associated considerations like performance issues, strengths and weaknesses.

5.2.1 Single Maximum Value Paths

As previously mentioned, we convert the computation of the maximum value path between two arbitrary vertices of the graph to a single shortest path problem. By doing so, we are able to employ SP-specific methods. Moreover, by converting the input data we can avoid having to adapt the algorithms themselves.

To execute the conversion, we must inverse the $[-1, 2]$ edge weight interval. Since many SP algorithms cannot handle negative edge weights, the procedure allows us to also circumvent this issue, by remapping the original value range into $\mathbb{R}_+$.

Let $G(V, E)$ be our undirected and weighted dependency graph, with V its set of vertices and E its set of edges. Let i and j be two arbitrary vertices in V. Let e_{ij} be the edge in E between i and j. Let $\omega(e_{ij})$ be the weight of the specified edge.

We achieve the weight conversion by using the value $2 - \omega(e_{ij})$ instead of the original edge weight. The resulting interval is $[0, 3]$.

For the calculation of the single MVP we considered two well-known algorithms. We will outline them shortly in the course of the next sections.

The Floyd-Warshall Algorithm

The Floyd-Warshall[1] algorithm [21] is a dynamic programming approach to the all-to-all shortest paths problem.

The algorithm has an $O(|V|^3)$ complexity, independent from the number of edges. Although other approaches to the same problem exist, they are dependent on the cardinality of $|E|$, leading in a worst case scenario, with a dense graph, to the same complexity as the Floyd-Warshall algorithm. The latter, however, due to its extreme simplicity bearing almost no overhead, performs better on dense graphs than other approaches with a lower average complexity.

The main disadvantages of the algorithm are its performance on sparse graphs, i.e., the same one as in dense graphs, and its inability to reconstruct the paths once the computation has finished.

[1] also known as the Roy-Floyd algorithm

The Roy-Floyd algorithm is described in the pseudocode listing 5.1.

Require: Graph $G(V, E)$
Require: Edge weights $\omega(e_{ij})$
for all $i \in V$ **do**
  for all $j \in V$ **do**
    $dist(i, j) \leftarrow \infty$ or $\omega(e_{ij})$ if $\exists\ e_{ij} \in E$
  end for
end for
for all $k \in V$ **do**
  for all $i \in V,\ i \neq k$ **do**
    for all $j \in V,\ j \neq k \wedge j \neq i$ **do**
      $dist(i, j) \leftarrow min(dist(i, j), dist(i, k) + dist(k, j))$
    end for
  end for
end for
Ensure: Shortest path weight matrix *dist*

Algorithm 5.1: The Floyd-Warshall algorithm

The method is trivial: it repeatedly tries to interpose a vertex k between all pairs of nodes (i, j) and compute an alternate, shorter route between the two vertices over k.

Dijkstra's Algorithm

Dijkstra's Algorithm [20] is one of the fastest approaches to the single-source shortest paths problem. Its complexity is $O(|E|+|V|^2)$ without any optimizations, respectively $O(|E| + |V| \log |V|)$ when using a Fibonacci heap [79] as priority queue.

Compared to Floyd-Warshall, when computing the *all-to-all* shortest paths, Dijkstra's algorithm has a running time of $O(|V| \cdot |E| + |V|^2 \log |V|)$, which means that for sparse graphs, i.e., for graphs where $|E| \sim \alpha \cdot |V|$ for an arbitrary constant α holds, this approach has a complexity of $O(|V|^2 \log |V|)$, as opposed to the previous method with $O(|V|^3)$.

For dense graphs, however, i.e., for graphs where $|E| \sim |V|^2$ holds, the running time reaches towards $O(|V|^3)$ as well, and, with its additional overhead, it is actually slower than the former algorithm.

Since our use case of the Dijkstra algorithm involves only the computation of the shortest paths between the vertices of a given DGE set V' with $|V'| \ll |V|$ this approach is best. Also, Dijkstra's algorithm allows the extraction of the edges forming the shortest paths for further analysis.

This algorithm is described in the pseudocode listing 5.2.

Require: Graph $G(V, E)$
Require: Edge weights $\omega(e_{ij})$
Require: Start node x
1: **for all** $i \in V$ **do**
2: $dist(i) \leftarrow \infty$
3: $prev(i) \leftarrow undefined$
4: **end for**
5: $dist(x) \leftarrow 0$
6: $Q \leftarrow V$
7: **while** $Q \neq \emptyset$ **do**
8: select vertex $i \in Q$ with minimal $dist(i)$
9: $Q = Q \setminus \{i\}$
10: **for all** $j \in Q, \exists\ e_{ij} \in E$ **do**
11: $alt = dist(i) + \omega(e_{ij})$
12: **if** $alt < dist(j)$ **then**
13: $dist(j) \leftarrow alt$
14: $prev(j) \leftarrow i$
15: **end if**
16: **end for**
17: **end while**
Ensure: Shortest path weights from x to all other nodes in $dist$
Ensure: Previous nodes on the paths in $prev$

Algorithm 5.2: The Dijkstra algorithm

Dijkstra's algorithm begins with a valid solution, i.e., the minimum distance from the start node to itself is zero, and incrementally builds the shortest paths by successively taking the next best node that has not been visited yet (thereby also ensuring the method's correctness) and inspecting its neighbors for an alternate, shorter route than the one already available.

This algorithm is also the starting point for the variant multiple maximum value paths approach, as we will see in the next section.

5.2.2 Variant Multiple Maximum Value Paths

Given a maximum value path p_{max} between two specific vertices x and y with a total weight of $\omega(p_{max})$, we define the set of δ-variant multiple maximum value paths (V-MVP) as the set of paths $P(x, y)$, such that $\forall\ p \in P(x, y)$, $\omega(p) \geq \delta \cdot \omega(p_{max})$ holds. δ is defined in the interval $[0, 1]$.

Please remember, however, that we have transformed our MVP problem into an SP problem (p_{max} is now p_{min}). In this context, we also need to remap our variance constraint: we are not looking for shorter paths, but for longer paths,

i.e., paths with length of maximum $(1 + \delta)$ times the shortest path length, for a variance factor defined as $\delta \geq 0$. For convenience and an improved readability we will henceforth use this latter definition of δ!

The starting point for the calculation of the variant paths is the Dijkstra algorithm. Since V-MVP only make sense if calculated for a relatively small input set of vertices, like a DGE set, we can obtain the maximum distances from each of the regarded nodes to all other nodes in the graph by running the Dijkstra algorithm first. We will then use this information to construct all paths respecting the specified weight constraint.

We developed two approaches. The first one follows a dynamic programming principle, while the second one is a more efficient bounded reversed depth-first-search (BR-DFS). The original solution needed to be re-engineered due to the exponential behavior of the V-MVP count in regard to the δ-coefficient. We will describe this correlation in detail, later in this chapter.

Dynamic Programming

This approach starts from the premises that at least one solution exists (for $\delta = 0$) and begins building this path backwards, from the target node.

Let x and y be the source respectively the target of the paths we are constructing. We define an *ethereal path* as a triplet $S(z, FP, \omega(FP))$, describing a partially fixed path $FP = (z, k_1, k_2, .., k_m, y)$ from z to y and its associated weight $\omega(FP)$. For all such partial solutions, the constraint $dist(z) + \omega(FP) \leq (1 + \delta) \cdot \omega(p_{min})$ must hold. This means that at least one complete path from x to y exists over the edge sequence FP.

Please note that $dist(z)$ is the shortest distance possible between x and z as obtained by the Dijkstra algorithm, and that we are using a $\leq (1 + \delta)$ constraint due to the MVP $\rightarrow$ SP transformation.

The start solution is $S_0(y, FP_0 = \emptyset, 0)$. A complete solution has the following structure: $S_n(x, FP_n = (x, k_1, k_2, .., k_n, y), \omega(FP_n))$.

In each step, we choose one valid ethereal solution from the available set and inspect all predecessors of the current node z. Should a predecessor w of z exist, for which the following constraint holds: $dist(w) + \omega(e_{wz}) + \omega(FP) \leq (1 + \delta) \cdot \omega(p_{min})$, we add the new solution $S(w, FP = (w, z, k_1, k_2, .., k_m, y), \omega(FP_n) + \omega(e_{wz}))$ to the available partial paths set.

This procedure is depicted in the pseudocode listing 5.3.

Bounded Reversed Depth-First-Search

This approach makes use of the *dist* vector provided by the Dijkstra algorithm as well. Contrary to the previous solution, we will not provide the pseudocode

Require: Graph $G(V, E)$
Require: Source node x, target node y
Require: Path length variance factor δ
Require: Shortest distances from x to all nodes in the graph in $dist$
1: $Sol \leftarrow \{S_0(y, FP_0 = \emptyset, 0)\}$
2: **for all** $S(z, FP = (z, k_1, k_2, .., k_m, y), \omega(FP)) \in Sol, z \neq x$ **do**
3: **for all** $w \in V, \exists\ e_{wz} \in E$ **do**
4: **if** $dist(w) + \omega(e_{wz}) + \omega(FP) \leq (1 + \delta) \cdot \omega(p_{min})$ **then**
5: $Sol \leftarrow Sol \cup \{S(w, FP = (w, z, k_1, k_2, .., k_m, y), \omega(FP) + \omega(e_{wz}))\}$
6: **end if**
7: **end for**
8: $Sol \leftarrow Sol \setminus \{S\}$
9: **end for**
Ensure: Sol holds all paths from x to y respecting the length constraint

Algorithm 5.3: A dynamic programming approach to the V-MVP

for this approach since it is basically a trivial depth-first-search [80] (DFS) with minor adjustments.

Again, we will build the paths backwards beginning the DFS in the target node and working our way towards the source. The procedure is therefore reversed. The direction of the search is dictated by the *dist* vector, i.e., from a given current node z, the only nodes reachable are the ones with a *dist* value lower than $dist(z)$. The DFS is therefore bounded. This condition also ensures that the algorithm stops after a finite number of steps, when *dist* reaches zero, i.e., when the search inevitably reaches x.

In contrast to the standard DFS, the visited nodes do not remain marked after the search retreats from the subgraph. This means they can be visited again later. This property facilitates a combinatorial generation of all possible paths. Although the exponential number of paths is of little biological relevance, it does provide statistical information on the underlying graph's structure.

5.2.3 Heuristic Spanning Trees

Two well-known algorithms for the computation of maximum spanning trees are described by Kruskal and Prim [22].

Prim is intuitively the algorithm we need, since it works with a small set of edges adjacent to the regarded start node, in our case adjacent to the *set* of start nodes. Furthermore, Prim is better suited for dense graphs, which is also relevant considering the quadratic amount of edges our graph holds.

On the other hand, Kruskal is the only one able to handle sub-tree merging. Prim has, per definition only one connected component. Joining the default

features of the two algorithms is trivial. Doing so while upholding the sub-tree depth constraint we mentioned previously is not.

Let us consider a specific example. Let x and y be two input nodes for which we wish to calculate their maximum spanning trees, while upholding a maximum tree depth constraint of two. Let us assume that after two iterations the first tree has evolved to $x \rightarrow k_1 \rightarrow k_2$. Due to the depth constraint, no new nodes can be appended after k_2. Now let us assume, the next edge available is $y \rightarrow k_2$. This would extend the constraint on the connected component for one more edge adjacent to k_2. Providing an optimal solution without falling back to a combinatorial approach is non-trivial.

We introduce a heuristic approach to this issue based on Prim's algorithm extended with Kruskal's ability to merge subgraphs. Also, each node we visit keeps track of its depth in its connected component. When reaching a vertex that has already been visited, we have two options: either the regarded node has a lower depth value that the current one, which means a shorter path to it already exists, or the regarded node has a higher depth value than the current one, which means we have just found a shorter path to it. In the latter case, we must update its depth value, e.g., we would have to update the depth of k_2 in the previous example.

This approach is depicted in the pseudocode listing 5.4.

5.3 Case Studies

In the course of this section we will introduce two specific datasets we used to analyze our research approach.

5.3.1 B-Cell Lymphoma

Lymphoma is a class of cancer diseases originating in the lymphocytes. One specific form is the B-cell lymphoma (BCL), affecting, as the name says, the B-cells, which play a key role in the humoral immune response.

The analysis procedure in this specific case is to map an experimentally determined list of significantly differentially expressed genes (in cancerous cells) on our reference dependency graph in an attempt to extract context information on the processes taking place in the diseased cells. The ultimate goal is to improve hypothesis generation for drug target discovery. As this is a computer science thesis, we will not go into more biological detail, but instead, we will analyze the method itself from a statistical point of view.

For the B-cell lymphoma we extracted genes from the Oncomine database [81] which are reported to show significant levels of differential regulation. Five studies were extracted from three publications [82, 83, 84], characterizing B-cell chronic lymphocytic leukemia, diffuse large B-cell lymphoma, and mantle cell lymphoma.

The five DGE lists named ONCO 389, 41, 40, 36 and 863 have 52, 169, 182, 388 and 890 genes, respectively.

5.3.2 Ischemic Reperfusion Injury

Ischemic reperfusion injury (IRI) refers to tissue damage caused by a returning blood supply after a period of ischemia, i.e., blood supply restriction. IRI is a negative side-effect occurring during organ transplantation that can lead, among others, to delayed graft function (DGF).

We are analyzing IRI in the context of kidney transplantation. Two types of donors exist: living and deceased. In case of deceased donors it is of critical importance to determine ex ante if the donated organ will exhibit primary, or delayed graft function / acute transplant renal failure.

Similarly to the B-cell lymphoma case study, we map a list of features, i.e., in this case a list of biomarker candidates for IRI on our reference dependency graph in an attempt to gain additional context information on the cellular processes involved. The ultimate goal is to improve hypothesis generation for biomarker discovery in order to facilitate the early detection of DGF.

For our IRI analysis we use a consolidated feature list consisting of 12 protein markers reported in the context of reperfusion injury in [85] and 25 biomarker candidates reported in the literature [86]. The non-redundant list holds 25 genes.

5.4 Results

In this section we will elaborate on the results obtained with our three biological neighborhood expansion methods for the BCL and IRI datasets.

5.4.1 Single Maximum Value Paths

Our analysis setup is defined in the following.

Let L be an arbitrary list of n genes for which a significant differential regulation has been experimentally determined. Let $C = \{0.5, 1.0, 1.3, 1.5, 1.7\}$ be the list of analyzed cutoffs.

- The first step consists of computing the single maximum value paths (S-MVP) between all pairs of genes $(x, y) \in L \times L$ for all cutoffs in C.
- The second step consists of generating 100 lists RL_i of size n with genes chosen randomly from our reference dependency graph.
- The third step consists of computing the S-MVP between all pairs of genes $(x, y) \in RL_i \times RL_i$ for all cutoffs in C, for all sets RL_i.

The random genes are picked from the complete reference graph, i.e., from the -1.0 cutoff. We will address the implications of this measure later in this chapter. Moreover, the previous procedure is executed both for *weighted* and *unweighted* paths ($\omega(e_{ij}) = 1, \forall\ e_{ij} \in V$).

We analyze the results by building a histogram of the path length distribution for the list L together with the average over all RL_i and their 95% confidence interval (CI). Further, we draw boxplots for the number of symbols in the random lists that have been found in each cutoff in C.

We are investigating the hypothesis that genes in DGE lists have shorter paths between them than genes chosen randomly from our dependency network.

All BCL lists share almost identical length distributions in regard to the same cutoffs, with larger input sets exhibiting a less erratic curve behavior at higher cutoffs due to their inherent large size. For this reason, we did not include the histograms of the single lists. Instead, figure 5.4 depicts the path length distribution for the consolidated BCL dataset, both weighted and unweighted (edge count) side-by-side, for the cutoffs 0.5, 1.0 and 1.3.

Although a minimal left-shift of the BCL curve compared to the average case is visible at lower cutoffs, it bears no statistical relevance, and the hypothesis stated previously is therefore invalid. This conclusion is confirmed by the IRI curves which show no relevant difference between the IRI and average shortest path length.

An interesting observation, however, is that the number of paths between the BCL- and IRI-associated genes is significantly higher than the number of paths between randomly picked nodes. The distribution curve of the consolidated BCL dataset is constantly above the 95% CI and even the IRI distribution curve is statistically significant at its peak, despite the small size of the dataset: 23 found genes for the 0.5 cutoff and 13 genes for 1.0.

The cause for this significantly higher number of paths between features of the BCL and IRI datasets may be twofold:

- The nodes of the reference sets lie closer together, i.e., in the same connected components and therefore a higher number of paths exists between them, as opposed to the randomly chosen nodes which are scattered across several connected components.

- The members of the BCL and IRI reference sets share a strong bias and are therefore present at higher cutoffs while the high dropout rate of the random nodes chosen in the -1.0 cutoff is keeping the path numbers between such vertices low.

To analyze this question, we have investigated the number of found features in each respective cutoff, using boxplots. Figure 5.6 depicts the obtained results.

While all BCL datasets exhibit the same behavior as can be seen in figure 5.6 (a), with a significantly lower node dropout than in the random test cases, the IRI dataset, due probably to its small size, has a significantly lower dropout rate only for the 1.0 cutoff than the average.

To reduce the effects of a hypothetical bias we have performed the same S-MVP experiment while making sure that each random set has the same number of found genes in a certain cutoff as the reference BCL and IRI input sets.

Figure 5.7 depicts the obtained results, both for weighted and unweighted edges. The former extreme discrepancy between the BCL and random path numbers is no longer visible, the BCL curve, however still being outside the 95% CI. This method may remove an implicit bias, but if the cutoff chosen for computation lies too high, it may induce an explicit bias, by randomly picking vertices from an already small set of available nodes, see figure 5.1.

5.4.2 Variant Multiple Maximum Value Paths

The analysis setup fort the V-MVP is similar to the S-MVP with the main difference that we repeat the complete procedure for each $\delta \in [0.000, 0.250]$ at 0.025 intervals. Also, we do not analyze the distribution of the symbols and path numbers over the different cutoffs, but instead, for a given cutoff over the different δ values. By doing so, we gain insight in how the number of possible variant maximum value paths changes with an increase of the variation factor. Again, in an attempt to remove the implicit information bias, we prevent dropouts in the random sets by always choosing the vertices from the cutoff we are currently using for the computation.

Figure 5.8 depicts the obtained results for the IRI dataset at a 1.1 cutoff and for the consolidated BCL dataset at a 1.4 cutoff.

Subfigure (a) shows an exponential increase in the number of possible paths between the vertices of the IRI dataset, for a linear growth of δ. Interestingly enough, the randomly chosen nodes do not exhibit the same behavior.

Subfigure (b) shows the distribution of the number of symbols in relation to the variation factor. The IRI dataset does not show any statistically relevant difference from the random tests.

Subfigure (c) shows the results obtained for the consolidated BCL dataset. The random tests exhibit a statistically significant increase in the affinity to additional vertices as opposed to the BCL list. This contradicts the results depicted in subfigure (a), but we must bear in mind that we are working both with a larger input dataset and at a higher cutoff. The path numbers do not follow a normal distribution and can therefore not be analyzed using boxplots.

5.4.3 Heuristic Spanning Trees

As opposed to the maximum value paths, the heuristic spanning trees are intended to be a tool integrated in a supervised iterative computational process and are less suited for automated statistical analysis.

Figure 5.9 depicts two possible outcomes of the spanning tree neighborhood expansion method for the IRI dataset at a 1.1 cutoff.

The trees in subfigure (a) were generated with maximum depth constraint of 1. The resulting information describes the most important *next neighbors* of the IRI vertices. Having drastically restricted the depth, the algorithm will visit direct neighbors regardless of the edge weights until the maximum node count criteria has been met. The topology is therefore biased, this constraint leading, for example, to star topologies even for nodes that are no hub proteins.

The trees in subfigure (b), on the other hand, were generated with a maximum depth constraint of 4. As opposed to the previous example, here, two nodes from the IRI list are sufficient to reach a relatively large section of the dependency graph. The topology plays in this case an important role, and can provide information on the function of the analyzed vertices. Please note, for example, that both highlighted nodes in subfigure (b) are necessary in order for the subgraph to remain connected. The biological plausibility of this information has, however, yet to be assessed by an expert in the field.

In this chapter we elaborated on the rationale fueling our three biological neighborhood expansion methods. We then provided the reader with algorithmic details regarding the expansion procedures. Finally we applied our approach to two specific case studies, the ischemic reperfusion injury and the B-cell lymphoma.

Require: Graph $G(V, E)$
Require: Initial set of nodes W
Require: The constraints $MaxDepth$, $MaxNodes$
for all $x \in W$ **do**
  $depth(x) \leftarrow 0$
  $component(x) \leftarrow x$
end for
$Edges \leftarrow \emptyset$
$Nodes \leftarrow W$
$PriorityQueue \leftarrow \{e_{xy} \in E \mid x \in W, y \notin W\}$
while $PriorityQueue$ not empty **do**
  get heaviest edge e_{xy} from $PriorityQueue$
  if $y \notin Nodes$ **then**
    if $depth(x) \geq MaxDepth \vee |Nodes| \geq MaxNodes$ **then**
      ignore node
    else
      $Edges \leftarrow Edges \cup e_{xy}$
      $depth(y) \leftarrow depth(x) + 1$
      $component(y) \leftarrow x$
      $Nodes \leftarrow Nodes \cup \{y\}$
      $PriorityQueue \leftarrow PriorityQueue \cup \{e_{yz} \in E \mid z \notin Nodes\}$
    end if
  else
    if $component(x) \neq component(y)$ **then**
      merge components
    end if
    if $depth(x) < depth(y) - 1$ **then**
      $depth(y) \leftarrow depth(x) + 1$
      reset $PriorityQueue$
    end if
  end if
end while
Ensure: The $Nodes$ and $Edges$ sets describe the resulting spanning trees

Algorithm 5.4: A heuristic approach to spanning trees

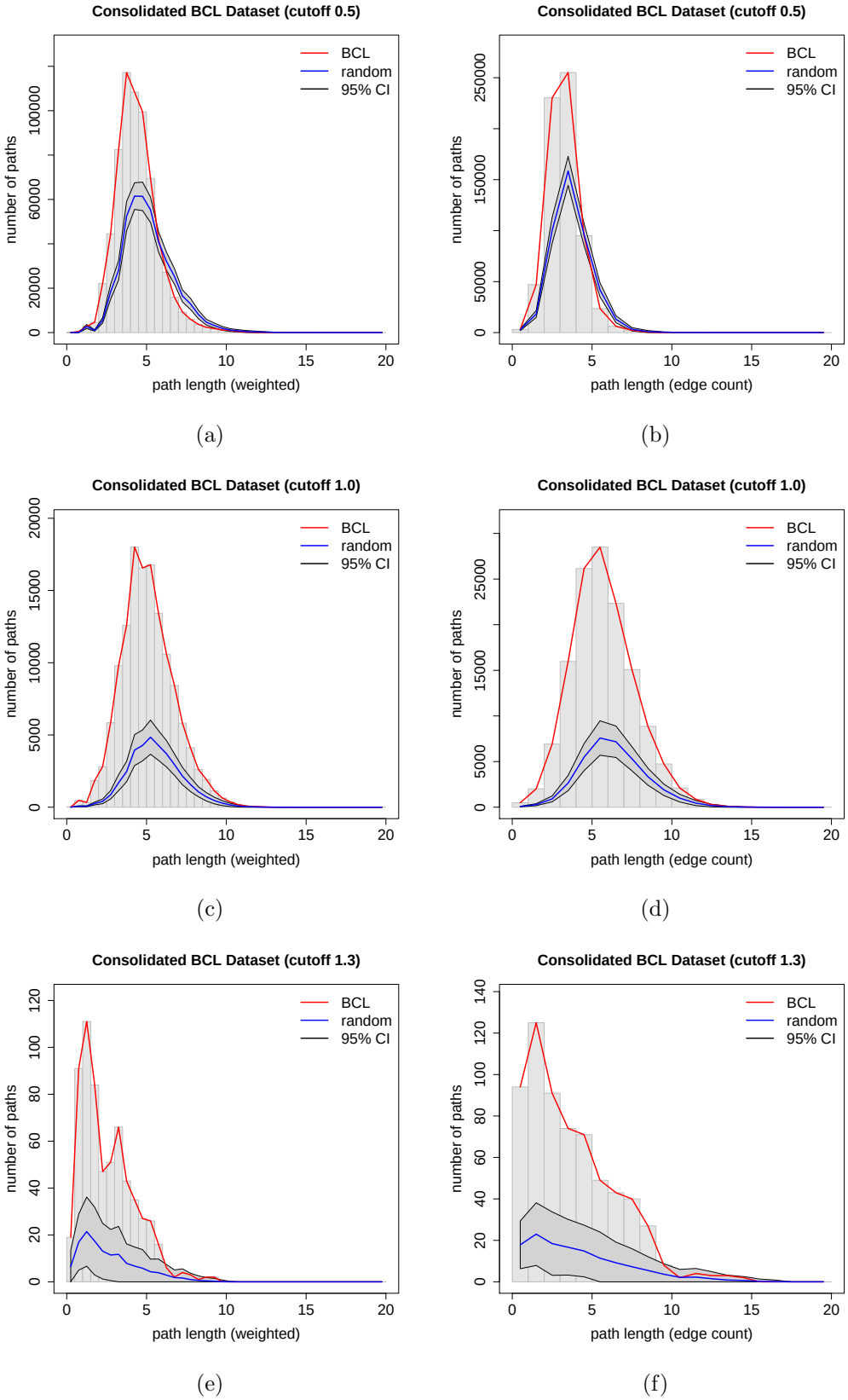


Figure 5.4: Consolidated BCL dataset S-MVP length distribution. The random lists are generated from the -1.0 cutoff.

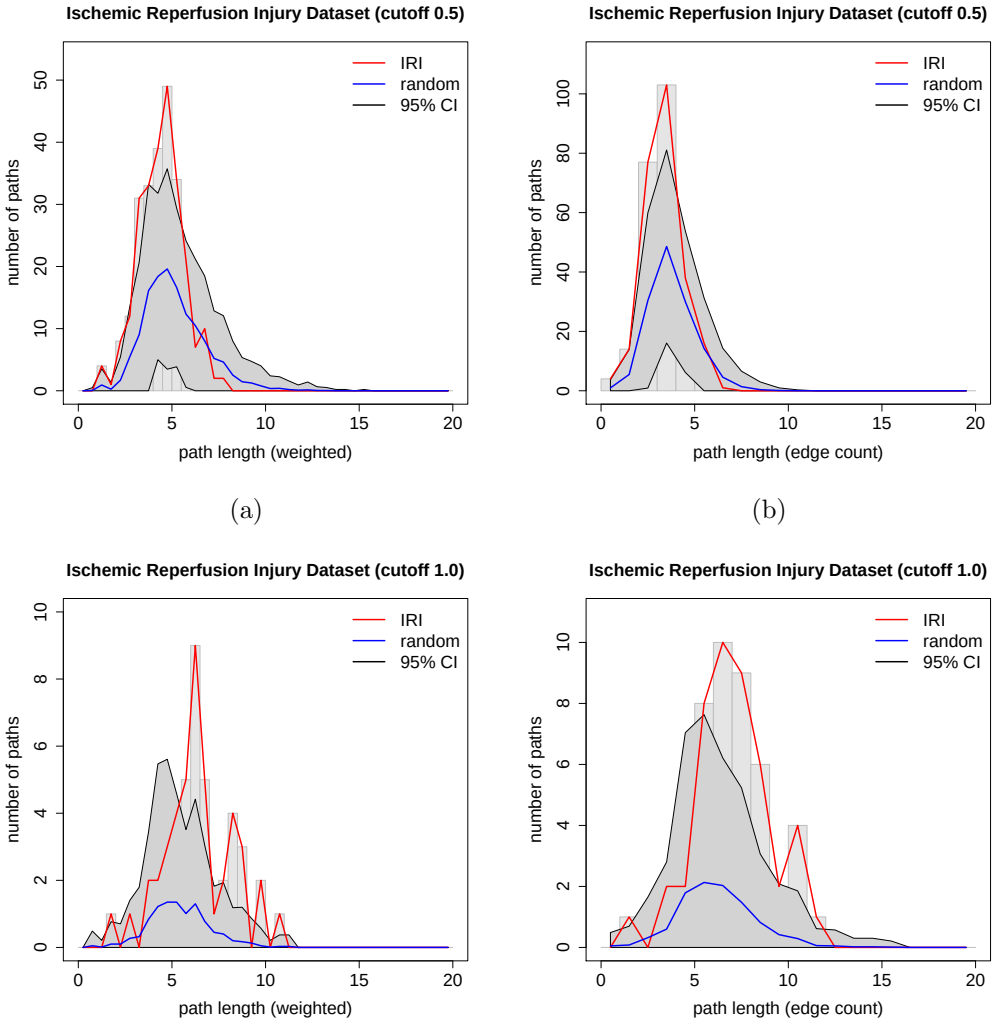
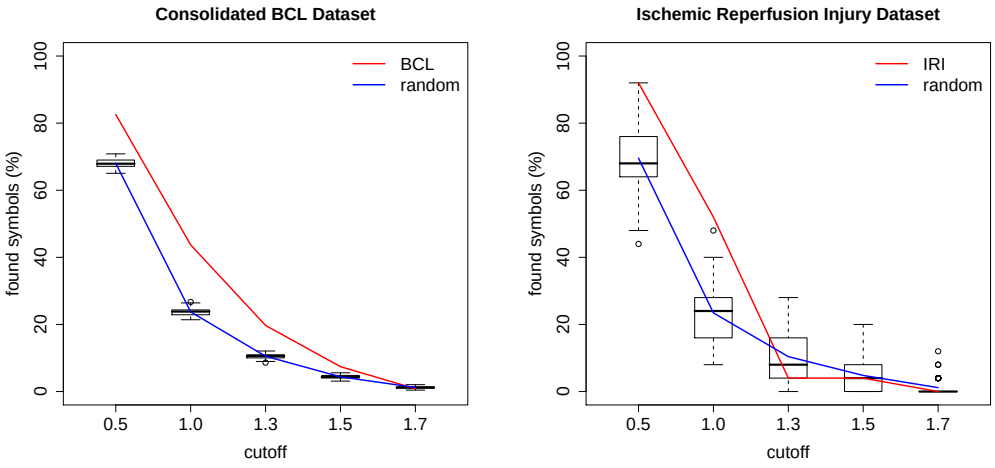
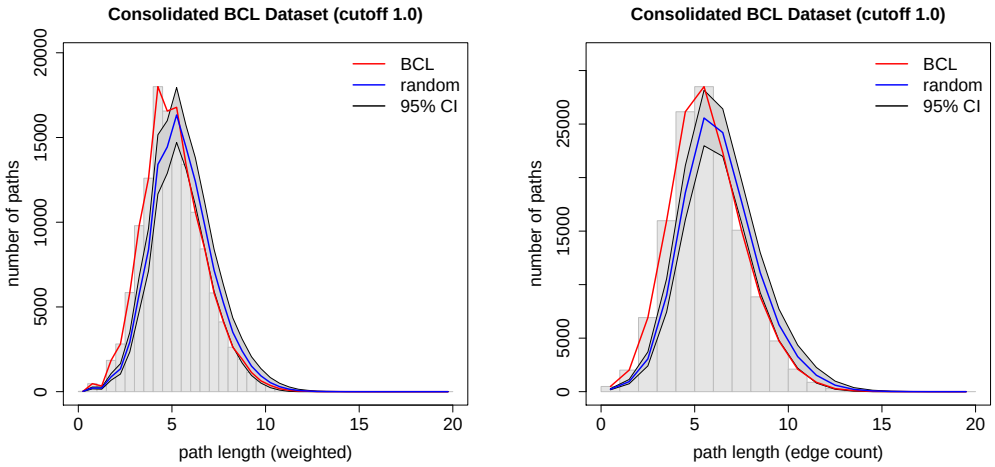


Figure 5.5: Ischemic reperfusion injury dataset S-MVP length distribution. The random lists are generated from the -1.0 cutoff.



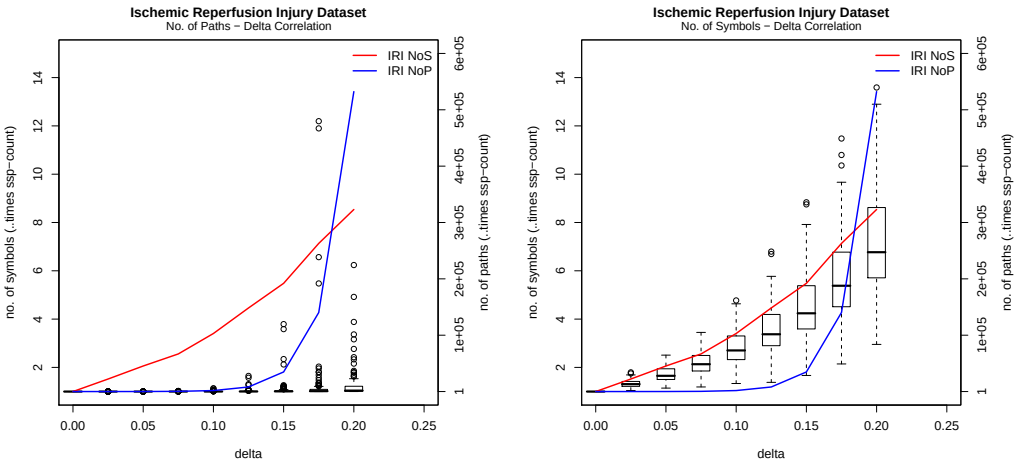
(a) (b)

Figure 5.6: The number of found symbols per cutoff

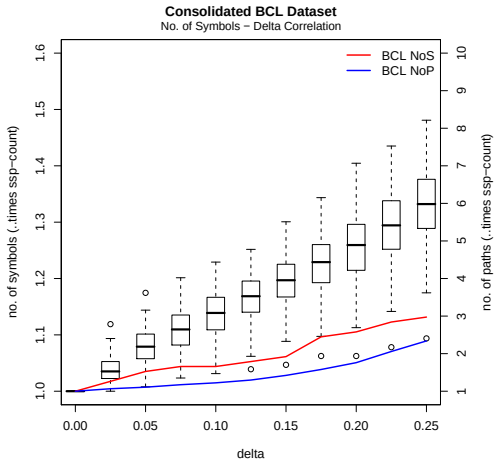


(a) (b)

Figure 5.7: Consolidated BCL dataset S-MVP length distribution. The random lists are generated from the same cutoff: 1.0

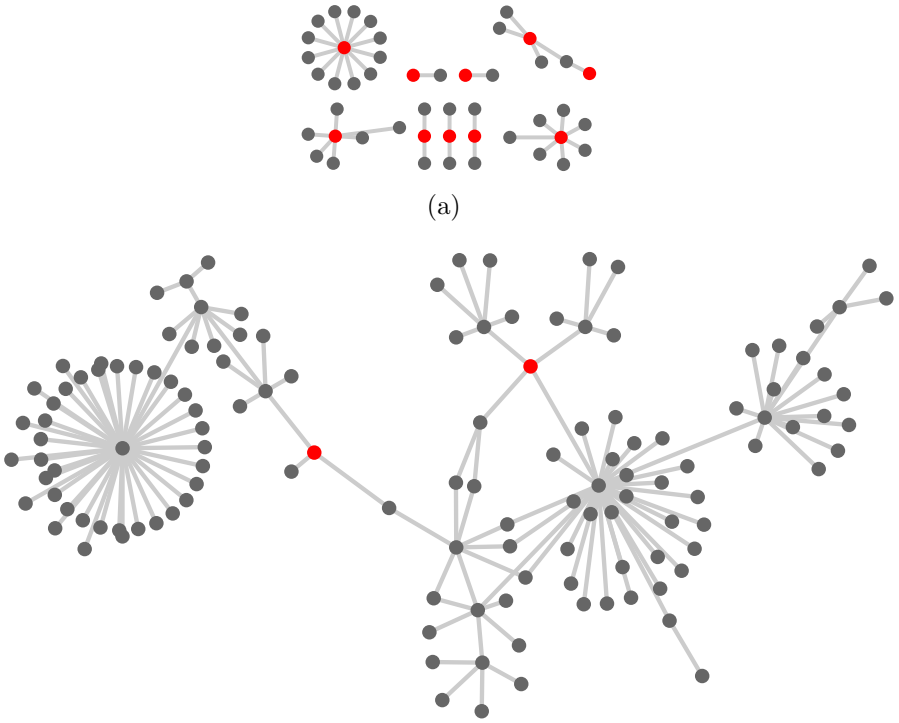


(a) (b)



(c)

Figure 5.8: NoS/NoP - delta correlation in V-MVP



(a)

(b)

Figure 5.9: Possible heuristic spanning trees for the IRI dataset

Chapter 6

CONCLUSIONS AND OUTLOOK

In this thesis we present a possible approach to information mining of biological data as means for inferring causal dependencies on the basis of descriptive data.

We introduce a modular framework providing a unified architecture for biological data integration, consolidation, as well as analysis and visualization.

The biological data integration is accomplished by means of data warehousing. We import several public domain databases into a unified structure, and provide a solution for the cross-database identifier inconsistency issue by introducing the concept of *biological hyperstructure* (HS), linking the genomic and proteomic namespaces. This mechanism also allows for a transparent extension to the transcriptomic namespace. Presently, information on approximately $71 \cdot 10^3$ HS exists in our database.

The information consolidation is accomplished by building a weighted, undirected dependency graph between all hyperstructures. For achieving this, we present a weight function based on gene expression, subcellular location, protein interaction and functional annotation data.

The subsequent analysis is implemented by means of biological neighborhood expansion. For this purpose, we propose single and multiple maximum value paths and heuristic spanning trees.

We analyze our scientific approach by running tests based on input datasets obtained on B-cell lymphoma and renal transplant ischemia reperfusion injury. Further, we demonstrate statistically significant differences between results obtained from random and reference input datasets.

Future plans and possibilities are manifold; some of the most important ones are mentioned in the following.

- *Further system development*

  This includes, among others, the integration of new data sources, the adaptation of the edge weight function by machine learning algorithms and the modification of the biological neighborhood expansion mechanisms.

- *Experimental hypothesis testing*

  Our project aims at improving hypothesis generation for laboratory experiments, with the ultimate goal of disease-associated biomarker and drug target discovery. However, our mechanisms are purely theoretical. Experimental testing has to follow.

- *Data basis improvement*

  This is still a challenge for the future. Experimental information currently available is still lacking in quality, either coming from experiment noise or from purely artificial information biasing. As methods change and improve, the quality of the results obtained with our system will improve as well.

Bibliography

[1] P. A. Corning, "The re-emergence of 'emergence': A venerable concept in search of a theory," *Complexity*, vol. 7, no. 6, pp. 18–30, 2002.

[2] K. Strange, "The end of 'naive reductionism': rise of systems biology or renaissance of physiology?," *Am J Physiol Cell Physiol*, vol. 288, pp. C968–C974, May 2005.

[3] H. Kitano, "Systems biology: a brief overview.," *Science*, vol. 295, pp. 1662–1664, Mar 2002.

[4] G. Goel, I.-C. Chou, and E. O. Voit, "Biological systems modeling and analysis: a biomolecular technique of the twenty-first century.," *J Biomol Tech*, vol. 17, pp. 252–269, Sep 2006.

[5] M. Tomita, "Whole-cell simulation: a grand challenge of the 21st century.," *Trends Biotechnol*, vol. 19, pp. 205–210, Jun 2001.

[6] D. Dori and M. Choder, "Conceptual modeling in systems biology fosters empirical findings: the mrna lifecycle.," *PLoS ONE*, vol. 2, no. 9, p. e872, 2007.

[7] P. Veber, M. Le Borgne, A. Siegel, S. Lagarrigue, and O. Radulescu, "Complex qualitative models in biology: A new approach," *Complexus*, vol. 2, no. 3-4, pp. 140–151, 2004.

[8] A. Spivey, "Systems biology: the big picture.," *Environ Health Perspect*, vol. 112, pp. A938–A943, Nov 2004.

[9] K. Aggarwal and K. H. Lee, "Functional genomics and proteomics as a foundation for systems biology.," *Brief Funct Genomic Proteomic*, vol. 2, pp. 175–184, Oct 2003.

[10] D. Hwang, J. J. Smith, D. M. Leslie, A. D. Weston, A. G. Rust, S. Ramsey, P. de Atauri, A. F. Siegel, H. Bolouri, J. D. Aitchison, and L. Hood, "A data integration methodology for systems biology: experimental verification.," *Proc Natl Acad Sci U S A*, vol. 102, pp. 17302–17307, Nov 2005.

[11] A. R. Joyce and B. . Palsson, "The model organism as a system: integrating 'omics' data sets.," *Nat Rev Mol Cell Biol*, vol. 7, pp. 198–210, Mar 2006.

[12] F. H. Crick, "On protein synthesis.," *Symp Soc Exp Biol*, vol. 12, pp. 138–163, 1958.

[13] L. You, "Toward computational systems biology.," *Cell Biochem Biophys*, vol. 40, no. 2, pp. 167–184, 2004.

[14] J. C. Smith and D. Figeys, "Proteomics technology in systems biology.," *Mol Biosyst*, vol. 2, pp. 364–370, Aug 2006.

[15] N. H. Barton, D. E. Briggs, J. A. Eisen, D. B. Goldstein, and N. H. Patel, *Evolution*. Cold Spring Harbor Laboratory Press, 2007.

[16] J. Hurwitz, "The discovery of rna polymerase.," *J Biol Chem*, vol. 280, pp. 42477–42485, Dec 2005.

[17] P. Perco, C. Pleban, A. Kainz, A. Lukas, G. Mayer, B. Mayer, and R. Oberbauer, "Protein biomarkers associated with acute renal failure and chronic kidney disease.," *Eur J Clin Invest*, vol. 36, pp. 753–763, Nov 2006.

[18] I. H. G. S. Consortium, "Finishing the euchromatic sequence of the human genome.," *Nature*, vol. 431, pp. 931–945, Oct 2004.

[19] M. Berth, F. M. Moser, M. Kolbe, and J. Bernhardt, "The state of the art in the analysis of two-dimensional gel electrophoresis images.," *Appl Microbiol Biotechnol*, vol. 76, pp. 1223–1243, Oct 2007.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, ch. 24.3, pp. 595–601. MIT Press and McGraw-Hill, 2. ed., 2001.

[21] R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, p. 345, 1962.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, ch. 23.2, pp. 567–574. MIT Press and McGraw-Hill, 2. ed., 2001.

[23] P. Flicek, B. L. Aken, K. Beal, B. Ballester, M. Caccamo, Y. Chen, L. Clarke, G. Coates, F. Cunningham, T. Cutts, T. Down, S. C. Dyer, T. Eyre, S. Fitzgerald, J. Fernandez-Banet, S. Grf, S. Haider, M. Hammond, R. Holland, K. L. Howe, K. Howe, N. Johnson, A. Jenkinson, A. Khri, D. Keefe, F. Kokocinski, E. Kulesha, D. Lawson, I. Longden, K. Megy, P. Meidl, B. Overduin, A. Parker, B. Pritchard, A. Prlic, S. Rice, D. Rios, M. Schuster, I. Sealy, G. Slater, D. Smedley, G. Spudich, S. Trevanion, A. J. Vilella, J. Vogel, S. White, M. Wood, E. Birney, T. Cox, V. Curwen, R. Durbin, X. M. Fernandez-Suarez, J. Herrero, T. J. P. Hubbard, A. Kasprzyk, G. Proctor, J. Smith, A. Ureta-Vidal, and S. Searle, "Ensembl 2008.," *Nucleic Acids Res*, vol. 36, pp. D707–D714, Jan 2008. ENSEMBL.

[24] C. Brooksbank, E. Camon, M. A. Harris, M. Magrane, M. J. Martin, N. Mulder, C. O'Donovan, H. Parkinson, M. A. Tuli, R. Apweiler, E. Birney, A. Brazma, K. Henrick, R. Lopez, G. Stoesser, P. Stoehr, and G. Cameron, "The european bioinformatics institute's data resources.," *Nucleic Acids Res*, vol. 31, pp. 43–50, Jan 2003. EBI Databases.

[25] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova, "Entrez gene: gene-centered information at ncbi.," *Nucleic Acids Res*, vol. 35, pp. D26–D31, Jan 2007.

[26] P. J. Kersey, J. Duarte, A. Williams, Y. Karavidopoulou, E. Birney, and R. Apweiler, "The international protein index: an integrated database for proteomics experiments.," *Proteomics*, vol. 4, pp. 1985–1988, Jul 2004.

[27] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M. J. Martin, D. A. Natale, C. O'Donovan, N. Redaschi, and L.-S. L. Yeh, "The universal protein resource (uniprot).," *Nucleic Acids Res*, vol. 33, pp. D154–D159, Jan 2005. UniProt Database.

[28] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider, "The swiss-prot protein knowledgebase and its supplement trembl in 2003.," *Nucleic Acids Res*, vol. 31, pp. 365–370, Jan 2003. UniProt: SwissProt (LOC).

[29] K. D. Pruitt, T. Tatusova, and D. R. Maglott, "Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins.," *Nucleic Acids Res*, vol. 35, pp. D61–D65, Jan 2007. Proteins: NCBI RefSeq.

[30] W. V. Criekinge and R. Beyaert, "Yeast two-hybrid: State of the art.," *Biol Proced Online*, vol. 2, pp. 1–38, Oct 1999.

[31] E. Phizicky and S. Fields, "Protein-protein interactions: methods for detection and analysis," *Microbiol. Rev.*, vol. 59, pp. 94–123, Mar 1995.

[32] S. Kerrien, Y. Alam-Faruque, B. Aranda, I. Bancarz, A. Bridge, C. Derow, E. Dimmer, M. Feuermann, A. Friedrichsen, R. Huntley, C. Kohler, J. Khadake, C. Leroy, A. Liban, C. Lieftink, L. Montecchi-Palazzi, S. Orchard, J. Risse, K. Robbe, B. Roechert, D. Thorneycroft, Y. Zhang, R. Apweiler, and H. Hermjakob, "Intact–open source resource for molecular interaction data.," *Nucleic Acids Res*, vol. 35, pp. D561–D565, Jan 2007. INT: IntAct.

[33] K. R. Brown and I. Jurisica, "Online predicted human interaction database.," *Bioinformatics*, vol. 21, pp. 2076–2082, May 2005. INT: OPHID.

[34] G. D. Bader, D. Betel, and C. W. V. Hogue, "Bind: the biomolecular interaction network database.," *Nucleic Acids Res*, vol. 31, pp. 248–250, Jan 2003.

[35] S. Peri, J. D. Navarro, R. Amanchy, T. Z. Kristiansen, C. K. Jonnalagadda, V. Surendranath, V. Niranjan, B. Muthusamy, T. K. B. Gandhi, M. Gronborg, N. Ibarrola, N. Deshpande, K. Shanker, H. N. Shivashankar, B. P. Rashmi, M. A. Ramya, Z. Zhao, K. N. Chandrika, N. Padma, H. C. Harsha, A. J. Yatish, M. P. Kavitha, M. Menezes, D. R. Choudhury, S. Suresh, N. Ghosh, R. Saravana, S. Chandran, S. Krishna, M. Joy, S. K. Anand, V. Madavan, A. Joseph, G. W. Wong, W. P. Schiemann, S. N. Constantinescu, L. Huang, R. Khosravi-Far, H. Steen, M. Tewari, S. Ghaffari, G. C. Blobe, C. V. Dang, J. G. N. Garcia, J. Pevsner, O. N. Jensen, P. Roepstorff, K. S. Deshpande, A. M. Chinnaiyan, A. Hamosh, A. Chakravarti, and A. Pandey, "Development of human protein reference database as an initial platform for approaching systems biology in humans.," *Genome Res*, vol. 13, pp. 2363–2371, Oct 2003.

[36] P. Pagel, S. Kovac, M. Oesterheld, B. Brauner, I. Dunger-Kaltenbach, G. Frishman, C. Montrone, P. Mark, V. Stmpflen, H.-W. Mewes, A. Ruepp, and D. Frishman, "The mips mammalian protein-protein interaction database.," *Bioinformatics*, vol. 21, pp. 832–834, Mar 2005.

[37] A. Zanzoni, L. Montecchi-Palazzi, M. Quondam, G. Ausiello, M. Helmer-Citterich, and G. Cesareni, "Mint: a molecular interaction database.," *FEBS Lett*, vol. 513, pp. 135–140, Feb 2002.

[38] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg, "The database of interacting proteins: 2004 update.," *Nucleic Acids Res*, vol. 32, pp. D449–D451, Jan 2004.

[39] B.-J. Breitkreutz, C. Stark, T. Reguly, L. Boucher, A. Breitkreutz, M. Livstone, R. Oughtred, D. H. Lackner, J. Bhler, V. Wood, K. Dolinski, and M. Tyers, "The biogrid interaction database: 2008 update.," *Nucleic Acids Res*, vol. 36, pp. D637–D640, Jan 2008. INT: BioGrid.

[40] M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes.," *Nucleic Acids Res*, vol. 28, pp. 27–30, Jan 2000. PHE: KEGG.

[41] H. Mi, B. Lazareva-Ulitsky, R. Loo, A. Kejariwal, J. Vandergriff, S. Rabkin, N. Guo, A. Muruganujan, O. Doremieux, M. J. Campbell, H. Kitano, and

P. D. Thomas, "The panther database of protein families, subfamilies, functions and pathways.," *Nucleic Acids Res*, vol. 33, pp. D284–D288, Jan 2005.

[42] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, "Gene ontology: tool for the unification of biology. the gene ontology consortium.," *Nat Genet*, vol. 25, pp. 25–29, May 2000. PHE: GO.

[43] P. Horton, K.-J. Park, T. Obayashi, N. Fujita, H. Harada, C. J. Adams-Collier, and K. Nakai, "Wolf psort: protein localization predictor.," *Nucleic Acids Res*, vol. 35, pp. W585–W587, Jul 2007. LOC: WPSORT.

[44] G. S. Omenn, D. J. States, M. Adamski, T. W. Blackwell, R. Menon, H. Hermjakob, R. Apweiler, B. B. Haab, R. J. Simpson, J. S. Eddes, E. A. Kapp, R. L. Moritz, D. W. Chan, A. J. Rai, A. Admon, R. Aebersold, J. Eng, W. S. Hancock, S. A. Hefta, H. Meyer, Y.-K. Paik, J.-S. Yoo, P. Ping, J. Pounds, J. Adkins, X. Qian, R. Wang, V. Wasinger, C. Y. Wu, X. Zhao, R. Zeng, A. Archakov, A. Tsugita, I. Beer, A. Pandey, M. Pisano, P. Andrews, H. Tammen, D. W. Speicher, and S. M. Hanash, "Overview of the hupo plasma proteome project: results from the pilot phase with 35 collaborating laboratories and multiple analytical groups, generating a core dataset of 3020 proteins and a publicly-available database.," *Proteomics*, vol. 5, pp. 3226–3245, Aug 2005. LOC: Plasma.

[45] T. Barrett, D. B. Troup, S. E. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. F. Kim, A. Soboleva, M. Tomashevsky, and R. Edgar, "Ncbi geo: mining tens of millions of expression profiles–database and tools update.," *Nucleic Acids Res*, vol. 35, pp. D760–D765, Jan 2007. GEX: BodyMap Dataset.

[46] M. Wiesinger, "In-silico prediction of transcription factor binding site in the human genome," Master's thesis, FH Obersterreich, July 2007.

[47] L. Strmbck and P. Lambrix, "Representations of molecular pathways: an evaluation of sbml, psi mi and biopax.," *Bioinformatics*, vol. 21, pp. 4401–4407, Dec 2005.

[48] M. Hucka, A. Finney, B. J. Bornstein, S. M. Keating, B. E. Shapiro, J. Matthews, B. L. Kovitz, M. J. Schilstra, A. Funahashi, J. C. Doyle, and H. Kitano, "Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (sbml) project.," *Syst Biol (Stevenage)*, vol. 1, pp. 41–53, Jun 2004.

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

[50] A. Finney and M. Hucka, "Systems biology markup language: Level 2 and beyond.," *Biochem Soc Trans*, vol. 31, pp. 1472–1473, Dec 2003. Format: SBML.

[51] B. E. Shapiro, A. Levchenko, E. M. Meyerowitz, B. J. Wold, and E. D. Mjolsness, "Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations.," *Bioinformatics*, vol. 19, pp. 677–678, Mar 2003.

[52] I. Goryanin, T. C. Hodgman, and E. Selkov, "Mathematical simulation and analysis of cellular metabolism and regulation.," *Bioinformatics*, vol. 15, pp. 749–758, Sep 1999.

[53] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. A. Hutchison, "E-cell: software environment for whole-cell simulation.," *Bioinformatics*, vol. 15, pp. 72–84, Jan 1999.

[54] P. Mendes, "Biochemistry by numbers: simulation of biochemical pathways with gepasi 3.," *Trends Biochem Sci*, vol. 22, pp. 361–363, Sep 1997.

[55] H. M. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano, "Next generation simulation tools: the systems biology workbench and biospice integration.," *OMICS*, vol. 7, no. 4, pp. 355–372, 2003.

[56] C. T. Brown, A. G. Rust, P. J. C. Clarke, Z. Pan, M. J. Schilstra, T. D. Buysscher, G. Griffin, B. J. Wold, R. A. Cameron, E. H. Davidson, and H. Bolouri, "New computational approaches for analysis of cis-regulatory networks.," *Dev Biol*, vol. 246, pp. 86–102, Jun 2002.

[57] C. J. Morton-Firth and D. Bray, "Predicting temporal fluctuations in an intracellular signalling pathway.," *J Theor Biol*, vol. 192, pp. 117–128, May 1998.

[58] L. M. Loew and J. C. Schaff, "The virtual cell: a software environment for computational cell biology.," *Trends Biotechnol*, vol. 19, pp. 401–406, Oct 2001.

[59] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka, "Libsbml: an api library for sbml.," *Bioinformatics*, vol. 24, pp. 880–881, Mar 2008. Format: SBML.

[60] P. D. Karp, M. Riley, M. Saier, I. T. Paulsen, J. Collado-Vides, S. M. Paley, A. Pellegrini-Toole, C. Bonavides, and S. Gama-Castro, "The ecocyc database.," *Nucleic Acids Res*, vol. 30, pp. 56–58, Jan 2002.

[61] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. R. Gopinath, G. R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein, "Reactome: a knowledgebase of biological pathways.," *Nucleic Acids Res*, vol. 33, pp. D428–D432, Jan 2005.

[62] M. Aladjem, G. D. Bader, E. Brauner, M. P. Cary, K. Dahlquist, E. Demir, P. D'Eustachio, K. Fukuda, F. Gibbons, M. Gillespie, R. Goldberg, C. Hogue, M. Hucka, G. Joshi-Tope, D. Kane, P. Karp, T. Klein, C. Lemer, J. Luciano, E. Pichler, D. Marks, N. Maltsev, E. Marland, E. Neumann, S. Paley, J. Pick, J. Rees, A. Regev, A. Ruttenberg, A. Rzhetsky, C. Sander, V. Schachter, I. Shah, A. Splendiani, M. Syed, E. Wingender, G. Wu, and J. Zucker, *BioPAX Biological Pathways Exchange Language Level 2, Version 1.0 Documentation*. BioPAX Workgroup, Dec 2005.

[63] H. Hermjakob, L. Montecchi-Palazzi, G. Bader, J. Wojcik, L. Salwinski, A. Ceol, S. Moore, S. Orchard, U. Sarkans, C. von Mering, B. Roechert, S. Poux, E. Jung, H. Mersch, P. Kersey, M. Lappe, Y. Li, R. Zeng, D. Rana, M. Nikolski, H. Husi, C. Brun, K. Shanker, S. G. N. Grant, C. Sander, P. Bork, W. Zhu, A. Pandey, A. Brazma, B. Jacq, M. Vidal, D. Sherman, P. Legrain, G. Cesareni, I. Xenarios, D. Eisenberg, B. Steipe, C. Hogue, and R. Apweiler, "The hupo psi's molecular interaction format–a community standard for the representation of protein interaction data.," *Nat Biotechnol*, vol. 22, pp. 177–183, Feb 2004. Format: PSI-MI.

[64] C. von Mering, L. J. Jensen, M. Kuhn, S. Chaffron, T. Doerks, B. Krger, B. Snel, and P. Bork, "String 7–recent developments in the integration and prediction of protein interactions.," *Nucleic Acids Res*, vol. 35, pp. D358–D362, Jan 2007.

[65] C. von Mering, L. Jensen, B. Snel, S. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M. Huynen, and P. Bork, "String: known and predicted protein-

protein associations, integrated and transferred across organisms," *Nucleic Acids Research*, vol. 33, pp. 433–437, 2005.

[66] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering, "String 8–a global view on proteins and their functional interactions in 630 organisms.," *Nucleic Acids Res*, Oct 2008.

[67] PubMed library: `http://www.ncbi.nlm.nih.gov/pubmed/`, Nov 2008.

[68] A. Platzer, P. Perco, A. Lukas, and B. Mayer, "Characterization of protein-interaction networks in tumors.," *BMC Bioinformatics*, vol. 8, p. 224, 2007.

[69] A.-L. Barabsi and Z. N. Oltvai, "Network biology: understanding the cell's functional organization.," *Nat Rev Genet*, vol. 5, pp. 101–113, Feb 2004.

[70] P. J. Ingram, M. P. H. Stumpf, and J. Stark, "Network motifs: structure does not determine function.," *BMC Genomics*, vol. 7, p. 108, 2006.

[71] J. M. Hellerstein, M. Stonebraker, and J. Hamilton, "Architecture of a database system," *Found. Trends databases*, vol. 1, no. 2, pp. 141–259, 2007.

[72] ORACLE: `http://www.oracle.com/`, Nov 2008.

[73] PostgreSQL Database: `http://www.postgresql.org/`, Nov 2008.

[74] MySQL Database: `http://www.mysql.com/`, Nov 2008.

[75] C. Bauer and G. King, *Java Persistence with Hibernate*. Manning Publications Co., 2007.

[76] M. Keith and M. Schincariol, *Pro EJB 3: Java Persistence API*. Apress, 1 ed., May 2006.

[77] W. Kent, "A simple guide to five normal forms in relational database theory," *Commun. ACM*, vol. 26, no. 2, pp. 120–125, 1983.

[78] Cytoscape: `http://www.cytoscape.org`, Dec 2008.

[79] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[80] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, ch. 22.3, pp. 540–549. MIT Press and McGraw-Hill, 2. ed., 2001.

[81] D. R. Rhodes, J. Yu, K. Shanker, N. Deshpande, R. Varambally, D. Ghosh, T. Barrette, A. Pandey, and A. M. Chinnaiyan, "Oncomine: a cancer microarray database and integrated data-mining platform.," *Neoplasia*, vol. 6, no. 1, pp. 1–6, 2004.

[82] A. Rosenwald, A. A. Alizadeh, G. Widhopf, R. Simon, R. E. Davis, X. Yu, L. Yang, O. K. Pickeral, L. Z. Rassenti, J. Powell, D. Botstein, J. C. Byrd, M. R. Grever, B. D. Cheson, N. Chiorazzi, W. H. Wilson, T. J. Kipps, P. O. Brown, and L. M. Staudt, "Relation of gene expression phenotype to immunoglobulin mutation genotype in b cell chronic lymphocytic leukemia.," *J Exp Med*, vol. 194, pp. 1639–1647, Dec 2001.

[83] A. Rosenwald, G. Wright, W. C. Chan, J. M. Connors, E. Campo, R. I. Fisher, R. D. Gascoyne, H. K. Muller-Hermelink, E. B. Smeland, J. M. Giltnane, E. M. Hurt, H. Zhao, L. Averett, L. Yang, W. H. Wilson, E. S. Jaffe, R. Simon, R. D. Klausner, J. Powell, P. L. Duffey, D. L. Longo, T. C. Greiner, D. D. Weisenburger, W. G. Sanger, B. J. Dave, J. C. Lynch, J. Vose, J. O. Armitage, E. Montserrat, A. Lpez-Guillermo, T. M. Grogan, T. P. Miller, M. LeBlanc, G. Ott, S. Kvaloy, J. Delabie, H. Holte, P. Krajci, T. Stokke, L. M. Staudt, and L. M. P. Project, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma.," *N Engl J Med*, vol. 346, pp. 1937–1947, Jun 2002.

[84] F. Zhan, J. Hardin, B. Kordsmeier, K. Bumm, M. Zheng, E. Tian, R. Sanderson, Y. Yang, C. Wilson, M. Zangari, E. Anaissie, C. Morris, F. Muwalla, F. van Rhee, A. Fassas, J. Crowley, G. Tricot, B. Barlogie, and J. Shaughnessy, "Global gene expression profiling of multiple myeloma, monoclonal gammopathy of undetermined significance, and normal bone marrow plasma cells.," *Blood*, vol. 99, pp. 1745–1757, Mar 2002.

[85] P. Perco, C. Pleban, A. Kainz, A. Lukas, B. Mayer, and R. Oberbauer, "Gene expression and biomarkers in renal transplant ischemia reperfusion injury.," *Transpl Int*, vol. 20, pp. 2–11, Jan 2007.

[86] I. Mühlberger, P. Perco, R. Fechete, B. Mayer, and R. Oberbauer, "Biomarkers in renal transplantation ischemia reperfusion injury." Dec 2008.