

# Extracting Information for Biology

Von der Philosophisch-Historischen Fakultät der Universität Stuttgart  
zur Erlangung der Würde eines Doktors der  
Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von

**Jasmin Šarić**

aus Bosanska Krupa (Bosnien-Herzegowina)

Hauptberichter: Apl. Prof. Dr. Uwe Reyle

Mitberichter: Prof. Dr. Christian Rohrer

Tag der mündlichen Prüfung: 16. September 2005

Institut für maschinelle Sprachverarbeitung  
Universität Stuttgart

2006



# Contents

<b>English Summary</b>	<b>7</b>
<b>German Summary</b>	<b>15</b>
<b>1 Introduction</b>	<b>21</b>
1.1 Motivation and contribution of this work . . . . .	21
1.2 The structure of this work . . . . .	24
<b>I Background</b>	<b>27</b>
<b>2 NLP technology for detecting information</b>	<b>29</b>
2.1 Information retrieval and passage retrieval . . . . .	30
2.2 Information extraction . . . . .	40
2.3 Question Answering . . . . .	47
2.4 Text understanding and DUC's . . . . .	49
<b>3 Language and Language Technology in Biology</b>	<b>51</b>
3.1 Biomedical Terminology . . . . .	51
3.2 Related work – BioCreAtIvE . . . . .	63
<b>4 System outline</b>	<b>69</b>
4.1 Establishing a collaboration . . . . .	69
4.2 What data should be extracted . . . . .	74
4.3 Conclusions for the system architecture . . . . .	78
<b>5 Summary</b>	<b>79</b>
<b>II Extracting Gene Regulatory Networks</b>	<b>81</b>
<b>6 System overview (Methods)</b>	<b>83</b>
6.1 Corpus and Filtering . . . . .	84
6.2 Tokenisation influences parsing . . . . .	86
6.3 Part-of-Speech Tagging . . . . .	91

6.4	Recognising Gene and Protein Names . . . . .	94
6.5	Semantic Tagging . . . . .	96
6.6	Recognising Entities and Relations with CASS grammars . . . . .	98
<b>7</b>	<b>Results</b>	<b>135</b>
7.1	Evaluation of Relation Extraction . . . . .	135
7.2	Evaluation of Entity Recognition . . . . .	137
7.3	Evaluation of POS-Tagging . . . . .	137
7.4	Evaluation of Tokenisation . . . . .	137
<b>8</b>	<b>Summary</b>	<b>139</b>
<b>III</b>	<b>The Language of Biological Databases</b>	<b>141</b>
<b>9</b>	<b>The Swiss-Prot Database and its FT-lines</b>	<b>143</b>
9.1	Introduction . . . . .	143
9.2	What Swiss-Prot covers . . . . .	144
9.3	Querying Swiss-Prot – state of the art . . . . .	148
<b>10</b>	<b>Adapting TIGERSearch for Cell Biology</b>	<b>153</b>
10.1	Introducing TIGERSearch . . . . .	153
10.2	Representing and querying FT-lines in TIGERSearch . . . . .	155
10.3	Database Construction . . . . .	156
10.4	The Representation and Query language . . . . .	161
10.5	Conclusion . . . . .	163
<b>11</b>	<b>Summary</b>	<b>165</b>
<b>IV</b>	<b>Modelling Biological Processes</b>	<b>167</b>
<b>12</b>	<b>Ontologies and Bio-Ontologies</b>	<b>169</b>
12.1	What is an ontology? . . . . .	173
12.2	Criteria for building an ontology . . . . .	178
12.3	Existing ontologies and taxonomies . . . . .	184
<b>13</b>	<b>Building a bio-ontology</b>	<b>193</b>
13.1	Motivation: Linguistics and Ontologies . . . . .	193
13.2	Basic gene expression concepts and relations . . . . .	196
13.3	Chemical elements and chemical bonds . . . . .	204
13.4	Building chains of nucleotides . . . . .	214
13.5	Amino acids and peptides . . . . .	231
<b>14</b>	<b>Summary</b>	<b>237</b>

<b>V</b>	<b>Conclusion</b>	<b>239</b>
14.1	Contributions . . . . .	241
14.2	Further Research . . . . .	242
<b>VI</b>	<b>Appendix</b>	<b>245</b>
<b>A</b>	<b>Swiss-Prot entries</b>	<b>247</b>
A.1	A Swiss-Prot example entry . . . . .	247
A.2	HTML version of a Swiss-Prot entry . . . . .	251
<b>B</b>	<b>The regular expression grammar</b>	<b>253</b>
B.1	Level 1 . . . . .	253
B.2	Level 2 . . . . .	254
B.3	Level 3 . . . . .	254
B.4	Level 4 . . . . .	255
B.5	Level 5 . . . . .	256
B.6	Level 6 . . . . .	258
B.7	Level 7 . . . . .	259
B.8	Level 8 . . . . .	259
B.9	Level 9 . . . . .	261
B.10	Level 10 . . . . .	262
B.11	Level 11 . . . . .	265
B.12	Level 12 . . . . .	266
B.13	Level 13 . . . . .	267
B.14	Level 14 . . . . .	268
B.15	Level 15 . . . . .	269
B.16	Level 16 . . . . .	270
B.17	Level 17 . . . . .	272
B.18	Level 18 . . . . .	273
B.19	Level 19 . . . . .	274
B.20	Level 20 . . . . .	275
B.21	Level 21 . . . . .	276
B.22	Level 22 . . . . .	278
B.23	Level 23 . . . . .	280
B.24	Level 24 . . . . .	281
B.25	Level 25 . . . . .	285
B.26	Level 26 . . . . .	288
B.27	Level 27 . . . . .	290
B.28	Level 28 . . . . .	292
B.29	Level 29 . . . . .	294
B.30	Level 30 . . . . .	295
	<b>Literature</b>	<b>297</b>

## *Contents*

# English Summary

## Motivation

High-throughput methods like the large scale sequencing of the human genome dramatically increase our knowledge of genetics and related biological processes. As a consequence these results accelerate the pace of research and development in the field of biomedicine. The overall goal of these research efforts is to obtain new findings about diseases in order to improve human health. However, these advances are responsible for an increase in complexity and a need for understanding when applying biomedical research and data. Meanwhile there is a strong agreement within life-science related academic laboratories and industry that addressing the complexity of biological data and knowledge entails intense interdisciplinary efforts. A major requirement for interdisciplinary research within life sciences is to correlate the data that is derived from text with data from experiments in biomedical laboratories (and with patient records). The main contribution of this work is to describe how natural language processing (NLP) methods and systems can fulfill this requirement by categorising, structuring and exploiting the massive amount of textual data available and in integrating the results with data derived from biomedical experiments.

## Overview

The present work is thematically divided into three parts. The first part is about text mining in the life sciences and is subdivided in two subsections. Subsection I presents an introduction to effective natural language processing techniques for identifying and retrieving information from large text collections. Furthermore it presents the characteristic features of biomedical terminology, which comprise synonymic, homonymic, orthographical, paragrammatical as well as other types of variance. This illustrates that the crucial difference between everyday language and the language used within biomedical scientific literature is mainly based on the difference of the terminology used. This subsection concludes with a description of basic criteria that an information extraction system has to meet. The implementation of such an information extraction system is described in the second subsection. This section documents a pilot study that was carried out in close collaboration with both the SDBV (Scientific DataBases and Visualisation) group of EML Research gGmbH and Peer Bork's group at the EMBL (European Molec-

ular Biology Laboratory) both located in Heidelberg. The system implemented is used for the extraction of information on gene expression relations from biomedical scientific publications.

The second part III focuses on the transfer of a computational linguistic tool (TIGERSearch), which was originally developed for the querying of hierarchical structures, to querying knowledge on protein domains from a protein database. It is demonstrated that TIGERSearch offers the possibility to make implicit knowledge about protein domains explicit by transforming the database entries to TIGERSearch-XML. In addition, TIGERSearch makes this implicit knowledge graphically visible. In fact, TigerSearch was initially developed for the querying and transparent representation of syntactically annotated corpora, so-called treebanks. This part also points out the problem that mapping the wide range of natural language annotations to precisely defined concepts presupposed by the search engine requires an ontological modelling of the domain.

The third part addresses the problem of ontological modelling in a more general and more comprehensive way. It consists of two chapters. The first chapter introduces basic notions of ontologies as well as an overview of guidelines to be considered when building an ontology. In addition some examples of implemented (both general and biomedical) ontologies are presented. The second chapter presents an axiomatisation of a sub-domain of molecular biology (*i.e.* gene expression) that comprises the domain of proteins and their domains.

The thesis demonstrates a highly interdisciplinary approach for text mining in the life sciences. Methods and knowledge from the fields of natural language processing, bioinformatics and biology have been successfully combined with knowledge from cell-biology and the problem of extracting knowledge from unstructured or partially structured data.

## Part I: Background

The first part of this thesis gives an overview of standard NLP techniques that can be utilised for effective extraction of information from biomedical scientific publications. These techniques comprise Information and Passage Retrieval, Information Extraction, and Question Answering. In addition, an overview of some standard competitions is given demonstrating how an evaluation of such systems can be carried out. The intent of these sections is not to replace an introduction to any of these NLP-subfields but rather to give an overview of possible methods for extraction and retrieval of knowledge from documents and to show their capabilities and limitations.

The methods and techniques introduced in the general background section are independent of a specific domain. However, the main obstacle for biomedical text processing is the detection, classification and identification of technical terminology. The main terminological hurdles (like synonymic, homonymic, orthographical, paragrammatical and other types of variance) are described and discussed in detail. The same section gives an overview of related work in the field of biomedical Information Extraction. Mainly the key issues that have been addressed within the *BioCreAtIvE* (*Critical Assessment of*

*Information Extraction Systems in Biology*) competition 2004 are reported. The *BioCreAtIvE* was an assessment for standardising NLP tasks for biomedical text processing.

The first chapter concludes with a description of the setting for a pilot study on biomedical information extraction. This study was carried out in collaboration with the Bork group at the European Molecular Biology Laboratory (EMBL) in Heidelberg. The main part of the chapter explains the details of the questions that the biologists at EMBL needed to have answered by such a system. It is a major achievement in this interdisciplinary setting that we managed to work out the details that such a system had to account for. On the one hand it was vital to adjust to and specify the needs of the biologists. At the same time this question had to fit an appropriate natural language processing system. It is important to mention that the precise questions we worked with explain – at least partly – the success and good results that were achieved with our system.

## Part II: Extracting Gene Regulatory Networks

Information extraction (IE) is a linguistically motivated engineering method for identification and extraction of factual knowledge from (large) text collections fitting into predefined information templates. In this part I describe a pilot study that uses IE methods to extract large-scale data with high precision from biomedical literature (PubMed abstracts). The system produces structured information that can be used for populating databases or for spreadsheet analysis. This implementation was a highly interdisciplinary piece of work and was carried out in collaboration with biologists and bioinformaticians. I worked for the SDBV (Scientific DataBases and Visualisation) group of EML Research gGmbH in close collaboration with the Bork group at the EMBL (European Molecular Biology Laboratory) to build this system.

The overall question that the system had to answer was which proteins are responsible for regulating the expression (*i.e.* transcription or translation) of which genes. As a literature resource we used PubMed abstracts. This differs from most comparable efforts. To give an insight into current research in this field I present the results that were achieved within the *BioCreAtIvE* assessment (see Yeh *et al.* (2005) and Hirschman *et al.* (2005) for details). *BioCreAtIvE* had as its main goal identifying gene-related entities in text and classifying these genes (or the corresponding gene products) according to the Gene Ontology (GO). GO was developed for the large-scale annotation of gene functions in databases (see Ashburner *et al.* (2000) for further details on Gene Ontology). Two things should be mentioned with respect to the *BioCreAtIvE* setting. One is that the identification of gene-related entities is extremely useful within biomedical text extraction systems and a comparison through shared tasks like *BioCreAtIvE* is useful. On the other hand the identification of corresponding GO terms seems to be too ambitious for NLP systems available at the moment. Nonetheless, the results show what is possible with current technology and which topics are of interest for further research.

The resulting system is able to extract a satisfactory number of relations (for yeast) comprising 422 relation chunks. Since one relation chunk can contain several binary relations we had 597 binary relations and after excluding duplicates, we retrieved (in

## Summary

a first version) 441 unique binary relations. The extracted relations can be represented as a directed graph in which each node corresponds to a gene or protein and each arc represents a pairwise relation. The "arrows" point from (known) regulators to targets. The system was able to distinguish three types of regulation, *i.e.* regulation, repression and activation. Such a graph clearly shows the distribution of the relation types of, for example, activation or repression assigned for 50% of the relations, the rest remains underspecified.

A domain expert – a biologist – evaluated the results for the yeast organism manually. The accuracy of the relations was evaluated at the semantic rather than the grammatical level. We thus carried out the evaluation in such a way that relations were counted as correct if they extracted the correct biological conclusion, even if the analysis of the sentence was not optimal from a linguistic point of view. Conversely, a relation was counted as an error if the biological conclusion was incorrect.

It should be noted that the evaluation showed that 75 of the 90 relation chunks (83%) within the pilot study extracted from the evaluation corpus were entirely correct, meaning that the relation corresponded to expression regulation, the regulator (R) and the regulated (X) were correctly identified, and the direction of regulation (up or down) was correct if extracted. Further 6 relation chunks extracted the wrong direction of regulation but were otherwise correct; our accuracy increases to 90% if we allow for this minor type of error. Approximately half of the errors made by our method stem from overlooked genetic modifications-although mentioned in the sentence, the extracted relation is not biologically relevant.

The two most important aspects of this work are the reusability of the system and the relevance of the results for the bioinformaticians. As regards reusability, it was possible to adapt the system to related questions. This means that with a series of modifications such as adaptation of the lexicon and manual extension of the set of rules, protein modifications could be extracted successfully. Results were presented at the SMBM 2005 symposium (Semantic Mining in Biomedicine) in Hinxton. This work is summarised in Šarić *et al.* (2005b) and was selected to appear in Bioinformatics 2005. The second and more important issue is that the IE system will be used in a biological scenario, which means that the results make sense biologically. The system developed will be used within a project called STRING (Search Tool for the Retrieval of Interacting Genes/Proteins). It will be integrated into the next version of STRING and will then be available for inspection at <http://string.embl.de/>.

## Part III: The Language of Biological Databases

In contrast to the sheer amount of computer-readable protein databases, only a small amount of information contained in these databases can be automatically retrieved by any of the existing query systems and data mining techniques. This is so because

1. most of the knowledge contained in these databases is not presented in a completely and satisfactorily structured way, but is expressed in text form, such as comment lines; and

2. important information is often distributed over different parts of a database without explicit cross-references connecting these parts, and hence without indication of their collective information potential.

If at all, only domain experts are capable of identifying, understanding and exploiting this information. Obviously, it is hard to figure out what information they are missing. Their capabilities are, however, constrained due to the considerable number of databases each of which contains a vast amount of entries. Hence, biological databases are a greatly underutilised knowledge source.

The aim of the work presented in this part of the thesis was to show a way of substantially increasing the amount of information that may be reliably derived from biological databases. As a case study this experiment was carried out with a sequence database, the reference protein database Swiss-Prot (for details see Boeckmann *et al.* (2003) or <http://www.expasy.org/sprot/>). I concentrated on structural properties of proteins that are contained in the database only implicitly because the structure of the database doesn't provide a particular (set of) slot(s) for it. This is the case in particular for the information given in the so-called FT-lines (feature table lines). An example of a feature table entry is shown in table 11.1.

Key word	Start	End	Length	Description
SIGNAL	1	21		
CHAIN	22	480		11S GLOBULIN BETA SUBUNIT
CHAIN	22	296		GAMMA CHAIN (ACIDIC)
CHAIN	297	480		DELTA CHAIN (BASIC)
...	...	...	...	...

Table 1: A shortened version of a representative set of FT-lines for a random Swiss-Prot entry. The FT-lines are annotated with keywords, start and end position and a description. It should be noted that the description field contains unstructured data, *i.e.* textual description of properties of the domains.

To show how advanced searching capabilities can be used in a scenario like the one described above I used the TigerSearch Engine to query the Swiss-Prot FT-lines. I present a system that has been developed to make the information contained in the FT-lines much more transparent to the user in comparison to standard search facilities. In addition, the use of TigerSearch allows the user to view the knowledge from different perspectives and combinations, which can roughly be described as an ontology-driven perspective on the FT-lines. The system for acquiring the information in the new format comprises two components:

**Translation Component:** Information that is only implicitly contained in the FT-lines is made explicit by translating the contents into a more structured representation,

that is the TigerXML representation. The additional structure relies on an implicit ontology. The thus enhanced representation of the FT-lines is therefore semantically sound and accessible to an intelligent search engine like TigerSearch;

**TigerSearch Engine:** The TigerSearch component allows the user to query the restructured databases. The query language as well as the deductive search engine incorporates the same ontological knowledge that is used in the translation component. This guarantees that a maximal amount of information can now be retrieved from the FT-lines.

Sample output of a TigerSearch tree is shown in section 10.3 in table 10.4.

At the moment relations such as dominance and precedence in the ontology-driven representation can be queried. The approach allows addition of further concepts and relations on the basis of a lexicon of biomedical terminology. Such a lexicon contains syntactic and semantic features for its entries. A further developed approach would thus allow the combination of techniques developed in Natural Language Understanding, Theorem Proving and Ontological Engineering. A team to further develop this work would need to be highly interdisciplinary, combining specialists in the fields of biology, computational linguistics and computer science (especially databases and ontologies).

## Part IV: Modelling Biological Processes

The last part of the present work is titled *Ontologies and Bio-Ontologies*. It consists of two related chapters. The first chapter explains the use of ontologies and their applications from a general point of view. The questions and items that are dealt with can be summed up as follows:

- What is the prime motivation for building ontologies?
- Why deal with ontologies? - moving towards applications.
- Why use ontologies for molecular biology?
- A scenario for using bio-ontologies.

In addition, I define what an ontology is and describe scenarios for the application of ontologies. In the following I introduce and discuss a set of criteria that are important when building ontologies. These comprise the following eight items: *expressivity of a language, complexity and efficiency, inference and interpretation, readability, clarity and visualisation, merging and integration of data, extensibility, granularity, and learning ontologies*. The first part concludes with a rough overview of different ontologies that are in use. I briefly present some general ontologies (*i.e.* Cyc and Wordnet) and in addition I present some biology-related ontologies.

The second chapter of the last part is titled *Building a bio-ontology*. In the beginning I present a linguistic justification for the use of ontologies. The guiding idea is that the previously described information extraction experiments are dependent on extensive

manual work for writing rules both for the detection of named entities and for the detection of relations on top of the named entities. It should be noted that the more a system like the one presented is extended, the more rules have to be added, and the less it is possible to control the interactions between the rules. To reduce the amount of manual work, to better control this process of adding new rules as well as achieving a higher consistency, an ontology based approach is suggested for further research. I discuss and present potential applications for the use of ontologies, *i.e.* both to make implicit knowledge in biological databases more explicit as well as in order to achieve more consistency with biological data.

The main part of this second chapter is titled *The basic gene expression concepts and relations* and it introduces briefly the biological process of gene expression on a general level. This general biology-driven picture is then axiomatised in a formal approach. The main concepts that are formally expressed are *DNA sequence* and its parts, the *Nucleotides*. In addition a rough conceptualisation of *Genome* and *Gene* is introduced as well as the corresponding part *Coding sequence* and *Non-coding sequence*. The subunits of these are introduced as well: *regulatory non-coding sequence*, *promoter*, *enhancer* and *binding site*. The products of the gene transcription and translation process are formalised in the following, being *RNA sequence*, *gene transcript*, *peptide sequence*, *protein*. Finally the formalisation on this level finishes with the relations *transcription* and *translation*.

This biology-driven formalisation is then followed by a chemistry-driven formalisation of gene expression consisting of three sections. The section *Chemical elements and chemical bonds*, section 13.3, inspects the details of chemical binding between atoms as well as binding between molecules. Within the section *Building chains of nucleotides*, section 13.4 the composition of nucleotide chains, *i.e.* DNA-strands and RNA-strands, is formally developed. The basic building blocks are the nucleotides, which themselves are built of the purine and pyrimidine bases, a sugar part (*i.e.* *ribose* and *2-deoxyribose*) and a phosphate group. The compositional structure with specific chemical properties is formalised and presented. In the last step the DNA and RNA-strands are composed of the corresponding and previously introduced nucleotides. The last section *Amino acids and peptides*, section 13.5, concludes with the potential result of gene expression, the proteins. Proteins are polymeric structures as well as the DNA strands and are formalised along the same lines.

Although roughly 100 axioms are presented, the main contribution of this modelling effort is not that all possible concepts and relations with respect to gene expression are formalised. The crucial contributions are two insights that are gained when setting up such a model. The first refers to questions of granularity. This is presented in the context moving from biological to chemical properties and showing the different perspectives within one and the same conceptualisation. The second insight is that depending on what is to be expressed (*i.e.* is the focus on the event structure of biomedical processes or is the focus on partonomic relations between biomedical entities) the logical framework has to meet the different needs and thus has to be chosen well.



# German Summary

## Motivation

Die bisherige Forschung in der Biologie war im wesentlichen auf die Untersuchung einzelner Prozesse beschränkt. Im ausgehenden 20. Jahrhundert hat sich die Genomforschung als ein neues zentrales Wissenschaftsgebiet der Biologie etabliert. Dieser Paradigmenwechsel hat die technischen Voraussetzungen geschaffen, potentiell alle Gene und Genprodukte eines Organismus zu identifizieren und zu analysieren. Dabei sollen die Funktionen der Genprodukte im Netzwerk der Lebensprozesse erforscht und aufgeklärt werden. Dieser Umbruch, der durch Projekte wie das Genomprojekt ausgelöst wurde, wird weltweit mit immer höherer Geschwindigkeit vorangetrieben. Es werden dabei enorme Mengen an Daten in unterschiedlichster Form produziert. Die Forschung in der Biologie verändert sich zunehmend hin zu maschinellen Methoden der Bioinformatik. Eine wesentliche Charakteristik ist, dass die interdisziplinäre Forschung immer mehr an Bedeutung gewinnt.

Die wissenschaftlichen Disziplinen im Umfeld von Biologie und Medizin sind durch ein rapides Anwachsen der Datenmengen gekennzeichnet. Die entstehende "Datenflut" ist vor allem im bereits genannten Bereich der Genomforschung zu beobachten. Sequenzdatenbanken, die Gen- und Proteinsequenzen verwalten zeigen ein hyperexponentielles Wachstum. Gleichzeitig mit den Datenbanken entsteht ein weiteres Datenrepositorium, das der wissenschaftlichen Publikationen in der Biomedizing. Neue Erkenntnisse im Bereich der Biologie oder Medizin werden selbstverständlich als wissenschaftliche Texte publiziert. Obwohl biomedizinische Daten im großen Maßstab über biologische Datenbanken zugänglich sind, ist ein Grossteil des biologischen Wissens nicht unmittelbar in Datenbanken abbildbar. Diese Information steht deshalb lediglich in textueller Form zur Verfügung.

Die Genomsequenzierung beschreibt lediglich den Anfang einer langen Kette biologischer Untersuchungen mit dem Ziel der funktionalen Analyse. Um dieses bei sogenannten "high-throughput" Methoden gewährleisten zu können, sind neue Formen der Interpretation biomedizinischer Experimente unabdingbar. Informationsverarbeitung mit riesigen Datenmengen unter Einbeziehung von Wissenskontexten und Hintergrundinformation wird immer wichtiger. Die dafür notwendigen Zusammenhänge sind häufig nicht vollständig strukturiert in Datenbanken abgelegt. Vielmehr sind diese fast ausschließlich in wissenschaftlichen Publikationen niedergeschrieben. Die mit Abstand größte bibliographische Datenbank der Biomedizin, PubMed, verzeichnet einen

täglichen Zuwachs von über 2000 neuen Einträgen. Insgesamt sind dort zur Zeit über 15 Millionen Publikationen verzeichnet. Gleichzeitig lässt sich feststellen, dass die Anfragen über das PubMed-Portal ebenfalls wachsen, was die zunehmende Bedeutung von wissenschaftlichen biomedizinischen Publikationen unterstreicht. Da sich lediglich die Möglichkeit einer Stichwortabfrage bietet, sind die Ergebnisse in der Regel aber nicht zufriedenstellend. Meist sind sie unvollständig und unspezifisch. Hinzu kommt, dass bei Experimenten, die große Datenmengen produzieren, die Suche extrem zeitaufwendig ist.

## Die Struktur der vorliegenden Arbeit

Im Rahmen der vorliegenden Arbeit habe ich Methoden der Computerlinguistik diskutiert, erarbeitet und eingesetzt, um eine maschinelle Extraktion biomedizinischer Daten aus wissenschaftlichen Publikationen und Datenbanken zu ermöglichen. Im wesentlichen habe ich dabei drei Themengebiete bearbeitet. Im ersten Teil der Arbeit stelle ich eine Pilotstudie vor, die die Methoden der Informationsextraktion anwendet, um für eine vordefinierte Fragestellung Antworten aus großen Mengen biomedizinischer Texte zu extrahieren. Diese Arbeit ist von unmittelbarer biologischer Relevanz. Denn nachdem diese Arbeit in enger Kollaboration zwischen der SDBV (Scientific DataBases and Visualisation) Gruppe der EML Research gGmbH und der Gruppe um Peer Bork des EMBL (Europäisches Molekularbiologie Labor) entstanden ist, wird das System zur Extraktion von Genexpressionsdaten am EMBL eingesetzt. Im zweiten Teil der Arbeit stelle ich Einsatzmöglichkeiten der TigerSearch Suchmaschine im molekularbiologischen Kontext vor. TigerSearch wurde für die Suche auf syntaktisch annotierten Sätzen entwickelt. Ich habe sie ausgewählt, um strukturierte Information über Proteindomänen transparent darzustellen und komplexe Anfragen bearbeiten zu können. Der letzte Teil der Arbeit beschäftigt sich mit der Modellierung biologischen Wissens. Dabei steht die formale Repräsentation biologisch relevanter Konzepte im Vordergrund. Als wesentliches Merkmal der vorliegenden Arbeit möchte ich den interdisziplinären Charakter betonen, vor allem, weil interdisziplinäre Forschung nicht nur in der Bioinformatik, sondern auch in anderen Forschungsfeldern zunehmend an Bedeutung gewinnt.

### Teil I: Background

Der erste Teil der Arbeit gibt einen Überblick über Standardtechniken für die Identifikation und Extraktion von strukturiertem Wissen aus unstrukturiertem Text. Die vorgestellten Methoden und Techniken sind *Information Retrieval* und *Passage Retrieval*, *Information Extraction*, *Question Answering* und *Text Understanding*. Die Methoden werden allgemein erläutert und diskutiert. Sinn und Zweck ist es, Auswahlkriterien für biomedizinische Anwendungen herauszuarbeiten. Zudem werden Methoden der Evaluierung solcher Systeme vorgestellt. Im weiteren werden die wesentlichen Merkmale biomedizinischer Terminologie herausgearbeitet. Diese stellen die wesentliche Hürde bei der Extraktion von Information aus biomedizinischen Texten dar. Dabei wer-

den unter anderem synonymische, homonymische, orthographische und paragrammatische Varianz in der spezifischen Terminologie diskutiert und analysiert. Dieser Teil schließt mit der Beschreibung der Anforderungen an eine Pilotstudie, die ich in enger Kollaboration mit der SDBV (Scientific DataBases and Visualisation) Gruppe der EML Research gGmbH und der Gruppe um Peer Bork des EMBL (Europäisches Molekularbiologie Labor) durchgeführt habe. Die Pilotstudie wird im folgenden Teil ausführlich beschrieben.

## Teil II: Extracting Gene Regulatory Networks

Dieser Teil der Arbeit beschreibt die Extraktion von Genexpressionsdaten aus PubMed Abstracts mit Hilfe der computerlinguistischen Methode der Informationsextraktion. Informationsextraktion ist eine linguistisch motivierte Technik, die es erlaubt, Antworten auf vordefinierte Fragen aus großen Textmengen zu extrahieren. Die Fragen, die das System beantworten sollte, sind:

- i Welche Proteine sind zuständig für die Regulation der Expression welcher Gene?
- ii Welche Art der Regulation liegt vor? Das heißt, ist sie unterspezifiziert, aktivierend oder hemmend?

Im Rahmen der vorgestellten Kollaboration war es uns möglich für den Hefeorganismus relevante Daten zu extrahieren. Dabei wurden 596 binäre Relationen extrahiert. Die extrahierten Relationen lassen sich als gerichtete Graphen darstellen, wobei jeder Knoten einem Gen oder einem Protein entspricht und die gerichteten Kanten den Relationen entsprechen. Die Art der Darstellung (wie im Kapitel 7 *Results* zu sehen ist) erlaubt es, die unterschiedlichen Typen der Relationen auszudrücken. Insgesamt lässt sich sagen, dass ca. 50 % der Relationen unterspezifiziert sind. Die Evaluation der Ergebnisse wurde von einem Domänenexperten (einem Biologen) durchgeführt. Dabei wurde die biologische Korrektheit der extrahierten Resultate ermittelt. Eine linguistische Evaluation wurde nicht vorgenommen.

Die Evaluation hat gezeigt, dass das System eine hohe Präzision erreicht. 90 Relationen wurden manuell evaluiert und das System erreicht 83% korrekte Resultate. In einer "nachlässigen" Evaluation, bei der die Gerichtetheit der Kanten ignoriert wurde, erreichte das System sogar 90% korrekte Resultate.

Die wichtigsten Aspekte dieser Arbeit sind zum einen die Wiederverwendbarkeit des Systems, zum anderen die biologische Relevanz der Ergebnisse. Das System, das zunächst organismusspezifisch implementiert wurde, konnte mit gleicher (und zum Teil sogar besserer) Präzision für andere Organismen eingesetzt werden. Es musste lediglich das organismusspezifische Lexikon mit Gen- und Proteinnamen adaptiert werden. Durch Modifikation und Erweiterung der Regeln konnten sogar Protein-Modifikationen erfolgreich extrahiert werden. Die Resultate dieser erweiterten Arbeit wurden beim SMBM-2005-Symposium (Semantic Mining in Biomedicine) in Hinxton im April 2005 vorgestellt. Die eingereichte Arbeit wurde (neben vier weiteren) für eine Publikation in der Bioinformatics-Zeitschrift ausgewählt. Der wohl aber wichtigere Beitrag dieser

Studie ist ihre biologische Relevanz. Das implementierte System wird dabei im Rahmen des STRING Projekts (Search Tool for the Retrieval of Interacting Genes/Proteins, <http://string.embl.de/>) eingesetzt und soll in der nächsten Version online verfügbar sein.

### Teil III: The Language of Biological Databases

Trotz der Verfügbarkeit einer beträchtlichen Anzahl an computerlesbaren Protein-Datenbanken kann lediglich ein vergleichbar geringer Teil an darin enthaltener Information automatisch extrahiert werden. Dafür gibt es zwei Gründe:

1. Ein Großteil des in diesen Datenbanken entalteten Wissens ist nich hinreichend strukturiert oder kommentiert.
2. Entscheidende Information ist oftmals über die gesamte Datenbank verstreut. Dabei gibt es keine expliziten Referenzen, die diese Teile adäquat zusammensetzen und somit das kollektive Informationspotential explizit machen.

Meistens ist es nicht einmal Domänenexperten vergönnt, diese Information zu identifizieren, zu verstehen oder weiterzuarbeiten. Es ist dabei offensichtlich, dass es nicht einfach ist, das nicht gefundene Wissen zu quantifizieren. Eine weitere Einschränkung in der Arbeit mit biologischen Datenbanken besteht in der Kombination unterschiedlicher Datenbankschemata und der enormen Datenmengen. Die geschilderte Situation zeigt deutlich, dass das Informationspotenzial biomedizinischer Datenbanken bei weitem nicht ausgeschöpft wird.

Dieser Teil der Arbeit beschäftigt sich damit, die im linguistischen Kontext entwickelte TigerSearch-Suchmaschine einzusetzen, um strukturierte Information über Proteindomänen transparent darzustellen und komplexe Anfrage bearbeiten zu können. Die zugrundeliegenden Daten wurden dabei der wichtigsten und größten Proteinsequenzdatenbank Swiss-Prot (Boeckmann *et al.* (2003) oder <http://www.expasy.org/sprot/>) entnommen. Ich habe mich dabei vor allem auf die so genannten "feature table" Einträge beschränkt. Im Rahmen der Arbeit ist es mir gelungen, die Information über Proteindomänen so zu übersetzen, dass sie in TigerSearch zur Verfügung stehen. Nun kann sowohl einfach nach komplexen Zusammenhänge gesucht werden, gleichzeitig ist eine transparente grafische Darstellung der erzielten Suchergebnisse möglich.

### Teil IV: Modelling Biological Processes

Der letzte Teil der Arbeit beschäftigt sich mit der Modellierung biologischer Prozesse. Dabei motiviere ich zunächst die Anwendung von Ontologien aus einem allgemeinen (nicht domänenspezifischen) Blickwinkel. Ich beantworte wesentliche Fragen im Bezug auf Ontologien. Ich definiere Ontologien und führe Kriterien ein, die beim Bau einer Ontologie beachtet werden sollten. Die diskutierten Kriterien reichen von Ausdrucksstärke der Repräsentationssprache bis hin zum Lernen von Ontologien aus Text. Dieser Teil endet mit der Vorstellung einiger allgemeiner und spezifisch biomedizinischer Ontologien.

Der letzte Teil beschäftigt sich mit der formalen Modellierung von Genexpression. Zunächst stelle ich dabei eine allgemeine Konzeptualisierung der Domäne vor. In dieser werden Konzepte wie *DNS-Sequenz*, *Gen* und *Protein* mit Relationen wie *Transkription* und *Translation* verknüpft. Diese eher biologisch motivierte Modellierung wird im folgenden durch eine eher chemisch motivierte Modellierung verfeinert. Dabei formalisiere ich chemische Elemente und chemische Bindung, Nukleotide und deren Polymerisierung, die chemischen Eigenschaften von DNS- und RNS-Ketten, wie auch Aminosäuren und Peptidbindungen.



# Chapter 1

## Introduction

### 1.1 Motivation and contribution of this work

In the last few years new experimental methods in the field of molecular biology and biomedicine have been developed which allow for large-scale data analysis, so-called high-throughput methods. To show the dramatic increase of data in the field of molecular biology I will briefly introduce some examples. The examples are genome sequencing, microarray analysis, pull-down assays, and sequence comparison. Note that there are several other high-throughput techniques that are not mentioned here. The main contribution of this work is to describe how natural language processing (NLP) methods and systems can help categorising, structuring and exploiting these massive amounts of data.

Genome sequencing is probably best-known from the Human Genome Project (HGP). The HGP was a 13 year project that finished its first phase in 2003. The most important result was the identification of around 30,000 genes in human DNA, which consists of a sequence of approximately 3 billion chemical base pairs. These results were stored in databases. To further analyse and process these data (especially the amount of related data) new methods had to be invented. Other organisms that were sequenced as well lead to comparable amounts of data for each of them. Details on these efforts are described in several publications. For the human organism, Venter *et al.* (2001) or Int. Human Genome Sequencing Consortium (2001) are the most relevant citations. For yeast, Goffeau *et al.* (1997) report the first results.

Another high-throughput technique is microarray technology, also frequently referred to as DNA microarrays, DNA arrays, DNA chips, gene chips. This technology makes use of the resources created by the genome projects. Its main goal is to answer the question of which genes are expressed (*i.e.* which genes are translated into a product like a protein) in a particular cell type of an organism, at a particular time and under particular conditions. As an example, experiments like these allow for comparison between normal cells and diseased cells. A microarray is a glass slide to which a specific DNA molecule is attached at a fixed location (spot). There may be tens of thousands of spots on an array. The result when a certain reaction on a spot takes place is a change in colour for each spot. This allows scientists to detect thousands of genes in a small

sample simultaneously and to analyse the expression of those genes. It is used to enable biotechnology and pharmaceutical companies to identify drug targets, in other words, to identify the proteins with which drugs actually interact. Details on this analysis technique can be found in several scientific articles. A smaller number, such as DeRisi *et al.* (1997), Cho *et al.* (1998), or Chu *et al.* (1998) report on the data.

Given the genome sequence, another type of high-throughput approach is studying the interactions and functions of gene products, *i.e.* proteins. One method for large-scale analysis of protein-protein interactions is pull-down assays. This is a method used to determine physical interaction between two or more proteins, commonly referred to as protein interactions. The result is usually a map or network of interactions. This does not imply that each of these interactions takes place in an organism. In the following this question is to be answered by detailed research on each single interaction. One such experiment has been carried out for yeast reporting thousands of protein-protein interactions detailed in Gavin *et al.* (2002).

It has to be mentioned that high-throughput methods are not only used for biological experiments but also for generating data from existing data, which again is stored in databases. Sequence comparison<sup>1</sup>, for example, computes the degree of similarity of two given sequences X and Y to each other. Similarity can be defined, for example, by the number of mutations (*i.e.* replacing, inserting, or deleting single items in the sequence) to obtain one sequence from the other<sup>2</sup>. The aim of automatic sequence comparison is large-scale detection of functional similarities from similarities in the sequence.

All these (and other) techniques have led to an explosion of data within life sciences<sup>3</sup>. Each experiment has to be presented to the scientific research community and thus brings scientific publications with it. Every year MEDLINE<sup>4</sup>, which is the most important repository for scientific publications in that field, adds more than 500,000 publications. At the time of writing this thesis it included over 15 million citations for biomedical articles dating back to the 1950's.

To get an idea what it means to only keep track of this vast amount of annual publications, suppose a group of 10 people responsible for following new publications every year. Then each of them must cover 50,000 publications per year. If each of them worked 250 days per year that breaks down to 200 publications per day. To oversee 200 publications in one day is a very demanding task and without computational help it is hardly (if at all) manageable. Nonetheless, as the methods for accessing this information in an automated fashion are not sufficiently developed, database maintainers for biological data still have to hire armies of annotators for studying scientific literature, extracting this information manually and adding it to the data repositories. However, this is only one part of the problem.

---

<sup>1</sup>Sequence comparison is applied to both DNA sequences and peptide sequences.

<sup>2</sup>It should be noted that this is a simplified view on sequence comparison and complex methods have been developed.

<sup>3</sup>Within molecular biology data explosion takes place in particular in rapidly expanding subfields of functional proteomics and transcription regulation, since a lot of effort is being invested in these fields.

<sup>4</sup>In fact, MEDLINE is a major part of PubMed (PubMed (2001)), which is a bibliographic database owned and maintained by the U.S. National Library of Medicine. It can be queried through:  
<http://www.ncbi.nlm.nih.gov/PubMed/>.

Textual data like scientific publications are often called unstructured data, in contrast to databases which usually are referred to as structured data. The term structured means to describe the fact that the storage of the data is transparent and thus easy access in an automatic fashion is possible. This is different for *biological databases* in some respects. The fact that research in the field of biology is stepping from structural data like the DNA sequence to structure analysis with a focus on functional analysis (of proteins), brings the problem of storing this new type of data in a structured, transparent and automatically accessible fashion within databases. Functional information is, however, not sufficiently understood and frequently not normalised in such a way allowing to store it in tables. Hence, free-text annotations in databases are common and databases tend to contain more and more unstructured information.

A second issue when querying biological databases is the growing complexity of biological data. To access the complete information associated with one protein usually pages of unstructured or only partially structured data have to be considered. Information on the peptide sequence, the three-dimensional structure (*i.e.* how each sequence folds), the domains of the protein (*i.e.* relevant functional parts of the protein), the functions it has (for example with which other chemical compounds it reacts), how it combines with other proteins (*i.e.* information about complex formation) and much more is offered. When querying these databases in order to analyse and thus to retrieve this data one often has to consider data from multiple relations rather than just one single table. In addition to the increase of biology-related literature, biology-related databases are steadily growing as well, in number and in size. It is therefore becoming increasingly important to extend search strategies from using pure string match to more elaborate, semantically-driven query and extraction methods.

To summarise three types of data generation for the molecular biology domain have been introduced so far: (i) the generation of experimental data, which can both result in structured data like database entries or in unstructured data like scientific publications or comment lines in database entries, (ii) the generation of structured data from structured data, and, (iii) that of generating structured data from unstructured data. The overall objective of this work is to contribute to (ii) and to (iii). With respect to (ii) this work presents an approach for retrieving information from databases by inspecting methods of making implicit knowledge explicit and thus allowing users to obtain data in a transparent fashion. With respect to (iii) this work inspects methods and approaches for the extraction of biological knowledge from textual data.

All in all, the amount of data in the field of biomedicine and bio-chemistry threatens to overwhelm the ability of researchers to take full advantage of available information. Automatic text processing and knowledge engineering is no longer just one available tool, but a necessity.

Besides all results that are described in each of the following chapters that might contribute to the fields of either NLP or to the bioinformatics or life-science community in general one of the key insights from this work is the issue of interdisciplinarity. For the complete work I continuously consulted school biology text books to learn about the biological details and to understand the biological problems. But, none of these text books could replace the extensive discussions with biologists and bioinformaticians

over the years and the insights gained through them. Biology has reached a level of complexity and data-richness that makes it impossible for a biologist to carry out his work without help from other scientific disciplines. Computational linguistics is only one discipline to contribute.

## 1.2 The structure of this work

The following work is structured in four parts. Part I presents an introduction of effective natural language processing techniques for identifying and retrieving information from large text collections. As the crucial difference between general language texts and biomedical texts is a consequence of the terminology used this chapter presents the characteristic features of biomedical terminology (such as synonymic, homonymic, orthographical, paragrammatical and other types of variance). This part concludes with a system outline that describes the basic characteristic for a specific information extraction system that is described in the second part.

Part II, *Extracting Gene Regulatory Networks*, describes a pilot study that was carried out in close collaboration between the SDBV (Scientific DataBases and Visualisation) group of EML Research gGmbH and Peer Bork's group at the EMBL (European Molecular Biology Laboratory) both located in Heidelberg. The pilot study can be described as the extraction of information on gene expression relations from biomedical scientific publications. The main contribution is the large-scale conception of this experiment. This refers both to the amount of textual data that has been processed as well as the amount of different types of structure that have been recognised and extracted. The results of the experiment show that by overcoming the terminological hurdle in information extraction, high quality results can be achieved.

Part III, *The Language of Biological Databases*, focusses on methods that have been developed in the domain of computational linguistics and offer a new view on biological data, *i.e.* protein domains. The main contribution is that implicit knowledge about protein domains from the Swiss-Prot protein sequence database (see Boeckmann *et al.* (2003) for details) can be made explicit and graphically visible with a tool called TigerSearch. TigerSearch was initially developed for the querying and transparent representation of syntactically annotated sentences, so-called treebanks.

Part IV, *Modelling Biological Processes*, presents work that is concerned with ontological modelling of biological knowledge. It consists of two chapters. The first chapter introduces basic notions of what an ontology is. I have provided an overview of guidelines which should be considered when building an ontology. In addition I present some examples of implemented general and biomedical ontologies. The second chapter presents an axiomatisation of a subdomain of molecular biology (*i.e.* gene expression). One contribution is that a series of concepts and relations with respect to gene expression are formalised in roughly 100 axioms. A second contribution is a discussion of issues of

granularity within this formalisation. This is presented in the context of moving from a biological view to a finer grained view of the chemical properties of the entities participating in gene expression.



# **Part I**

## **Background**



## Chapter 2

# NLP technology for detecting information

As already described within the introduction section, effective extraction of information from large text collections is becoming necessary more and more in the field of life sciences. To give an overview the following sections (sections 2.1, 2.2, 2.3, and 2.4) present a series of techniques for retrieving information from large text collections. The intent of this chapter is not to replace an introduction to any of these NLP-subfields but more to give an overview of possible methods to extract and retrieve knowledge from documents and show their capabilities and limitations.

As there have been a series of assessments for establishing evaluation criteria for these techniques I have chosen to present two representative ones. One is related to the field of information retrieval, the Text Retrieval Conference (TREC) where the overall goal is to evaluate document detection performance. The other is the series of Message Understanding Conferences (MUC), which was a forum for assessing and discussing progress in the natural language processing subfield of information extraction.

The section 3.1 portrays the main issues that have to be considered when dealing with biomedical terminology considering related work in this domain. This section is then followed by section 3.2, which reports on an assessment in this bio-related field and which is called BioCreAtIve (Critical Assessment of Information Extraction systems in Biology). BioCreAtIve has been introduced for the biology-related information extraction community to establish both biologically meaningful tasks for information extraction systems as well as benchmarks for the evaluation of these systems.

The last section reports on the establishment and prerequisites of a project that I have carried out with the Bork group of the European Molecular Biology Laboratory (EMBL). I introduce the background to this project and then specify details about the needs of information extraction systems to solve these domain specific questions and demands.

## Various approaches for detecting information

The rapid growth of the world wide web has led us to face an increasingly large pool of documents from various sources. Other resources like company's intranet document resources or scientific publications are further examples. In order to avoid information overload, techniques for retrieval of relevant information are necessary. We all benefit daily from the simple and efficient service that search engines like Google<sup>1</sup> offer. Other text processing techniques like document summarisation, document clustering and document relationship visualisation are achieving promising results and are already of value. Considering the amount of different technologies available to process large-scale data, some approaches yield too coarse-grained information for scientific questions. For example, document summarisation has as result sets of summarised documents, which still have to be read.

I thus restricted the methods to be considered to information retrieval (*i.e.* the fast access to documents characterised by a bag of words approach), text understanding (*i.e.* the full understanding and interpretation of text), question answering (*i.e.* natural language queries are automatically answered, where text collections are used as knowledge base) and information extraction (*i.e.* predefined, highly specific templates are automatically filled and text collections are used as knowledge base) for two reasons. First, all four have proven to be reasonable in large-scale data collections scenarios. Second, and even more important, all four have been used for specific tasks or questions. The relevant details on each of these four methods are given in the following sections.

## 2.1 Information retrieval and passage retrieval

One method to derive information from large-scale documents is to utilise the technique of information retrieval. Information retrieval can be defined as the study of systems for indexing, searching, and recalling large collections of documents. The term document retrieval is also in use and probably better describes the fact that the result to a query is a set of documents. Basically two types of document categories have to be distinguished. One category comprises image data, sounds, music, and videos usually referred to as multimedia retrieval. The second category deals with textual data and thus comprises any kinds of text collections. Although the focus of this work is on textual data, I will also present some results from the field of multimedia retrieval, especially since life-science processes not only structured data (*i.e.* data in databases) or unstructured data such as text but also image data and video data.

Generally speaking the results that an information retrieval system running on text collections produces is a set of documents, where each document should contain relevant information. If the result is a set of paragraphs the method is referred to as passage retrieval. The following subsection provides a brief description of the methods and results of textual information retrieval. For a detailed but concise introduction into the

---

<sup>1</sup><http://www.google.com>

field of textual information retrieval Baeza-Yates and Ribeiro-Neto (1999) and Jones and Willet (1997) can be recommended.

### 2.1.1 Textual information retrieval

Textual information retrieval involves several steps of processing. The overall goal is to retrieve relevant documents from a large text collection which contain relevant information in a fast and easy way.

The text collection can be online documents or a locally stored set of files. The information that the user is interested in is specified by the user through a query using an interface provided by the system. The system searches in the document collection according to a predefined algorithm and presents the results to the user as a ranked list of documents. A prototypical information retrieval system comprises four processing steps, which are: (1) an indexing step for the file to be queried, (2) transforming the query to an internal representation, (3) the calculation of the results, and (4) the presentation of the results. These processing steps are described in more details in what follows:

1. In order to be able to query a collection of documents, an analysis and efficient storage of the unstructured text collection has to be carried out. This means that the data representation has to be structured and preprocessed in such a way that efficient search strategies can be applied. Usually this means indexing of the text collection. The indexing strategy within a retrieval system influences what can be retrieved. Basically two indexing methods are possible. The first possibility is a full-text representation containing all word information that the text collection provides. However, for huge text collections this is usually not possible for reasons of memory limitation. A second possibility is indexing of reduced documents. This means that the original documents are reduced to a certain extent. The most common step is the elimination of so-called stopwords. Stopwords are words like *and*, *in*, *for*, *a*, *the*, etc. that occur in almost any text. A sentence like “The garage is responsible for the car maintenance.” is reduced to the index terms: *garage*, *responsible*, *car*, *maintenance*. Basically, there are two motivations for filtering out stopwords. One is to reduce the amount of data for reasons of efficiency. The second motivation reflects the fact that not all words are equally informative. The system discards the words that do not distinguish the documents, for example adding a word like *and* matches nearly every document. A further step in reducing the documents is to not index all the words in this reduced set of words (and their frequency of occurrence) for each document but only to index the word stems, or lemmas, (and their frequency of occurrence) for each document. The advantage of such a morphological reduction of words to their stems is that the search space (*i.e.* the amount of searchable terms) is reduced and the systems performs more efficiently. An additional step can be to combine indexing with a clustering step according to a similarity measure that classifies similar documents as belonging to the same class. This can help to retrieve additional documents that need not contain the search terms specified by the user.

2. A query which is given by the user is automatically translated into an internal representation. The internal representation is a representation that is “understood” by the system, so that it can be dealt with. The query that is given by the user can either be a natural language expression or it can be a set of terms, probably with a set of Boolean operators, usually *AND*, *OR*, and *NOT*. Both query types specify the information the user is interested in. The details of the internal query representation usually follows the same principles as the indexing method to assure the comparability of query and document representation.
3. The calculation of the results typically follows one of the three major retrieval models. These are the Boolean model, the vector based model and the probabilistic model, which are introduced below. It has to be mentioned that these models are usually used in modified and adapted versions. Nonetheless the key ideas that are described here form the basis for these adaptations.
  - The `Boolean model` is based on set theory and Boolean algebra. The main idea is to represent each document and the query as a set of terms. The search terms can be connected with the Boolean operators *AND*, *OR*, and *NOT*. This boolean representation of the query describes which terms have to be present in combination and which terms should not occur. The main advantages are that this representation is easy to model and thus intuitively understandable. In addition, it produces high precision results. To give an example, a query like “*car AND maintenance*” retrieves all documents that use both terms. However the result might contain those documents that do not talk about automobiles and more on other not immediately related issues but mention these terms. Since this approach produces no ranking of the results, it can lead to cumbersome selection of the right document. There might be thousands of documents that fulfil the conditions and the user is left with too many search results. Nonetheless, variants of the Boolean model provides a basis for many search engines, especially for bibliographic database search. It is common to combine a boolean model with some kind of ranking algorithm. The ranking procedure aims at presenting the most relevant results first. Generally speaking it can be said that the more documents get retrieved by a system the more important is a good ranking of these results.
  - The key idea behind `vector based models` is to transform textual data into numerical vectors and matrices. Each document or query is represented as a vector, where the value of each document (and query) vector is determined by the count of each term that does or does not occur in the document. As an example `[0, 5, 3, 0, 3, . . .]` represents a document, where the first element is not present, thus 0, the second is 5 times present etc. A document containing the term *car* five times thus differs from a document that contains the term *car* only once at least for this value. To determine the result for a given query the distances between the query vector and each document vector has to be calculated. The distance is the cosine measure between the query vector

and the document vector. A major benefit of this approach is that the calculation yields a ranking of the results. The more similar two vectors are, the higher the corresponding document is ranked. In addition, it allows for partial matching since the most similar document is chosen even if not all query terms match. At least two drawbacks of this approach exist. First, the system does not consider term dependencies, like *car* and *automobile*. (However, it has to be mentioned that extensions like latent semantic indexing overcome this hurdle.) Second, it cannot deal with negation, which limits the expressivity of the query language. All in all it is the most popular retrieval technique yielding good retrieval results.

- The probabilistic model is the third variant and is based on a probabilistic interpretation of document relevance to a given user query. The system judges how relevant a document is for a specific user query based on a statistical model. A conditional probability  $p$  representing the relevance of a document is calculated given a query  $q$  and a document  $d$ . This method has the advantage of returning good results for specific document collections. In addition a ranking of the retrieved documents is provided. A major drawback is that it is computationally expensive and not applicable to huge document collections.
4. The last step is the presentation of the retrieved documents to the user. Most systems (depending on the search strategy) offer a ranked list of documents, where the first is the document with the best match. The selection of specific documents by the user is a necessary feedback for some systems as this information can be used for re-training the system.

### 2.1.2 Evaluation of textual information retrieval

The evaluation of an information retrieval system is not a straightforward task. There are several facets that can be evaluated with respect to different needs of the user and the evaluation of an IR system can easily be a time consuming task. One aspect in evaluating an IR system is the efficiency of an IR system concerning time for indexing, storage needs and runtime behaviour. This issue is more relevant for industrial use cases than for academic research, so I will not discuss it in detail. Another aspect is the effectiveness of the system, which is also evaluated. Effectiveness means whether the system is able to retrieve all relevant documents (recall) and whether the system is able to retrieve only the relevant documents (precision). Note that relevance is highly subjective and thus can be discussed controversial. Precision and Recall are the most common parameters when evaluating IR systems. This implies for the evaluation that the correct results have to be known beforehand. Of course, “relevance” is highly dependent on the user and hard to define. Usually the relevant documents get marked by a set of users and this is taken as evaluation set. Evaluation data for information retrieval systems and thus standards have been set up through a series of conferences and assessments like the TREC (Text REtrieval Conferences, see section 2.1.3).

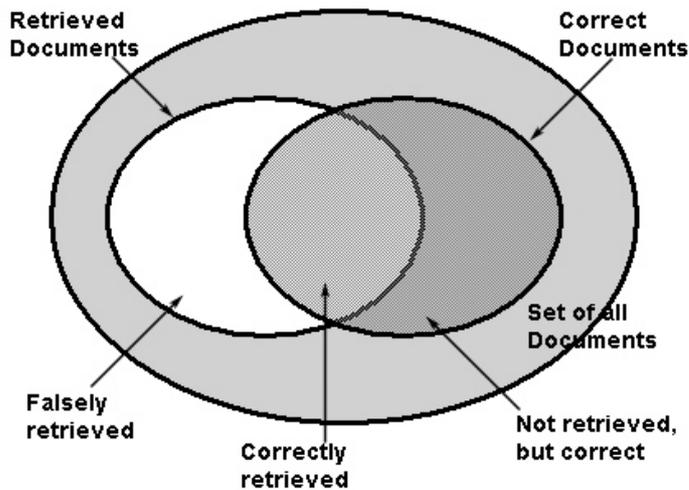


Figure 2.1: **Information retrieval evaluation:** The figure shows the three partitions that have to be considered when evaluating an IR task. These are: (i) the correctly retrieved documents, also referred to as *true positives*, (ii) the retrieved but falsely retrieved documents, the *false positives*, and (iii) the correct but not retrieved (missed) documents, also called *false negatives*.

The figure 2.1 shows all partitions of possible outcomes that can occur within a quantitative evaluation of results gained from an information retrieval experiment. For the evaluation there are only three partitions to be taken into account. These partitions result from the intersections of the set of retrieved documents and the set of documents that were annotated as correct. One partition is the set of correct documents, that have not been retrieved (*false negatives*). The second partition consists of the correctly retrieved documents and are correct (*true positives*). The third partition subsumes the falsely retrieved documents (*false positives*), the ones which were chosen but have not been annotated as correct.

Recall (equation 2.1) is the ratio between the true positives and all correct documents. Precision (equation 2.3) is the ratio between the true positives and all retrieved documents (true positives and false positives). Note that reducing the set of retrieved documents usually leads to high precision and low recall. Enlarging the set of retrieved documents usually leads to lower precision but higher recall. In order to be able to compare systems reflecting the strong interdependence between these two measures F-score (or F-measure) was introduced (see van Rijsbergen (1979) for details). It is a combination of both measures and it is common to weight recall and precision equally (also called balanced F-score) as is shown in equation 2.4.

$$(2.1) \quad \text{Recall} = \frac{\text{number of correctly retrieved documents}}{\text{number of all correct documents}}$$

(2.2)

$$\text{Precision} = \frac{\text{number of correctly retrieved documents}}{\text{number of all retrieved documents}}$$

(2.3)

$$\text{F-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

I finish this section with a short summary of the advantages and drawbacks of IR as a technique for accessing information from large document collections. The major advantages are: (i) The methods are applicable to large-scale data with satisfactory processing time and thus the scalability of IR systems is given, and, (ii) the described approaches described for building IR systems are domain independent and robust. The main drawbacks of this approaches are: (i) The shallow processing of textual data leads to coarse grained information. The output of IR systems are documents or passages. (ii) The query as well as the document are transformed to a bag of words losing the grammatical relations between them. Queries depending on the grammatical structure (for example subject vs. object of a relation, *e.g.* “who does what to whom?”) cannot be retrieved.

The following section reports on the most important IR conference, the Text REtrieval Conferences (TREC). Besides the TREC there are two other important IR conferences, which are not described here. These are the ACM SIGIR (Special Interest Group on Information Retrieval, which is organised by the ACM<sup>2</sup>) and the ACM CIKM (Conference on Information and Knowledge Management).

### 2.1.3 TREC – the Text REtrieval Conferences

The fundamental idea of the TREC<sup>3</sup> (Text REtrieval Conference) conferences is to provide a common platform for the evaluation of information retrieval systems. TREC is an annual conference taking place regularly since 1992 in the U.S. It is organised by the National Institute of Standards and Technology (NIST). The main contribution is to provide the information retrieval community with an infrastructure for the evaluation of systems that process large-scale (textual) data. This infrastructure has meanwhile grown to be the standard for information retrieval evaluation. The TREC has an annual cycle with a fixed script. It starts with registration of participating groups. In a following step, development collections and sample queries are distributed to the participants. After the participants had time to build their system and adapt it to the given development set, an evaluation set of data is provided to test the developed systems. To ensure that the results come from an automated system two requirements have been stipulated. First, the time for the automatic generation of data is short (on the order of days). Second, the amount of information that has to be retrieved and especially queried is so large that manual querying would not be possible within a reasonable time span. The

---

<sup>2</sup>See [www.acm.org/sigir](http://www.acm.org/sigir) for details.

<sup>3</sup>More information on the TREC conferences is available at: <http://trec.nist.gov/>

Track	'92	'93	'94	'95	'96	'97	'98	'99	'00	'01	'02
Ad Hoc	18	24	26	23	28	31	42	41	-	-	-
Routing	16	25	25	15	16	21	-	-	-	-	-
Interactive	-	-	3	11	2	9	8	7	6	6	6
Spanish	-	-	4	10	7	-	-	-	-	-	-
Confusion	-	-	-	4	5	-	-	-	-	-	-
Database Merg.	-	-	-	3	3	-	-	-	-	-	-
Filtering	-	-	-	4	7	10	12	14	15	19	21
Chinese	-	-	-	-	9	12	-	-	-	-	-
NLP	-	-	-	-	4	2	-	-	-	-	-
Speech	-	-	-	-	-	13	10	10	3	-	-
Cross-Language	-	-	-	-	-	13	9	13	16	10	9
High Precision	-	-	-	-	-	5	4	-	-	-	-
Very Large Corp.	-	-	-	-	-	-	7	6	-	-	-
Query	-	-	-	-	-	-	2	5	6	-	-
Question Answ.	-	-	-	-	-	-	-	20	28	36	34
Web	-	-	-	-	-	-	-	17	23	30	23
Video	-	-	-	-	-	-	-	-	-	12	19
Novelty	-	-	-	-	-	-	-	-	-	-	13
Total participants	22	31	33	36	38	51	56	66	69	87	93

Table 2.1: Overview of TREC tracks from 1992 to 2002 with the number of participating groups for each track. As can be seen from the last row, the number of participating groups is steadily increasing.

last step of the cycle is a conference where the participants present their systems and results.

Traditionally the TREC conferences emphasised text retrieval. Over the years there has been an increasing interest in finding the relevant information regardless of the medium that contains the information. Thus the term *document* is meanwhile interpreted in the context of TREC as textual data, picture data, spoken text etc. This range of tasks and thus the range of assessments related to Information Retrieval is shown in table 2.1. The table gives a brief overview of the tracks carried out from 1992 to 2002 (table and more details can be found in Harman (2003)). As can be seen from the table, from year to year not all tracks are maintained and new tracks are added.

As an example, at the 12th TREC in 2003, (all in all) 93 groups participated in the six tracks. These six tracks are presented here in more detail to provide more contextual information about the IR community. The last of the six tracks, the *Genomics Track* is mostly related to this work. This is why I chose the genomics track to give a more detailed overview.

**HARD Track:** HARD abbreviates **high accuracy tetrieval from documents**. The HARS track took place for the first time within the 12th TREC. The overall goal of this

## 2.1 Information retrieval and passage retrieval

track is to achieve high accuracy retrieval from documents. The granularity of the results is thus the important measure and may be a total document, a passage, or even a sentence. For details see: Allan (2003).

**Participants:** The HARD Track had 14 sites participating.

**Data:** The evaluation corpus was a combination of newswire text from the New York Times, from the Associated Press Worldstream, and the Xinhua English articles.

**Novelty Track:** In contrast to the 2002 Novelty Track, where for an existing TREC topic an ordered list of documents had to be returned that the system judges as relevant and novel, within the 12th TREC fifty new topics were developed specifically for this task. The Novelty task consisted of four sub-tasks: (i) Given the set of 25 relevant documents for the topic, all relevant and novel sentences had to be identified. (ii) Given the relevant sentences in all 25 documents, identify all novel sentences. (iii) Given the relevant and novel sentences in the first 5 documents only, the relevant and novel sentences in the remaining 20 documents had to be found. (iv) Given the relevant sentences from all 25 documents, and the novel sentences from the first 5 documents, the novel sentences in the last 20 documents had to be found. More details on the novelty track are described in Soboro and Harman (2003).

**Participants:** 14 groups participated in the novelty track.

**Data:** Fifty new topics on a collection of three contemporaneous newswires were set up. For each topic, the assessor composed the topic, selected 25 relevant documents by searching the collection, and labelled the relevant and novel sentences in the documents.

**Question Answering:** The question answering track in 2003 contained two tasks, *i.e.* the passages task and the main task. Within the passages task the systems had to return a single text snippet in response to factoid questions; in the evaluation the number of snippets that contained a correct answer were counted. In the main task three separate types of questions, factoid questions, list questions, and definition questions had to be answered. Each of the questions was tagged with a type and the each type was evaluated separately. The details of the question answering track can be seen from Voorhees (2003a).

**Participants:** 11 different groups submitted to the QA track.

**Data:** It was the same document collection used as in the TREC 2002 QA track, *i.e.* the AQUAINT Corpus of English News Text, consisting of the AP newswire from 1998-2000, the New York Times newswire from 1998-2000, and the English Xinhua News Agency from 1996-2000 (all in all around 3 gigabytes of text).

**Robust Retrieval Track** The robust retrieval track aims at improving the consistency of retrieval technology by focusing on poorly performing topics. The task was a traditional ad hoc task, where the topic set consisted of a total of 100 topics. It aimed at investigating the performance of systems that search a static set of documents using previously-unseen topics. For each topic, participants create a query and submit a ranking of the top 1000 documents for that topic. The details are described in Voorhees (2003b).

**Participants:** 16 groups participated in the robust retrieval track.

**Data:** The document collection was from the set of documents on TREC disks 4 and 5 and contained approximately 528,000 documents with 1,904 MB of text.

**Web Track:** The web track consisted of both a non-interactive stream and an interactive stream. The non-interactive stream was centered around two tasks: a topic distillation task and a navigational task. The topic distillation task involves finding relevant homepages, given a broad query. The navigational task is also known as the “home/named page finding task”. Each query involves finding a particular page, which is a homepage in 50% of queries, where the participants did not know which queries were for homepages and which were for non-homepages. For example for the title *cotton industry*, the search task was to construct a resource list for high school students who are interested in cotton industry, from growing, harvesting cotton and turning it into cloth. The query asks for the page by name. The interactive stream (in contrast to the non-interactive) focused on human participation in a topic distillation task over the .GOV collection. More details can be found in Craswell *et al.* (2004).

**Participants:** 2 groups participated in this task.

**Data:** The tasks use the 18 gigabyte .GOV collection, which is available through: <http://es.csiro.au/TRECWeb/>.

**Genomics Track:** The genomics track in 2003 featured two tasks. Both tasks were centered around the Gene Reference into Function (GeneRIF) resource of the National Library of Medicine. The GeneRIF provides annotation of gene function with pointers to the MEDLINE reference for the article that discovered that data.

1. The first task was the *ad hoc document retrieval task* for textual data related to genomics. It was structured very similar to most previous TREC ad hoc tasks (*i.e.* ad hoc task requires a document collection as relevance judgements). The task was to find all MEDLINE references for a gene X that focus on the basic biology of the gene and its product for the designated organism. Basic biology includes isolation, structure, genetics and function of genes or proteins in normal and disease states. 50 sets of genes were each distributed as training and development data. It was recognised that GeneRIFs could serve as pseudorelevance judgments, even though it was suspected that they were

incomplete. The application scenario that motivated this kind of task was a researcher or student that explores a new domain.

2. The second task is related to information extraction, which came in the shape of reproduction of the GeneRIF annotations. This means that the GeneRIF annotation had to be reproduced for a set of genes from MEDLINE records. This task was more exploratory in nature especially because no "gold standard" was provided. The benefit was that the participants compared their methods and results in between each other. A problem in this task was that while some GeneRIF snippets are direct quotations from article abstracts, others are paraphrased. The data for the secondary task consisted of 139 GeneRIFs representing all of the articles appearing in five biological journals.

More details on each of tasks can be obtained through Hersh and Bhupatiraju (2003).

**Participants:** This track attracted 29 groups who participated in one or both tasks. Thus the genomics track was the second largest track of all seven.

**Data:** Both tasks centered around the Gene Reference into Function (GeneRIF) resource of the National Library of Medicine, which was used as both pseudorelevance judgments for ad hoc document retrieval as well as target text for information extraction. At this time more than 7,000 gene entries were available. A subset of the training data consisting of 10 queries were analysed manually for relevance in addition with the top 20 retrieved documents for each query. From these data it could be shown that the data provided by GeneRIF is very accurate for document retrieval, *i.e.* no annotation was wrong. At the same time it was shown that the data was highly incomplete, *i.e.* more than 40% of the retrieved data were relevant for the given topic but not annotated in GeneRIF. As the organisers pointed out, the genomics track was "constrained by lack of resources but partially overcome by great enthusiasm". For the relevance judgement a set of 525,938 MEDLINE records were provided.

**Results:** For the evaluation of the first task the *Mean Average Precision* (MAP) measure was used, which combines recall- and precision-oriented aspects. In a first step the average precision for each query is calculated. Then the MAP is computed over all queries. For the first task the best group finished with a MAP of 0.4165 and the worst with a MAP of 0.0271. The mean over all groups was 0.2313. The evaluation of the second task was difficult since it had to be judged whether the automatically generated variant was the same as the ones in the evaluation set and no straightforward mechanism of normalisation was available. Thus different types of the DICE coefficient were applied<sup>4</sup>. The most striking result was that just taking the title of a Medline

---

<sup>4</sup>The DICE coefficient  $Dice(A, B) = (2 * Z) / (X + Y)$  is defined with  $X$  representing the number of words in  $A$ ,  $Y$  as the number of words in  $B$ , and  $Z$  as the number of words occurring in both  $A$  and  $B$ . It is a measure to express the overlap between two descriptions.

abstract as GeneRIF annotation, 2/3 of the systems could be outperformed.

### 2.1.4 Summary information retrieval

---

Methods	Document indexing in combination with statistical methods (Bag-of-words approach)
Query	Any set of words is treated as bag of words
Domain	Mostly Un-restricted
Result	Set of Documents or Passages
Interpretation of Results	The user interprets the retrieved documents
Evaluation	Clear: Precision and recall can be measured

---

Table 2.2: Summary: Information and Passage retrieval

## 2.2 Information extraction

The following gives a short overview of information extraction (IE). It is kept short to be comparable to the other presented techniques like *information retrieval*, *question answering* or *text understanding*. More details are presented through examples in the section 2.2.1 on the Message Understanding Conferences.

IE is generally understood as a method that extracts information bits like entities and relations from text collections. The information bits are used for filling of pre-defined templates that encode the extracted information. Since this kind of outcome is highly structured it is aimed to be used for populating databases or for spread sheet analysis. This process is depicted in figure 2.2. What the picture shows in addition is that information pieces, which are identified in text collections get copied into predefined template without preserving the order of the information bits in the text, which is indicated by the crossing edges of the arcs.

In general two approaches have been pursued within the NLP community, where one is based on hand-crafted rules and the other can be described as a machine learning approach for learning these rules. Although the development of specific rules is time-consuming it has two major advantages. First, hand-crafted rules are highly specific. They are manually adapted to exactly fit the desired templates. Usually, this approach leads to high quality results. The second advantage is that the principles that have been used when writing the rules for extraction are explicit and transparent so that they offer the possibility for fine tuning or adaption to related issues in a later stage.

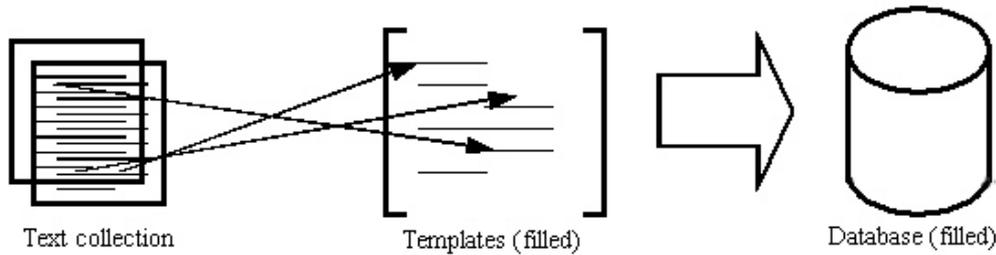


Figure 2.2: **The general information extraction scenario:** The picture shows that information pieces that are identified in a text collection are used to fill predefined templates, where the ordering of the information in the text need not be preserved. The structured information from the templates can be used to populate databases.

The machine learning approach for deriving the rules has the ostensible advantage of being easy to adapt since the learning procedure is carried out in an automatic fashion. On the other hand one has to be aware that this kind of approach is only applicable if there exists a manually annotated corpus to train the system on. And, creating manually annotated corpora is a highly time consuming task. This is especially so for highly specialised domains like the life sciences where much domain knowledge is both needed to develop a relevant and consistent annotation schema as well as for the annotators to apply this schema. In contrast to a hand-crafted approach it is not possible to shift to a related topic without a newly annotated corpus. In the case of hand-crafted rules it is often possible to adapt the rules to related issues. Since annotated corpora are just being established in the life science domains I will neglect the machine learning approach in the following.

The common rule-based approach is described in Hobbs (2003). I describe here a version with some extensions. Generally all architectures are based on a preprocessing step. Usually this step is followed by a cascade of finite state transducers. This gives a cascaded set of levels that are arranged into a common architecture:

1. **Preprocessing of the documents.** This step comprises recognising sentence and word boundaries as well as the assignment of a part-of-speech tag for each word. Depending on the further processing, stemming<sup>5</sup> is also carried out.
2. **Detection of complex words** like multiwords and proper names. Multiwords in common English text – for example newspaper text – are words like *anno domini*, *grown ups*, or *in terms of*. Usually their internal or compositional structure is not of interest. In the domain of biomedicine, examples of multiwords can be *Saccharomyces oviformis*, *binding site*, or *upstream activating factor*. This step can

<sup>5</sup>Reducing words to their root is usually called stemming or lemmatising. In the stemming process words are stripped down to their roots by truncation. This allows for mapping of linguistic variants of one and the same word to its root. Thus *activates*, *activated*, and *activate* would all get stemmed as *activate*.

be carried out efficiently by combining terminological dictionaries and finite-state techniques.

3. **Detection of basic phrases** like noun groups (*i.e.* nouns with their left modifiers) and verb groups (*i.e.* auxiliaries and head verb) are carried out in this processing step. As an example noun groups can be marked as  $[_{nx} \dots]$  and verb groups can be marked as  $[_{vx} \dots]$ . In (2.4) this is shown for a common English sentence, where in (2.5) this is shown for a biomedical example.

(2.4)  $[_{nx} \text{ I}]$  saw  $[_{nx} \text{ two dogs}]$  that  $[_{vx} \text{ were running}]$  in  $[_{nx} \text{ the park}]$ .

(2.5)  $[_{nx} \text{ The ATR1 promoter region}]$  contains  $[_{nx} \text{ a binding site}]$  for  $[_{nx} \text{ the GCN4 activator protein}]$ .

4. **Complex phrases like** complex noun groups are identified. This means that nested structures are recognised. As an example a noun group that is embedded through a preposition ought to be recognised within this processing step.

(2.6)  $[_{nx} \text{ The ATR1 promoter region}]$  contains  $[_{ng} [_{nx} \text{ a binding site}]]$  for  $[_{nx} \text{ the GCN4 activator protein}]$ .

5. The structures detected in the previous levels are **transformed into domain patterns**. This means the syntactic structures are mapped to semantic representations for the information in question. In some cases information can only be partially recognised, *i.e.* the templates cannot be filled completely. If this is the case the following step is necessary.
6. **Merging of the partial results** from the previous levels. If several partially filled templates exist for a paragraph or a sequence of sentences. These patterns can sometimes be merged. It should be noted that this step is error-prone, since semantic criteria are the only elusive, which frequently are not available.

To give a short overview of the advances and limitations in IE within the last years, the message understanding conferences (MUC) that have been carried out between 1987 and 1997 are the most popular reference. In the following sections I present an overview of the tasks carried out within these conferences and the results that were achieved by the participating systems.

### 2.2.1 MUC – The message understanding conferences

The message understanding conferences (sometimes also referred to as *message understanding competitions*<sup>6</sup>) were carried out by DARPA (Defense Advanced Research Project Agency) between 1987 and 1997. Regarding the rapid growth of on-line texts, the main motivation was to develop methods for formal evaluation of information extraction

---

<sup>6</sup>The first two conferences were called Multilingual Entity Task (MET) since they were mainly interested in entity recognition for multilingual information extraction systems.

tasks. As a consequence these methods developed as standard benchmarks for comparing different IE systems. The organisation allowed only for a short system preparation time to stimulate portability to new domains and scalability of the systems. The focus on a certain kind of task for each conference was key for establishing Information Extraction as an applicable technology in the field of computational linguistics.

### Overview of tasks evaluated in MUC-3 through MUC-7

#### Named entity (NE) task – Identification of entities of predefined type

Within the NE task the goal is to identify each string that represents a named entity, or more general specific types of nouns or noun phrases according to a given taxonomy. Named entities within the message understanding competitions were divided into three taxonomical categories, which are (i) entity name expressions (annotated as ENAMEX), consisting of person, organisation, location name, or (ii) time expressions (TIMEX), comprising date and time expressions, as well as (iii) number expressions (NUMEX), comprising monetary and percentage expressions. In comparison to the other MUC tasks the NE task proves to have the most reliable results. This is reflected by the F-score rates of up to 95%. One should consider that human annotators reach an agreement of 97% according to Robinson *et al.* (1999).

**NE difficulties** – There are several reasons that make the NE task a challenge. I will mention here three important ones. First, there are too many different NE to be simply listed, *i.e.* language is productive and infinitely new named entities can be created. Second, the naming is changing constantly (*i.e.* new names occur, other names in use change their meaning by-and-by and thus refer to other entities) A third reason is the variance among the names to refer to the same entity. As an example a person can be referred through his first name, through his surname, through his nickname, through his position in an organisation, through a combination of the given names, through an abbreviated version etc. Several other reasons could easily be named. A more detailed overview for the biomedical domain is given later in section 3.1 *Naming in Biology*.

**NE example** – A MUC-7 example showing the annotations for named entities referring to *locations*, *date expressions*, *person names* and *organisation names* marked with SGML tags:

```
<ENAMEX TYPE="LOCATION">Italy</ENAMEX>'s business
world was rocked by the announcement <TIMEX TYPE="DATE">last
Thursday</TIMEX> that Mr. <ENAMEX TYPE="PERSON">Verdi</ENAMEX>
would leave his job as vice-president of <ENAMEX
TYPE="ORGANIZATION">Music Masters of Milan, Inc</ENAMEX> to become
operations director of <ENAMEX TYPE="ORGANIZATION">Arthur
Andersen</ENAMEX>.
```

Another entity that also belongs to this class is the class of temporal expressions, comprising date expressions and expressions referring to times of the day. Similarly, number expressions like monetary expressions or percentages also belong to the class of named entities.

**Template element (TE) task** – Identification of attributes to the NE

The TE task deals with the extraction of information or attributes pertaining to instances of the previously detected entities like organisations, person, and artifact entities, drawing evidence from everywhere in the text. All in all the TE results reached up to 80% F-score. In comparison to these automatically derived results humans reach roughly 93%. This shows that the automatic systems reach a reasonable and useful approximation.

**TE difficulties** – Sometimes the distinction between attributes and template relation is not sharp. Usually the relevant attributes are pre-defined in the task. Like for NE systems, the TE systems have to be adapted when porting it to a new domain.

**TE example** – For the above given example one would expect that the attributes of *being vice-president* and of *being operations director* for the NE *Verdi* are recognised.

**Template relation (TR) task** – Identification of facts

The extraction of (binary) relational information between the previously identified entities (or entity templates) is called TR task. The template relation task reached 76% F-score within the MUC-7.

**TR difficulties** – As mentioned above, the boundary between Template element (TE) and Template Relation (TR) is not always sharp.

**TR example** – In the case of the previously introduced NE example one possible template relation could be the leaving of *Mr. Verdi* from *Music Masters of Milan, Inc.* Some other general example relations are *employee-of(x,y)*, *manufacturer-of(x,y)*, *location-of(x,y)* etc.

**Scenario template (ST) task** – Identification of event information

The ST task deals with extraction of a pre-specified type of event information. It thus relates the event information to particular entities that were identified before. An example for the ST task can be management succession, where information about the company, the previous manager and the succeeding manager is gathered. The ST task exceeds sentential boundaries. The highest Scenario Template performance was 47% recall and 70% precision within the MUC-6.

**ST difficulties** – The main difficulty in the ST task is that the relevant information is scattered over several phrases and sentences. Thus a step for merging the information pieces (mostly being recognised in the preceding steps) has to be carried out. This mostly relies on semantic criteria and frequently makes

it necessary to understand the text (details on this issue are described in the section *Text Understanding and DUC's 2.4*).

**ST example** – Following the NE example from the beginning, the desired outcome would be to keep track of the complete management change at *Music Masters of Milan, Inc.* This comprises not only that *Mr. Verdi* has left this position. But within the context a successor must be identified.

**Co-reference (CO) task** – Identification of co-referring entities

The identification of co-referring expressions in the text has been evaluated within the co-reference task. This means that all mentions referring to one and the same entity, including those marked in NE and TE had to be identified and linked. It has to be mentioned that only nouns had to be linked – relations involving verbs have been ignored. The ability to achieve good results, which is indicated by interannotator agreement 95%, shows that co-reference is well definable.

**CO difficulties** – This task exceeds sentential boundaries and thus long-distance dependencies within the text have to be identified. The difficulties arise from both the amount of possible co-referents as well as the semantic criteria that are responsible for identifying these co-referents.

**CO example** – The following shows an example with correct SGML annotation where *Haden MacLellan PLC of Surrey, England* and *Haden MacLellan* had to be recognised as referring to the same entity. The dots indicate that some words, phrases or even sentences between the co-referring expressions have been omitted.

```
<COREF ID="100" MIN="Haden MacLellan PLC">Haden MacLellan PLC of
Surrey,
England</COREF>
... <COREF ID="101" TYPE="IDENT" REF="100">Haden MacLellan</COREF>
```

### Evaluation in the MUC tasks

Evaluation of information extraction experiments was carried out according to a scoring system, where the number of correctly filled slots is related to the number of slots that were filled at all. Similar to the evaluation in information retrieval (especially TREC), recall and precision (and thus F-score) were regarded to judge the quality of the extracted results. A short overview of the MUC domains, and relevant evaluated tasks with the achieved results are shown here.

**MUC/MET 1 Year: 1987**

**Domain:** Messages about naval operations

**Evaluated tasks and Results:** Named entity task in Chinese (F-score: < 85%), in Japanese (F-score: < 93%), and in Spanish (F-score < 94%) have been carried out.

**MUC/MET 2 Year: 1989**

**Domain:** Messages about naval operations

**Evaluated tasks and Results:** The named entity tasks were carried out in Chinese (F-score: < 91%) and in Japanese (F-score: < 87%).

**MUC 3 Year: 1991**

**Domain:** News articles about terrorist activity

**Evaluated tasks and Results:** The scenario template task in English reached a precision of < 50% and a recall of < 70%.

**MUC 4 Year: 1992**

**Domain:** News articles about terrorist activity

**Evaluated tasks and Results:** The scenario Template task in English reached a F-score of < 56%.

**MUC 5 Year: 1993**

**Domain:** News articles about joint ventures (JV) and microelectronics (ME).

**Evaluated tasks and Results:** The ST task in English for JV with a F-score of < 53% and for ME with a F-score of < 50% as well as in Japanese for JV with a F-score of < 64% and for ME with a F-score of < 57% have been carried out.

**Participants:** 23 groups participated in MUC 5.

**MUC 6 Year: 1995**

**Domain:** News articles about management changes.

**Data:** The experiments were conducted using Wall Street Journal texts (provided by the Linguistic Data Consortium, LDC).

**Evaluated tasks:** Named entity task in English reached an F-score of < 97%, the Coreference Task in English reached a precision of < 63% and a recall of < 72%. The Template Element task performed in English with a F-score of < 80%, and the Scenario Template task in English performed with a F-score of < 57%.

**Participants:** All in all 16 groups participated in various tasks.

**MUC 7 Year: 1997**

**Domain:** News articles about space vehicle and missile launches

**Data:** The MUC 7 was carried out on *New York Times News Service*, which was supplied by the Linguistic Data Consortium with approximately 158,000 articles.

**Evaluated tasks** Named entity task in English (F-score: < 94%), Coreference Task in English (F-score: < 62%), Template Element task in English (F-score: < 87%), Template Relation task in English (F-score: < 76%), Scenario Template in English (F-score: < 51%).

**Participants:** All in all 18 groups participated in MUC 7.

### 2.2.2 Summary information extraction

Methods	Linguistic Methods like chunking semantic interpretation (i.e. template filling)
Query	Predefined Questions = Templates f.ex. <i>Who did what to whom?</i>
Domain	Restricted
Result	Filled Templates and Facts
Interpretation of Results	The IE-System interprets the Result
Evaluation	Clear: Precision and recall can be measured

Table 2.3: Summary: Information Extraction

## 2.3 Question Answering

In contrast to information retrieval systems, question answering (QA) systems return an actual answer, rather than a ranked list of documents, in response to a question. The question need not be restricted to a domain. It is written in natural language and the answer is obtained from a (*large*) document collection. The TREC conferences have had a question answering track since 1999; in each track the task has been defined such that the systems were to retrieve small snippets of text that contained an answer for open-domain, closed-class questions (*i.e.* fact-based, short-answer questions that can be drawn from any domain). Within TREC-8<sup>7</sup>, for example, the participants received a text collection and 200 fact-based questions, like “How many calories are there in a Big Mac?”. The participants had to return a ranked list of five answers per question. Two run types were offered. One, where 50 bytes (characters) was the maximum length, the other, where 250 bytes was the limitation. Each string was read by a human assessor who made a binary decision as to whether or not the answer is correct. The reciprocal of the rank was counted as the achieved score.

It is becoming clearer and clearer that most questions in a realistic application scenario are not simply what is known from television quiz shows (*e.g.* When was television

<sup>7</sup>The annual Text REtrieval Conference (TREC) carries out a question answering track since 1999. For a large number of newspaper articles, the participating groups try to answer a given set of questions fully automated.

invented?), but rather questions where the asker seeks a brief summary of facts related to the question. As an example, a question such as *Who was Miles Davis?* cannot be answered by a two words answer. It requires a number of facts ranging from the fact that he was an American citizen, that he was born on 26th of May, 1926 in Alton, Illinois/USA and died on 28th of September 1991 in Santa Monica, California/USA, that he was one of the most influential jazz trumpet players and jazz composers, etc. This information can easily be distributed across multiple documents, many of them repeat the same facts in various ways. Hence, QA quite naturally is related to multi-document summarisation. This shows already that QA cannot be regarded isolated like IR or IE but is connected to and depends on several other techniques.

At present, several textual question answering systems are described in the literature. As an example, in 2002 the TREC conference 34 groups participated in the question answering track, with each group using their own system. Nevertheless, a series of features can be identified within most systems that constitute a kind of general architecture. According to Monz (2003) this architecture can be described by four components: (i) *question analysis and classification*, (ii) *document retrieval*, (iii) *analysis of the retrieved documents*, and (iv) *selection of answers*, which mainly can be described as follows:

- (i) The main purpose of the question analysis and classification step is to determine which kind of answer is appropriate for this (type) of question. It is common to do this by examination of the question words. For example *who is/are* refers to a person or a group of persons, where questions like *how many* expect a number as answer.
- (ii) Document retrieval techniques are used by almost all question answering systems to identify documents that are likely to contain a relevant answer to the posed question. This step is frequently referred to as *pre-Fetching*. The key approaches for *document retrieval* are already introduced in section 2.1.
- (iii) The document analysis step is a non-trivial process where the retrieved documents are analysed in depth for the sentences (and sometimes paragraphs) that can be linked to the question. This usually comprises a deeper analysis of the restricted set of retrieved documents. Deeper analysis means application of methods like *syntactic parsing*, *synonym detection/linking*, *pattern matching* to find the entities that are questioned in the first step.
- (iv) In the last step the answer selection takes place. The best candidate from step (iii) has to be chosen for each question. Various criteria can be taken into account for this step like syntactic constraints in the question-answer pairs, or frequencies of retrieved results etc. An appropriate (set of) answer(s) is thus retrieved and presented to the user as a response.

Methods	Mixture of IR and IE methods
Query	Natural language questions
Domain	Mostly Un-restricted
Result	Short answer in natural language
Interpretation of Results	The QA system interprets the results
Evaluation	Rather clear: Precision and recall can be measured in the majority of cases.

Table 2.4: Summary: Question Answering

### 2.3.1 Summary question answering

## 2.4 Text understanding and DUC's

Human text understanding can be seen as taking a certain perspective towards a text (Franzén *et al.* (2002)). Taking the perspective of seeking for special information in a newspaper article, this perspective depends on the background the information seeker has, his current information need, his attitude, his temporal and physical constraints etc. To give an example, in a specialised domain, like the biomedical domain a biologist (*i.e.* a domain expert) has a different perspective towards a text on protein interactions than a layman has. Thus from differences in ability or differences in demand differences in reception result. It is pretty obvious that having so high demands – to be able to model each facet of a text meaning – on an automatic system is not feasible and thus can only be disappointing. It should be noted that *text understanding* in general has no predetermined specification of the semantics or communicative aspects. The representation formalism should in principle be rich enough to capture all aspects of meaning in a text. Since there is no clear criterion of what the result should look like it is hard to judge what a successful system should look like. The approximation that is been done in the field of automated text understanding is to define specific perspectives on a text like the specific example of *information extraction* in the previous section. Another perspective is *text summarisation*, which can also be seen as way of understanding text. In this context the Document Understanding Conferences have been established in the last year. I will give a rough overview over the tasks that are evaluated there as a kind of representative for *text understanding tasks*.

### 2.4.1 Document Understanding Conferences (DUC)

Over the last years an increasing interest in summarisation research can be witnessed. A DARPA (Defense Advanced Research Project Agency) program, the *Translingual Information Detection, Extraction, and Summarisation* (TIDES), has been initiated with calls for major advances in summarisation technology. As part of this program the National Institute of Standards and Technology (NIST) began an evaluation series in the area of text summarisation tentatively called the Document Understanding Conferences (DUC). Although more detailed information on the DUC conferences can be obtained through: <http://duc.nist.gov/> it should be noted that some information is retained for registered users only.

Between 2001 and 2004 a growing number of research groups participated in the evaluation of generic and focused summaries of English newspaper and newswire data. Various summary sizes – between 10 and 400 words – have been requested for both single-document summaries as well as summaries of multiple documents – between 30 and 60 sets of around 10 documents per set – where the resulting summaries have been evaluated. Both manual as well as automatic evaluations have been explored.

Automatic text summarisation can be seen as consisting of three major steps. In a first step, the *analysis step*, a content representation of the source text is constructed. In a second step, the *transformation step*, the content representation is mapped into a summary representation a so-called *condensate* or a textual summary (extract) by way of selection, compaction or condensation. And in a last step, the *synthesis step* an appropriate output format is generated for the summary representation of the condensate where mainly natural language is used.

### 2.4.2 Summary Text understanding

Methods	Deep linguistic processing and artificial intelligence methods
Query	Any concerning the text(s)
Domain	Restricted
Result	Interpreted text(s)
Interpretation of Results	The TU system interprets the text
Evaluation	Not clear at all

Table 2.5: Summary: Text Understanding

## Chapter 3

# Language and Language Technology in Biology

The preceding chapter introduces the standard techniques for retrieving and extracting information from large-scale textual data. These techniques have been introduced on a general background mainly independent of a certain domain. The two following sections introduce domain specific biomedical text processing topics that have to be taken into account in such given information retrieval or information extraction scenarios. The first section deals with terminological issues within the biomedical domain, it mainly deals with synonymy, homonymy, polysemy and other types of variations in naming biomedical entities. The second section describes related work in the field of biology related information extraction, it mainly describes details of the *BioCreAtIvE* (*Critical Assessment of Information Extraction Systems in Biology*) competition 2004, which was designed as a kind of biomedical MUC.

### 3.1 Biomedical Terminology

To extract relations between entities from textual data, the names referring to these entities and their occurrences within the textual documents have to be recognised first. In Hobbs (2003) the author points out that technical terminology in the biomedical domain presents the main challenge of applying information extraction techniques. What makes it hard to deal with (this type of) terminology is that many forms of variation are in use. By variation variations in term formation are meant. As this process is highly productive it allows the generation of several variants, this refers to syntactic variations. On the other hand semantic ambiguity is an issue as well. The biomedical literature is rich in examples where both one term refers to many entities and several names refer to one and the same entity. Natural language offers a variety of ways for expressing one and the same thing in several syntactic different ways. This is a well-known problem for terminological parsing. There are several surveys reporting on these variations for biochemical terminology to be found in literature. Each has its own advantages and shortcomings. In what follows I try to give a classification and an overview of common

terminological issues in this domain, which overcomes these shortcomings and extends especially Collier *et al.* (2002) and Nenadić *et al.* (2003). The new items originate from own experiences within the conducted IE experiments. These will be described in the Chapter 6. In what follows I introduce three classes of terminological issues with a series of examples. These are (i) *Synonymy, homonymy and polysemy*, (ii) *Variations in naming*, and (iii) *Coordinations and appositions*.

### 3.1.1 Synonymy, homonymy and polysemy

**Synonymy** describes the fact that different names refer to the same entity or object. The challenge for synonymy identification consists of two issues. First, text spans referring to entities have to be identified. In a second step the terms referring to the same entities have to be collected and annotated as such. It is important to mention that the detection of synonymic variants has immediate consequences for the recall numbers of an IE system. If the system can identify more terms as synonyms, it can thus extract more facts about this entity. As a consequence the recall number of the system increases. In contrast, if the synonymic variants of an entity cannot be identified the reported facts might be regarded as useless, negatively affecting the recall of the system.

The succeeding items list gives some relevant examples of biomedical subdomains where synonymic variation is a challenging issue:

1. The naming of chemical compounds and macromolecules can in principle be done according to the existing IUPAC (International Glossary of Pure and Applied Chemistry) nomenclature (see IUPAC (1993) and IUPAC (1979) for details). Nonetheless, two major problems exist for the automatical identification of names of chemical compounds and macromolecules in text. Although IUPAC regulates the naming of chemical compounds it is not trivial at all to detect and classify the names in literature. As an example, the fact that trivial names, which abbreviate longer systematic or recommended names, occur as parts in the formation of new names, which makes it necessary to maintain a dictionary that maps trivial names to the full systematic names. At the moment, such a dictionary is not available. In addition several abbreviations for chemical compounds are in use. Sometimes these are invented by the authors. As a consequence there's not only one way to name a chemical compound. Several names for one and the same chemical compound are a consequence. Unfortunately some are according to the official nomenclatures, which are not restricted enough with respect to issues of synonymy. The combination of official and unofficial abbreviations, trivial names and systematic names offer several possibilities for naming.

To give an example, the KEGG database (see Kanehisa and Goto (2000) for details ) (on April 19th 2004) had three different entries for one and the same chemical compound: *i.e.* **NTP**, **Nucleotide Triphosphate**, and **Ribonucleoside Triphosphate**, where all refer to the same chemical compound. The first of these three names abbreviates the second name. The second has as first part the word *nucleotide*, which in principle has one phosphate group as a substantial part. By

adding *triphosphate* this should result in a nucleotide with four phosphates (which is surely not intended). An alternative – and this is what is intended – would be to regard this as a substitution, where the *mono*-phosphate is replaced by a *tri*-phosphate. The third name is *ribonucleoside*, which can be described as a *nucleotide* lacking a phosphate group. The naming thus reflects that adding the *triphosphate* to the *ribonucleoside* would yield a nucleotide with three phosphate groups attached. This example shows that synonymic variants can derive from different intended implicit semantics.

2. Regarding the naming of *enzymes* the situation is even worse. The enzyme *alcohol dehydrogenase* (E.C. 1.1.1.1) for example comes with 26 official synonyms (following the BRENDA database<sup>1</sup>). Since these names reflect the chemical properties (i.e. the chemical reactions they catalyse), these names are often long and subject to multiple types of variations, although the names are gathered and maintained to avoid these variations (in BRENDA). It should be noted that the list is not complete and steadily new names occur that are added to the database.
3. One could think that in principle the amount of existing *organisms* is stable and thus the list of names could be easily maintained. In fact there are at the moment at least  $1,7 \times 10^6$  species of living organisms that have been discovered and the list is steadily growing. Luckily the amount of organisms that is considered commonly in molecular research projects is very limited. According to the NCBI taxonomy (see Wheeler *et al.* (2000)) these are 21 organisms. Nevertheless, problems of synonymic variance are common, at least for some species. Some of these standard organisms do not only have the standard Latin names (e.g. *Mus musculus*) and a common English name (e.g. *house mouse*) but also several types of subspecies (e.g. *Mus musculus bactrianus*) with their common English variant (e.g. *southwestern Asian house mouse*). Further combinations of the Latin names and the common English names (e.g. *Mus musculus domesticus mouse*) occur in several cases as well. Another source of variation results from gene mutation. This means that if a certain species is genetically mutated a naming variant is generated from its Latin or common name plus some combination of letters and numbers (e.g. *C57BL/6N mouse*, *Mus lymphomagenic MCF13*) indicating the gene locus of the mutation. In contrast to the mentioned examples names for the human species are very limited (i.e. *human*, *man*, *homo sapiens*).
4. Amongst all synonymic variants protein and gene names are perhaps the most popular example of biology-related synonyms. Different names for one and the same protein and/or gene is very frequent although there exist guidelines for gene nomenclature. For example the Human Gene Nomenclature Committee (HGNC), part of the Human Genome Organisation (HUGO), has released guidelines<sup>2</sup> and

---

<sup>1</sup>BRENDA is a collection of enzyme names with functional data. It is available free of charge for academic, non-profit users via <http://www.brenda.uni-koeln.de>. Details are described in Schomburg *et al.* (2002).

<sup>2</sup>To given an example of the guidelines, here some principles taken from the <http://www.gene.ucl>.

has developed a nomenclature database (see Wain *et al.* (2004) and Wain *et al.* (2002) for details). The practice shows that researchers rarely follow these guidelines.

The naming conventions for non-human genes is even worse as Weeber *et al.* (2003) reports. The naming process in molecular biology is creative (*i.e.* highly productive). One possible explanation is that this is due to different researchers detecting one and the same protein or gene in different contexts. Later, after the work on each of these proteins or genes has been published it turns out that both refer to the same gene or protein. Sometimes this identification leads to a third name. As an example **BYP1**, **CIF1**, **FDP1**, **GG51**, **GLC6**, **TPS1**, **TSS1**, and **YBR126C** are all synonyms for the same gene/protein in yeast. Note that the names of proteins or genes can as well consist of names that refer to the function of the proteins, *e.g.* **alcohol dehydrogenase**, **Alcohol dehydrogenase I**, **1-aminocyclopropane-1-carboxylate synthase CMW33**, or **S-adenosyl-L-methionine methylthioadenosine-lyase**. These examples illustrate that it is highly dependent on the intentions and contexts a researcher has in mind when discovering a new protein or gene is discovered.

5. In scientific publications abbreviations are obviously a popular means to reduce tedious repetition of spacious technical terms. From a named entity recognition point of view abbreviations can be regarded as synonymic variants since one and the same entity is referred to by different names. At the same time abbreviations bring about the problem of homonymy, since one and the same abbreviation can refer to different entities.

An example for synonymic abbreviations is *TCR*. It can abbreviate *T-cell antigen receptor*, *T-cell receptor protein* or *Tetracycline resistance protein*, all being proteins. It is an especially hard nut to crack when abbreviations are used within larger technical terms. Example 3.1 shows that *IL-2* abbreviates only *interleukin-2*. But, the complete technical term to be recognised would be *interleukin-2 promoter DNA*. The whole term has an abbreviation as part of it. In case the reader wants to find out what the whole expression refers to, he has to expand *IL-2* correctly to *interleukin-2*.

(3.1)The interleukin-2 (IL-2) promoter DNA consists of several independent T-cell receptor (TcR) responsive elements.

Generally, abbreviations can be classified and/or generated according to at least the following mentioned ways:

- (a) Acronyms are a subtype of abbreviations that are simply generated by taking the first letter of each word from the full name. Examples are *CDK* for *cyclin dependent kinase* (see example 3.2) and *REP* for *Rab escort protein* as shown

---

[ac.uk/nomenclature/guidelines.html](http://ac.uk/nomenclature/guidelines.html): (i) The initial character of the symbol should always be a letter. The subsequent characters may be other letters, or if necessary, Arabic numerals. (ii) All characters of the symbol should be written on the same line; no superscripts or subscripts may be used.

in example 3.3. In the latter case the abbreviation is just used for a part of the whole term *Rab escort protein 1* and thus occurs within the term.

(3.2)The *Saccharomyces cerevisiae* gene KIN28 is a member of the cyclin dependent kinase (CDK) family.

(3.3)Rab escort proteins (REP) 1 and 2 are closely related mammalian proteins required for prenylation of newly synthesized Rab GTPases by the cytosolic heterodimeric Rab geranylgeranyl transferase II complex (RabGG transferase).

- (b) Another possibility to form abbreviations is by mixing of capital or numerical characters from the full name. Example (3.4) shows that *IL2-R* abbreviates *interleukin-2 receptor*.

(3.4)The largest experience exists for the MoAb's against the interleukin-2 receptor (IL2-R);

- (c) Instead of using the each initial character of the words, the initial characters of the syllables of the full name can be used as well. One variant is to keep the order of appearance. This is shown in example (5c).

(3.5)Human ICAM-3 (InterCellular Adhesion Molecule-3), also known as CD50, is a 120 kDa, type I transmembrane glycoprotein that appears to be (normally) expressed only on select hematopoietic cell types.

- (d) For matters of better readability or associativity a series of initial character of a syllable from the full name is used. *Tre6P* in example 3.6 thus abbreviates *trehalose 6 phosphate*.

(3.6)Synthesis of trehalose in the yeast *Saccharomyces cerevisiae* is catalysed by the trehalose 6 phosphate (Tre6P) synthase phosphatase complex , ...

- (e) By taking initial characters of the syllable and substituting the original order another type of abbreviation can be generated. In the example 3.7 additionally the arabic numeral "5" has been taken instead of the roman numeral "V".

(3.7)One gene for subunit V of cytochrome oxidase (COX5b) has also been shown to contain an intron.

- (f) Two further possibilities are shown within the example (3.8). One possibility is by taking any characters from the full name not necessary being the initial character of a syllable or word. A second possibility is also by adding variations in capitalisation, like *cTPx II* in example 3.8.

(3.8)We observed that the transcription of *Saccharomyces cerevisiae* cytoplasmic thiol peroxidase type II (cTPx II) (YDR453C) is regulated in response to various stresses (e.g. oxidative stress, carbon starvation, and heat shock).

- (g) Several other types of abbreviations in use are generated by combinations of the given principles. In addition, two things should be noted. First, that

there's no 1:1 correspondance between abbreviations and full names. For a given term several abbreviational variants exist and are in use. For example to refer to *nuclear factor kappa B*, the following possible abbreviational variants can be identified:

NF(kappa)B, kappaB, NFkB factor, NF-KB, NF kB, NF kappa B, NF-kappaB

At the same time abbreviations can be ambiguous. One and the same abbreviation can abbreviate two different terms: *e.g. EGFR* can abbreviate *epidermal growth factor receptor* as well as *estimated glomerular filtration rate*. This means that it is of limited use to maintain dictionaries of abbreviations and corresponding expressions.

Second, as already mentioned, abbreviations can occur as part of larger names and thus be part of nested constructions. This makes the detection and semantic categorisation of technical terms challenging.

**Homonymy** and **polysemy** both refer to cross-over of terms between semantic classes. This means that one name can refer to different entities and thus to different concepts. Or, to put it in other words that one name can have different meanings. While on the one side the term polysemy is used for one and the same expression having different but related meanings (*e.g. the word cherry* which is either a sweet red berry with pip or the tree bearing this berry) homonymy is used for words with entirely different meanings (*e.g. the English word bear* thus can either refer to an animal or to the verb meaning to carry something). It should be noted that for the extraction of relations from a text both homonyms or polysems that remain unrecognised affect the precision number of such a system (*i.e. a wrongly recognised categorised entity will produce a wrong result*). This is in contrast to synonymic variations which effect the recall number of an information extraction system (*i.e. if one synonym is not recognised it will not be extracted*). A few biology related examples for both polysemy and homonymy are presented below.

Since a gene and its gene product (*i.e. an expressed protein or stable RNA*) are frequently named by one and the same term this is a frequent source of polysemy. It appears that over 50% of all gene and protein names in PubMed abstracts occur without a head noun specifying the semantic type of the entity. Although some homonyms can be disambiguated through the POS-tags (*i.e. protein or gene names can only be tagged as nouns*), most of them remain ambiguous. Sometimes these names can be disambiguated by the selectional restrictions of the verb. But, in a series of cases the names still remain ambiguous. Disambiguation is then very complex, because more context has to be taken into account.

Depending on the organisms the naming of gene names is more or less standardised. An example organism for which it is very challenging to detect related gene names is the fruit fly *Drosophila*. Scientists working on this organism seem to enjoy applying gene names with primary meaning outside the biological domain. Examples like *boss*, *disco*, *eve*, *gypsy*, *vamp*, *zip* or *ogre* that have a meaning in common English as well show the problems for recognising these words as gene names.

An example for homonymy where the determination of the meaning highly depends on

the context is the word *insulin*. It can, for instance, refer to a gene, to a protein, to a hormone or to a therapeutic agent.

A problem that concerns both synonymic and homonymic types of naming is that nearly all classes of names are **open and steadily growing**. To give an idea of the growth I show some numbers on the amount of entries in NCBI Reference Sequence (RefSeq) collection within the last year<sup>3</sup>. It contained:

929,473 entries in the latest release (Mar 24, 2004),

844,408 entries in the Jan 14, 2003 release,

831,287 entries in the Oct 21, 2003 release, and

785,143 entries in the release of Jun 30, 2003.

The figure 3.1<sup>4</sup> shows the exponential growth of identified base pairs for DNA sequences over 20 years (i.e. 1982 - 2002). GenBank is a genetic sequence database, an annotated collection of all publicly available DNA sequences. This reflects the dramatic increase in identified genes and thus the increase in naming them. Of course this affects the corresponding gene products (mainly proteins).

#### 3.1.2 Variations in naming

Two other types of variations that are concerned with the spelling of terms are orthographic and paragrammatic variations. While orthographic variation refers to variants in spelling that are all grammatical the paragrammatic variations are not grammatical and usually subsume typing errors. In addition to the spelling variants two other types of variations are syntactical and semantic variations. While syntactic variation refers to different syntactic variants for the same terms, the semantic variation refers to the interpretation of these syntactic structures. The latter depends on the nestedness of terms and thus on the syntactic structure as well.

The grammatical spelling variants are frequently referred to as **orthographic variants**. In general there are several possibilities for orthographic variations. What makes the detection of orthographic variants computationally expensive is the combinatorial explosion of the variants. Some of the various possibilities are shown in the following list:

1. A frequently occurring set of variations appears with respect to capitalisation of letters. Although in most cases uppercase/lowercase variants concern the initial character of terms nearly all variants of uppercase/lowercase combinations can be found in the literature. Some examples are given in 3.9 to 3.13.

(3.9)**Hap1** dimerization domain is composed of ...

---

<sup>3</sup>RefSeq aims to provide a comprehensive, integrated, non-redundant set of sequences, including genomic DNA, transcript (RNA), and protein products, for major research organisms. Available through: <http://www.ncbi.nlm.nih.gov/RefSeq/index.html>

<sup>4</sup>The figure is from NCBI and can be found through: <http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html>

## Growth of GenBank

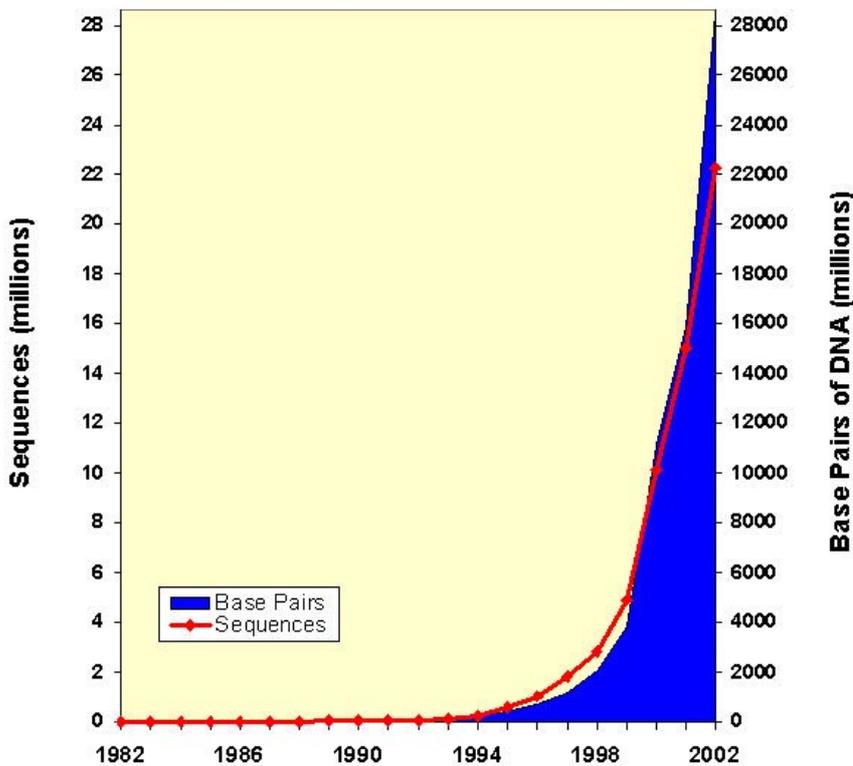


Figure 3.1: **Genbank growth** over the years 1982 - 2002. Considering this huge number of new gene products being discovered it is not surprising that no established naming system is in use.

(3.10)... Hsp90 promotes **Hap1** activation ...

(3.11)**HAP1** has a zinc finger DNA binding domain ...

(3.12)... the **HAP1** gene ...

(3.13)Deletion of the **hap1** gene ...

Similarly, the following list of variants for *HSP90* (the heat shock protein with a weight of 90 kDa) can also be found frequently. They all refer to the same protein. *hsp90* = *HSP90* = *HSP 90* = *Hsp90* = *Hsp 90* It should be noted that on the other hand capitalisation of letters can be relevant as well. While *NaSe* can refer to a chemical compound, *NasE* can refer to nitrit assimilation.

2. Word or term separation is not standardised and thus varies by using or omitting

blank or hyphen characters. The following names in examples 3.14 to 3.18 of an enzyme all refer to the same entity.

(3.14) 1-aminocyclopropane-1-carboxylate synthase

(3.15) 1-Aminocyclopropane-1-Carboxylate-Synthase

(3.16) 1-Amino-Cyclopropane-1-Carboxylate Synthase

(3.17) 1-aminocyclo-propane-1-carboxylate synthase

(3.18) 1-aminocyclo-propane-1 carboxylate synthase

Even names for diseases, which are used not only in biomedical literature, show differences with respect to orthography, for example both *promyelocytic leukemia* as well as *promyelocytic leukaemia* are being used.

3. For naming chemical compounds the IUPAC nomenclature is used as standard. Nonetheless both “ungrammatical” – according to IUPAC – as well as “grammatical” terms occur. Generally round brackets, square brackets and braces are used to express certain chemical conformations in structure. In addition, apostrophes and squared brackets are equivalent when naming side chains of chemical complexes. This leads to grammatical variations. According to the IUPAC nomenclature variations in bracketing are allowed. For example parentheses may be used to add clarity as in (*thiobenzoic*) *anhydride*, however *thiobenzoic anhydride* is also grammatical.

**Paragrammatical variations** are more frequent in life science publications than in common English. One reason is possibly due to the large number of publications by non-native English speaking authors as Netzel *et al.* (2003) point out. Typing errors are an additional source for paragrammatical constructions. To give some examples of paragrammatical variations the following variations were all found in scientific publications for the common organism name *baker's yeast*:

*bakers yeast, baker's yeast, baker 's yeast, bakers' yeast, bakers ' yeast, . . .*

Similarly for the Latin naming variant *saccharomyces cerevisiae* the following paragrammatical variants can all be found in scientific publications:

*Saccharomyes cerevisiae, Sccharomyces cerevisiae, Saccaromyces cerevisiae, Saccharomyes cerevisiae, Sccharomyces cerevisiae, . . .*

Since recall is an issue for an extraction system, robustness has to be an issue, too. This means that paragrammatical variations have to be dealt with since they occur in biomedical scientific publications.

**Syntactic variations** in term formation are a type of synonymic variations as well. At least two aspects have to be considered in this category. One variational possibility arises from permutation when concatenating biomedical terms. It should be noted that the order is not (always) fixed. As an example *tumor suppressor protein p53* and *p53 tumor suppressor protein* refer to the same entity. Another source of variance comes from the fact that the terms can be combined with prepositional modifications in the

term formation process. To refer to the *beta chain* of the complex *succinyl-CoA synthetase* the following syntactic variations all can be found in scientific publications: *succinyl-CoA synthetase beta chain*, or *succinyl-CoA synthetase, beta chain*, or *succinyl-CoA synthetase (beta chain)*, or *beta chain of Succinyl-CoA synthetase*, or *beta subunit of Succinyl-CoA synthetase*.

Once the boundaries of a technical term are recognised the semantic classification of this term is still needed. The question *what does this term denote* or less specific *to which semantic class does it belong* is not trivial at all and needs to be mentioned separately. As can be seen from the already given examples the semantic range that is covered with biomedical terminology is vast. I do not intend to discuss possible semantic classification in this work. Nonetheless to illustrate **semantic variance** some examples are given here: *Mitogen-activated protein kinase kinase 3* is a kinase and thus a protein, *IL-2* is a protein vs. *IL-2 receptor alpha chain* is a protein or a part of a protein complex vs. *IL-2 receptor alpha chain promoter* is a part of a DNA or a part of a gene. It should be noted that the longer the names are the harder it is to detect the semantic class the terms belongs to. This is due to the fact that the amount of ambiguities in syntactic parsing and thus in semantic interpretation increases accordingly.

### 3.1.3 Coordinations and appositions

**Coordination** in term formation is highly relevant and ubiquitous. In the following it should become clear that the disambiguation of coordinated noun phrases is frequently based on semantic and/or syntactic criteria. Basically two kinds of semantic restrictions influence how to calculate the proper bracketing of the coordinated terms. First, the semantic types of the coordinated elements. In general one can assume that only constituents of the same (or, similar, or, strongly related) semantic type are coordinated. The second criterion originates from the selectional restrictions of the verb that governs the coordinated nouns. A verb determines which types of arguments it accepts, or, which type of argument it does not accept. These semantic criteria each correspond to the syntactic arguments of the relational word. Sometimes both have to be taken into account to calculate both the type of the relational word and the intended bracketing of the coordinated structure. One possibility can be that a coordination has to be read as a coordination between sentences instead as a coordination of noun phrases. The following examples give an overview of the various possibilities.

- The example in 3.19 has two possibilities for bracketing of the coordination, *i.e.* in example 3.20 a coordination of the same head noun is reflected shown by the bracketing, where in example 3.21 a coordination of two nouns with different heads is given.

(3.19)However , we have found that HAP2 and HAP3 gene products are involved in the expression of HEM3.

(3.20)However , we have found that [[HAP2 and HAP3] [gene products]] are involved in the expression of HEM3.

(3.21) However, we have found that [HAP2 and [HAP3 [gene products]]] are involved in the expression of HEM3.

To judge which of these two bracketings are correct one first has to identify the semantic type of the entities *HAP2* and *HAP3*. This means that the system has to recognise that both are proteins, or more specific transcriptional activators. Assuming the fact that the corresponding gene products are proteins or stable RNA the bracketing in 3.21 would be a coordination of a transcriptional factor and a protein (or stable RNA). A coordination of nouns referring to entities of the same semantic types is given in 3.20 where either two proteins or two stable RNAs are coordinated. It is highly probable that this is the intended reading and thus the bracketing to be preferred.

- The following example 3.22 shows two more possibilities of bracketing coordinations. In this case here the selectional restrictions from the verb *activate* having a *transcription factor* as agentive subject forces to chose the bracketing given in 3.23. *CLN1* and *CLN2* are both genes and the transcription of both has to be activated.

(3.22) A transcription factor composed of Swi4 and Swi6 proteins (called SBF) activates CLN1 and CLN2 transcription via a positive feedback loop in which Cln proteins activate their own transcription.

(3.23) ... activates [[CLN1 and CLN2] transcription]

(3.24) ... activates [CLN1 and [CLN2 transcription]]

- In contrast to the examples given before (3.25) shows an example where the head noun of the second conjunct is not interpreted distributive. The bracketing in (3.26) is thus preferred. From an ontological point of view *BCK2* is a pathway component, which has to be identified, and this component is coordinated with the noun phrase *other pathway components*. From a linguistic point of view *other* presupposes that *BCK2* is a pathway component.

(3.25) Genetic interactions between BCK2 and other pathway components suggested that BCK2 functions on a common pathway branch with PPZ1 and PPZ2.

(3.26) Genetic interactions between [BCK2 and [other pathway components]] ...

(3.27) Genetic interactions between [[BCK2 and [other pathway]] components]

...

- In 3.28 the plural head noun *lines* has to be distributed. In addition *human* has to be distributed as well. This means that a term identification system should identify *human B-cell lines* and *human T-cell lines*.

(3.28) Using human B- or T-cell lines transfected with ZpCat reporter gene constructs, ...

- It should be noted that besides the well known coordinations *and*, *or*, *as well as*, etc. other biobdomain specific coordinators also appear. In some cases, slashes (/) are used to indicate a complex of different proteins as shown in example 3.29. Another type of coordination symbol are hyphens (-) as can be seen from example (3.30), where it is used to indicate the sequence *Asn-Pro-Ala* consisting of the amino acids *asparagine*, *proline* and *alanine*.

(3.29)Endotoxin increased NF-kappaB p50/p65 heterodimer binding.

(3.30)The Asn-Pro-Ala signature motifs are indicated ....

- A last type of coordination needs to be mentioned. It is the well-known enumeration type, where the coordinated terms are listed by comma separation and the last conjunct is always prefixed with an *and*. Example 3.31 shows an example of a sequence of peptides.

(3.31)The factor thus prepared was a peptide composed of Lys1, His1, Trp2, Gln2, Pro2, Gly1, Met1, Leu2 and Tyr1.

**Appositions** are optional constituents of a nominal phrase (definition according to Bußmann (1983)) occurring either before or after a noun assigning its semantic or ontological category. It is used to provide the reader with more details about a certain entity in a compact manner. To give an example, the noun phrase *German chancellor Gerhard Schröder* has *German chancellor* as apposition assigning the semantic type to the name *Gerhard Schröder*. Some more biology-related examples are shown in example (3.32) and example (3.33). These examples illustrate that an apposition can occur both before or after the corresponding noun phrase. In example 3.33 the apposition *protein kinase* is mentioned after the noun phrase *Cdc28*, where in example 3.33 the apposition is in front of the noun phrase *Pho85*.

(3.32)The Cdc28 protein kinase functions in the G1 to S phase transition of the cell cycle of the budding yeast *Saccharomyces cerevisiae*.

(3.33)The cyclin dependent protein kinase Pho85 is a known negative regulatory factor for two stress response genes , PHO5 and GSY2 , which ...

Frequently it is necessary to recognise the apposition with its appositive information to calculate the correct reading. As can be seen from example 3.34 the term *heterodimer* presupposes two distinct entities *A* and *B* forming a complex. A semantic construction component might thus expect the two words occurring to the left of it to be the necessary entities *A* and *B*, *i.e.*  $A = NF-kappaB$  and  $B = p50/p65$ . However if we resolve before, that *NF-kappaB* is used as apposition and the name of the protein family comprises both *p50* and *p65* and if in addition we recognise that the slash / is coordinating these two words, than the two entities can be linked properly with  $A = p50$  and  $B = p65$ .

(3.34)Endotoxin increased NF-kappaB p50/p65 heterodimer binding.

Example 3.35 shows an embedded term being modified by an apposition. From a pure noun chunking perspective the sequence of nouns *nuclear factor NF-kappa B p50 precursor* would be recognised as a noun chunk with no internal structure like  $[_{nx} \text{ nuclear factor NF-kappa B p50 precursor}]$ . However, to recognise the entity with its correct semantic category one has to recognise the apposition *nuclear factor* modifying *NF-kappa B* and the compositional structure of *p50* modified by *precursor* like:  $[_{nx} [[\text{nuclear factor}] \text{NF-kappa B}] [\text{p50 precursor}]]$ .

(3.35) We report a DNA sequence encoding the nuclear factor NF-kappa B p50 precursor from primary murine B-lymphocytes that differs from that previously published for the murine 22D6 B-cell line.

It is important to note that the main challenge is thus to recognise the noun phrases that constitute the apposition separate from recognising the whole multiword term as one. This can also be characterised as decomposing nominal phrases and resolving its internal syntactic structure.

As we have seen, the range of terminological variance is broad, especially when we consider that combinations of the introduced types of variance are possible. For the information extraction system, that will be introduced in the following, we intended to limit the mentioned terminological problems to a minimum. When building our information extraction system we did not focus on solving all various types of terminological variance but we wanted to focus more on the extraction of relational information. We managed to do that by two means. First, we chose a highly specific extraction task limiting the biological sub-domain to gene interactions only (see section 4.1 and next section 4.2). This means for example that identification of enzymatic reactions was not of interest for this task. Second, the entities we had to deal with come from a list of gene and protein names that is available through databases. We only had to recognise these genes and proteins that were relevant in our scenario. Although this list is not complete it gave us a good starting point as the results show (see chapter 7.1 for the results).

## 3.2 Related work – BioCreAtIvE

A part of the related work has already been introduced in Chapter 2 *NLP technology for detecting information* on the level of general information extraction with a focus on the *Message Understanding conferences*. In addition the previous section describes the terminological obstacles which are especially frequent in the biomedical domain. Nevertheless this does not tell to what degree this affects a common information extraction system. As a consequence the entities that had to be identified by our system was reduced to proteins and genes. To be able to tell this it is worth to compare the general performance figures with the levels of interannotator agreement in the biomedical domain. Interannotator agreement is a measure to tell how good human experts perform when given a specific task. This usually is interpreted as a kind of natural limitation inherent to the task that can be reached by an automated system. For the MUC-7 named

entities an agreement between annotators of 97% has been measured (according to Robinson *et al.* (1999) or according to Marsh and Perzanowski (1998)). In contrast to these numbers for the biomedical named entity annotation Dingare *et al.* (2004) cites two similar results, where according to one of the two (Hirschman (2003)) 89% agreement were achieved and according to the other (Demetriou and Gaizauskas (2003)) measured and agreement of 87%. This difference of up to 10% indicates clearly that the biomedical task is more challenging than the comparable MUC task.

To establish common standards and shared evaluation criteria for comparison among the different approaches **BioCreAtIvE** has been set up in 2003. *BioCreAtIvE* is an abbreviation for *Critical Assessment of Information Extraction systems in Biology*. Following the tradition of the NLP community with the above-mentioned Message Understanding Conferences (MUC) and the Text Retrieval Conferences (TREC), *BioCreAtIvE* aims to establish an evaluation standard for biomedical NLP tasks. It should be noted that besides BioCreAtIvE there are other competitions and assessments like the KDD Cup <sup>5</sup> and the NLPBA conference <sup>6</sup> with comparable efforts.

I've chosen to present the BioCreAtIvE initiative since it is the assessment which is most related to the work presented here. This is reflected by the two main objectives of BioCreAtIvE which were defined as follows.

- Define *biologically meaningful* tasks, tasks that would be recognised by biologists as a contribution to their work and that constitute a meaningful challenge for current text mining systems.
- Support "gold standard" data for training and testing. The data is made available in sufficiently large quantities.

The BioCreAtIvE offered two related tasks (*i.e.* task 1A and 1B) plus one separate task (*i.e.* task 2) for the participants. Tasks 1A and 1B were both related to *entity identification*. Task 2 was after automatising the process of functional annotation of genes. Overall, 27 groups participated in the assessment from Europe, USA and Asia. The tasks are briefly described as follows:

**Task 1A** was also called *gene name extraction* and was about identification of gene names in PubMed abstracts. One should note that not only gene name mentions were identified but mentions of *names* related to genes, including binding sites, motifs, proteins, promoter regions, the corresponding proteins etc.<sup>7</sup> This task can be seen as a kind of building block for following tasks, task 1B and task 2.

---

<sup>5</sup>KDD abbreviates *Knowledge Discovery and Data mining*. More details on the KDD Cup 2003 are available through: <http://www.acm.org/sigs/sigkdd/kdd2003/>.

<sup>6</sup>NLPBA abbreviates *Joint Workshop on Natural Language Processing in Biomedicine and its Applications* and is a annual workshop. In 2004 it was held out in conjunction with the Coling conference. As a part of the workshop a shared task for bio-medical named entity recognition from the GENIA corpus. Details including test- and training-data are available through: <http://www.genisis.ch/~natlang/JNLPBA04/>

<sup>7</sup>Compared to the experiment which I describe in this work here, the granularity of the results of this task are still too coarse grained. Additional refinement would be necessary concerning the distinction gene vs. corresponding gene product.

**Example:** “The LMW FGF-2 up-regulated the PKC epsilon levels by 1.6 fold; by contrast the HMW isoform down-regulated the . . .”

For this given example the answer file had to contain: *LMW FGF-2*, *PKC epsilon*, and *HMW isoform*.

**Participants:** All in all 15 teams participated in the task 1A.

**Data:** The BioCreAtIvE consortium provided training data with 7.500 sentences and approximately 9.000 mentions of gene names. The provided test data had 2.500 sentences with roughly 3.000 gene name mentions. The final evaluation data had 5.000 sentences with 6.000 gene name mentions.

**Results:** The best participants achieved results with a recall of 85%, a precision of 86%. The system with the best F-score reached 83%. Altogether four groups had a precision and recall of > 80% showing that the best performing systems were quite close together.

**Task 1B** is related to task 1A and was about normalisation of gene name mentions in text. The systems had to resolve the gene mentions to unique gene IDs by requiring to associate a set of unique identifiers (of genes) with a given research journal abstract.

**Example:** *est-6* and *esterase-6* had to be recognised as referring to the same entity.

**Participants:** There were 8 groups participating in this task.

**Data:** The data provided by the consortium had again training, test and final evaluation data. Each of the data sets were provided in three partitions for fly, mouse and the yeast organism. The size for the training data consisted of 500 annotated abstracts for each organism and for the test data it consisted of 108 abstracts for fly, 250 for mouse and 110 abstracts for yeast. The evaluation data had for each organism 250 abstracts.

**Results:** The results differed according to the organisms. For yeast the balanced precision and recall was around 92%, where for fly 82% were achieved. For mouse the best result was 79%. The high interannotator agreement of 96% shows that this task was well-defined in terms of being reproducible. This compares well with MUC-7 interannotator agreement of about 97%.

**Task 2** consisted of two subparts. Task 2.1 was such, that for a given triple of protein name, document and GO id, the system had to return the evidence text from the document supporting the annotation of the protein with the GO id. Within task 2.2 for a given a pair of protein name and document, the system had to return  $n$  GO-annotations for the given protein based on the given document. In addition the supporting evidence text from the document for each annotation.

**Example:** <sup>8</sup>

---

<sup>8</sup>Example was taken from Krymolowski *et al.* (2004).

```
PMID: 10026212
go code: GO:0004337
go name: geranyltranstransferase activity
go def: Catalysis of the reaction: geranyl
        diphosphate + isopentenyl diphosphate
        = diphosphate + trans,trans-farnesyl
        diphosphate.
evidence: Geranylgeranyl diphosphate (GGPP) synthase (GGPPSase)
catalyzes the synthesis of GGPP, ...
```

**Participants:** 9 groups participated in the GO functional annotation task.

**Evaluation:** The results for task 2.1 that the participants had to submit were triplets: protein-paper-GO term plus an additional evidence text. The curators evaluated the GO terms using three possible categories: high, generally and low. *High* was used for GO-terms or protein names in case they were correct. *Generally* was assigned for GO terms that were not entirely wrong but too general to be really useful for annotation. In case of protein names, *generally* was used if the mentioned protein was not there but a homologue from another organism. *Low* was used if the prediction of the GO term or protein name was wrong. Every individual result of the participating teams had to be evaluated which turned out to be too much work (three curators dedicated several months to the revision of the results). As a consequence some, not all, results could get checked, meaning that some entire proteins were skipped (*i.e.* not evaluated).

**Results:** The range of the “perfect predictions” for task 2.1 ranged between 1.53% and 80.00%. It should be noted that the team with the most results (1048) had 25.57% “perfect” predictions. The system achieving the 80.00% precision had only 45 results presented. The average precision in this category was 30,2% (note that I did not consider the amount of data for each of the achieved results). The precision numbers for correct protein and “general” GO term ranged between 0% and 14.57%. The precision numbers for task 2.2 for “perfect” predictions ranged between 0.64% and 34.15% with an unweighted average of 14.03%. The precision numbers for correct protein and “general” GO term ranged in task 2.2 between 0% and 10.71%. Altogether the results show inhomogeneity (*i.e.* a wide range) with respect to the precision numbers regardless of the subtask. In addition the best results show that still much work has to be invested in systems to reach satisfying results for this task.

To ensure that the results were automatically generated temporal criteria, volume criteria and evidence criteria had to be fulfilled. The time for the production of the data was short (see further down the temporal schedule). The amount of data that had to be processed in the given time period should be at least a few hundred papers to annotate. Evidence data had to be provided that justifies the selection of the annotation.

The BioCreAtIvE time schedule for the participants went from July 2003 until March 2004 where the following deadlines had to be met:

**July 2003:** Training data was released with initial task guidelines. The participants had about 2 months to build a first system.

**September 2003:** Release of the full training set and revised version of the task guidelines was provided.

**November 2003:** Evaluation data release and submissions. The groups had to return their results a few days later. As already mentioned temporal criteria in combination with volume and evidence guaranteed that the results were automatically generated.

**December 2003:** The results are automatically evaluated and the evidence was checked by experts (*i.e.* Swiss-Prot annotators).

**March 2004:** The workshop after the evaluation was held out in Grenada with a publication and discussion of the achieved results.

It needs to be mentioned that a second BioCreAtIvE challenge will be held during October 2006, with the corresponding workshop to be held in spring 2007. It will consist of three tracks.

1. One track will focus on finding mentions of genes and proteins in sentences drawn from PubMed abstracts. It will be the same as Task 1A from the first BioCreAtIvE assessment.
2. A second track will be about producing a list of the EntrezGene identifiers for all human genes and proteins mentioned in a collection of PubMed abstracts. It is comparable and to the first BioCreAtIvE Task 1B.
3. The third track is new. It will involve identifying protein-protein interactions from full text papers. It will include the extraction of excerpts from those papers that describe experimentally derived interactions for curation into one of two interaction databases: IntAct (Hermjakob\* *et al.* (2004) and MINT (??).



# Chapter 4

## System outline

### 4.1 Establishing a collaboration

The previous chapter has shown that more and more information that is relevant for biology and medicine is buried in scientific publications. To get hold of this information database maintainers like Swiss-Prot (see Boeckmann *et al.* (2003) for details) extract this information manually. Since there is a massive amount of publications which is steadily growing this task is extremely time consuming, labour intensive and has to be done over and over again. The main source for information in the field of medicine and biology is PubMed<sup>1</sup>. Although in most related publications MEDLINE is referred as the most relevant textual knowledge deposit for biomedical literature, I will refer to PubMed, since PubMed includes all MEDLINE references and has additional citations for biomedical articles preceding the date that a journal was selected for MEDLINE indexing. There are some additional life science journals submitted as full text to PubMed-Central as well. PubMed is a bibliographic database as a service of the National Library of Medicine. On April 18th 2004 it included over 14 million citations for biomedical articles back to the 1950's. And, every year more than 500,000 new publications are added.

Suprisingly, a similar problem can be identified for gene, protein, enzyme or peptide sequence databases like Swiss-Prot, BRENDA (Schomburg *et al.* (2002)), KEGG (Kanehisa and Goto (2000)) etc. Although a lot information is accessible in a structured way from databases, a large amount of information is stored through natural language comments in so-called commentary lines. One reason for including comment lines is that in a series of cases it is not possible to foresee and provide database schemata that model all possible variants of information that might come along when building up such a database. The field of molecular biology is so complex that - as an example - for a certain protein or gene it is not always clear from the beginning what different types of information will have to be added. To give an example, the functional analysis associated with protein complexes is such a case. There is no exhaustive list of protein functions and contexts. One and the same complex can act as an activator in a one

---

<sup>1</sup>PubMed is online available trough: <http://www.ncbi.nlm.nih.gov/PubMed/>.

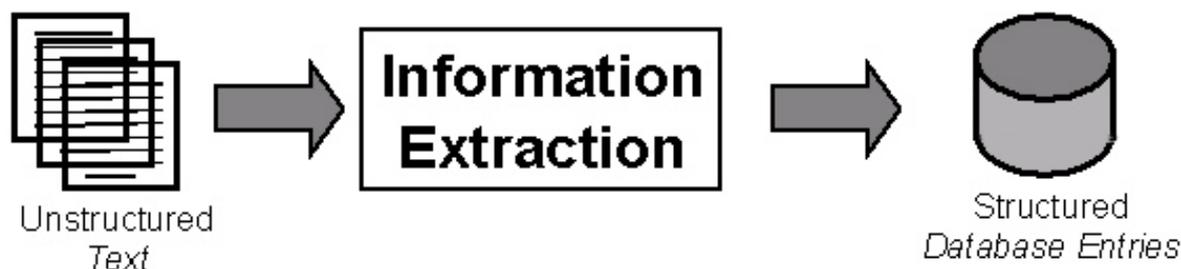


Figure 4.1: A simplified picture demonstrating the information extraction process that is responsible for transferring information from unstructured data (text) to structured database entries. For common biological databases this work is carried out in labour-expensive manual work.

context and in a different context it can act as a suppressor. In other cases it might be clear what the exact structure of the data should be, but it was not considered that important for later research. To give an impression of the numbers of natural language terms in databases I inspected two commentary slots in Swiss-Prot (, *i.e.* the Function and the Subunit slots). The actual database version at the time of writing this thesis contained 61,984 Function slots with 1,366,279 tokens and 25,865 Subunit slots with 294,002 tokens. This means that each peptide entry has 22 and 11 words of comment in average for the Function and Subunit slot, respectively. Altogether a valuable piece of information.

The overall goal of the work described within this and the following chapters was to build system to automatically extract information from large-scale textual data relevant for the biomedical domain. A schematic representation is shown in figure 4.1. The field of biomedicine covers a huge variety of subdomains. It is way to complex to cover all facets within a dissertational project like this. As a consequence I had to decide on a highly specific biomedical subtask. On the one hand it has to meet the requirements of relevance for people working in the field of biomedicine. And, at the same time it had to contribute to the scientific life of the NLP community. This means that an adequate and relevant task had to be set up in co-ordination with a field expert. I decided for the given reasons to develop the work in close collaboration with a group of biologists and bioinformaticians that base their work on massive amounts of data and process them computationally. The Bork group at the European Molecular Biology Laboratory (EMBL) in Heidelberg met all the requirements, and were at that time looking for a collaboration with someone working in the field of computational linguistics. The overall aim was to automatically access additional data for a project called STRING (Search Tool for the Retrieval of Interacting Genes/Proteins<sup>2</sup>) from text. The STRING project is about the development of a search tool for the retrieval of interacting genes and proteins. The interactions include direct (*i.e.* physical) and indirect (*i.e.* functional) associations, which are derived from four sources:

1. Genomic Context

---

<sup>2</sup>The STRING system can be used through: <http://string.embl.de/>

2. High-throughput Experiments
3. (Conserved) Coexpression
4. Previous Knowledge

STRING is a database of known and predicted protein-protein and protein-gene interactions for different organisms automatically retrieved from the above mentioned sources. For the aim of this work these sources need not to be specified in detail.

On April 18th 2004 the database contained 444,238 genes in 110 species. The last of the above mentioned sources, “previous knowledge”, subsumes textual data. This means that extraction from results on protein-protein and protein-gene interactions for different organisms from scientific publications is a necessary requirement to achieve a high recall of results. But, regarding the amount of known proteins and the amount of organisms in combination with possible variety of interactions, this task was still too wide, *i.e.* not specific enough. After intensive discussions we decided to narrow down the task to the following two questions that the system should answer.

For a given (large) set of known proteins and genes we wanted to know:

1. **Which proteins regulate the expression (*i.e.* transcription or translation) of which genes?**
2. **What kind of regulation takes place, *i.e.* activation, repression, or underspecified?**

For reasons of better comprehensibility for non-biologists and the computational linguistics focus of this work I will shortly describe the biological background behind these questions. The main concepts that will be explained in what follows are: gene, protein, RNA, transcription factor, and gene expression.

A term like gene or protein is subject to several discussions within the biological domain. However, within this information extraction scenario it is enough to regard a gene as a sequence of nucleotides on a DNA strand that has a name. In most cases there exists a reference to a database entry for each gene providing the sequence of the gene and its name. In this sense, we can treat a protein analogously as being built of a sequence of peptides with a name. Again, in most cases this information can be obtained through a database. In biology the term “gene expression” usually refers to the ability of a gene to be transformed to (,or, “to code for”) a biologically active protein (or, a stable RNA<sup>3</sup>). This production is carried out by special enzymes and consists of two subprocesses, the step of transcription and the subsequent step of translation. Transcription refers to the transformation of genetic information into information which is encoded in a messenger RNA (mRNA, a specific type of RNA). The translation step describes the transformation of this mRNA information into a proteinaceous structure (*i.e.* a protein). In some cases the genetic information is only transcribed into a stable type of RNA, *i.e.* transfer RNA (tRNA) and ribosomal RNA (rRNA). These are not translated into proteinaceous structures.

---

<sup>3</sup>RNA abbreviates ribonucleic acid. It is a nucleic acid polymer, *i.e.* a chain of nucleid acids.

Whether a gene is expressed or not is not only determined by the gene itself but is dependent on contextual factors. Many of the genes in a genome of an organism are expressed only if the gene is “switched on” at a certain time and then *speaks out*. The “turning on” (and “turning off”) of the genes is performed by so-called transcription factors (TF). Certain proteins act as such transcription factors. They govern the expression of corresponding genes. Each TF does not act on its own on one gene but forms complexes with other TFs to be able to regulate gene expression as a whole. For the human organism there are around 30,000 regulatory regions known in the genome. In average 2 to 3 TFs act co-ordinated. As already mentioned, we decided to reduce complexity as much as possible and thus to restrict ourselves to the question of determining which protein (*i.e.* transcription factor) takes part in the regulation (*i.e.* activation or repression) of the expression of which gene<sup>4</sup>. Even more specifically, the biologists wanted to know for a given set of proteins and a given set of genes, if and what kind of regulation can be found for these protein-gene pairs. This means that the system need not answer which other proteins participated in building the complex that regulates the gene expression.

Another concern which we had to address with our approach was the organism where the gene expressions takes place. I like to mention that besides the fact that the principles which can be deduced for a specific organism are interesting in themselves there is always the hope that the findings will have wider relevance. There is always the hope that the principles that have been discovered for one special organism can be generalised and found to be valid for organisms as well. In the best case they bring new knowledge about the human organism and influence further research (for example drug development). It is quite common in the bioinformatics domain to start with the yeast organism and then to adapt the system to be used for other organisms as well. The reason is that from all the given organisms the most complete knowledge is available for yeast. Large-scale experiments have been carried out for the yeast organism and it has a completely sequenced genome. All this knowledge is well documented and can be retrieved through the Saccharomyces Genome Database (SGD, see Dwight *et al.* (2002) for details). The basic idea was that once we have a system that produces relevant results for yeast we could later easily adapt our system to produce results for other organisms as well.

To summarise the overall orientation of this work from the previous paragraphs: For each organism the system should produce triples specifying a regulator (R), a target (T) and a type of Regulation (typ): R-typ-T. Each target can code for a regulator. And since homonymic naming of genes and proteins are frequent (see section 3.1 item 3.1.1 for details) in biology a target can occur with the same name as a regulator. The result is thus a network of gene expression relations.

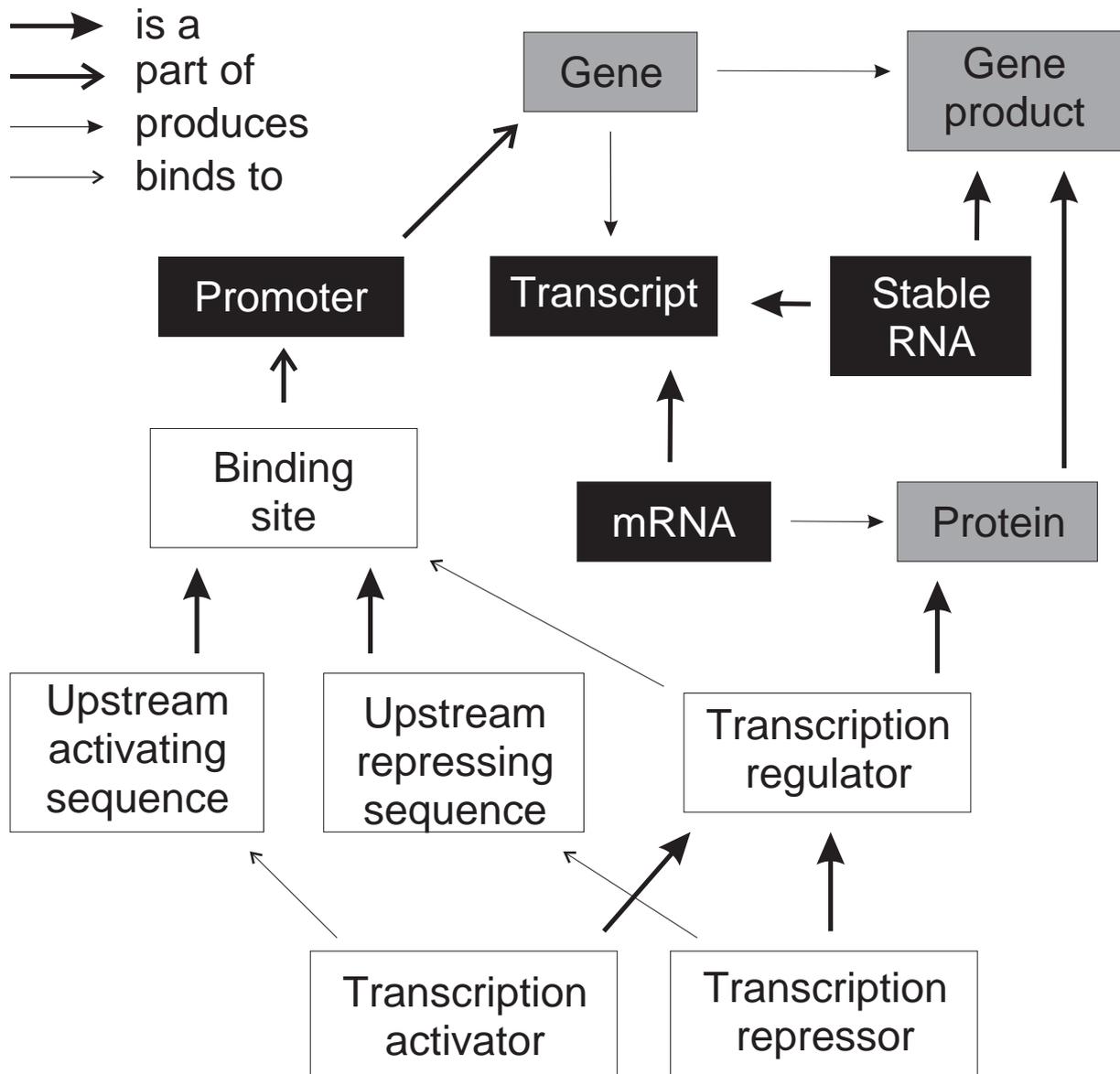


Figure 4.2: A **simplified ontology for transcription regulation**. The boxes show the main concepts that are necessary for transcription regulation. The background colour used for each term signifies its semantic role in relations: regulator (white), target (black), or either (gray). The four different types of arrows between the concepts identify the relevant relations between these concepts (*i.e.* *is a*, which also can be read as *is a type of*, *part of* meaning *is a substantial part of*, *produce* subsumes *transcription* and *translation*, *binds to* means that a protein binds to a specific *binding site*). Only one relation between two neighbouring concepts is labelled although more relations might exist.

## 4.2 What data should be extracted

Extracting that a certain sentence describes the regulation of gene expression is a challenging problem due to several reasons. On the one hand there are semantic reasons like homonymy. As mentioned above a gene and its corresponding gene product (*i.e.* the protein) can have the same name. This makes it difficult disambiguate between these two entities. Thus one challenge is to distinguish gene expression (*i.e.* a protein-gene interaction) from a protein-protein interaction. Another challenge is more of syntactical nature. It relates to the fact that gene expression can be expressed in a variety of ways – possibly mentioning neither the biological process (*expression*) nor any of the two biological entities (*genes* and *proteins*) involved.

The succeeding examples (4.1) through (4.3) illustrate a set of variants all describing the regulation of the expression of a gene **X** through a protein or regulator **R**. After each of the examples the information that should be extracted by the system is listed.

(4.1) Transcription of the HXT4 gene is regulated by Gcr1p and Gcr2p in the yeast *S. cerevisiae*.

To extract (i): Gcr1p regulates HXT4 (in yeast)

To extract (ii): Gcr2p regulates HXT4 (in yeast)

(4.2) Overproduction of Ndd1 also enhances the expression of SWI5, whose ...

To extract (i): Ndd1 up-regulates SWI5<sup>5</sup>

(4.3) We show here that Snf1 regulates transcription of FLO11 , which encodes a cell surface glycoprotein required for invasive growth.

To extract (i): Snf1 regulates FLO11<sup>6</sup>

(4.4) Fzf1p of *Saccharomyces cerevisiae* is a positive regulator of SSU1 transcription and ...

To extract (i): Fzf1p up-regulates SSU1 (in yeast)

In example (4.1) it is clearly mentioned that *HXT4* is a gene (*i.e.* the *HXT4* gene), which is getting transcribed and the regulators (*i.e.* *Gcr1p* and *Gcr2p*) are mentioned. It is not specified if both regulators can act separately or only as a complex. In contrast to this, example (4.2) does not mention explicitly that *SWI5* is a gene. But, since *SWI5* occurs as an argument of the relational noun *expression of* it has to be a gene. Similarly in example (4.3), to identify *FLO11* as a gene it is necessary to know that only genes can be transcribed, thus *transcription of* triggers that *FLO11* is a gene. Example (4.4) shows a mention of activation (or, up-regulation) of a certain expression. These examples already show the interdisciplinary setting of this project. Both linguistic knowledge as well as biological knowledge have to be considered for building a information extraction system.

---

<sup>4</sup>It should be noted that we avoid answering the question of how different TFs influence each other when commonly regulating one and the same gene.

<sup>5</sup>Note that the organism is not specified.

<sup>6</sup>Again the organism is not specified.

As a consequence we decided to first set up a description of the domain concepts and relations that are relevant for this task. Figure 4.2 shows the result of this effort, a simplified ontology for gene expression. It provides an overview of the biological entities involved in gene expression as well as the relevant ontological relationships between the concepts. The relations show the different possibilities of how these entities can interact. To build this simplified ontology before writing extraction rules was a great help in at least two respects. On the one hand it was useful to describe a complex relationship in an transparent and well-understandable picture for a non-biologist. For example when looking at a gene and its products one can easily see that a *gene* produces (*i.e.* codes for) either a *gene product* or a *transcript*. A *transcript* is either a *stable RNA* or *mRNA*. *mRNA* produces (*i.e.* translates to) a protein. These dependencies can be read fairly easy from the picture.

On the other hand it immediately suggested a large number of relevant patterns when writing extraction rules. Some examples that correspond to the relations in the picture are:

(4.5) *promoter* contains *upstream activating sequence*

(4.6) *promoter* contains *upstream repressing sequence*

(4.7) *transcription regulator* binds to *promoter*

All mentioned patterns follow from indirect relationships via the concept *binding site*.

Generally speaking what we can see from the picture is that a gene codes for (produces) a product. As (substantial) a part of a gene we have a promoter that has a binding site which is either an (upstream) activating site, a (upstream) repressing site or it is underspecified in the case of simply being mentioned as transcription regulator. This means that a transcription regulator binding to that binding site can affect the expression of the corresponding gene to be activated or down-regulated (and sometimes it is unknown). The task for the extraction experiment is the extraction of pairs of transcription factors and genes and additionally the relation between these pairs (*i.e.* activation, repression or underspecified regulation).

It is often not known whether the regulation takes place at the level of gene transcription (*i.e.* the process of converting the genetic information to mRNA) or translation (*i.e.* the process of translating mRNA to a peptide ) or by an indirect mechanism. Indirect mechanism comprises about anything besides transcription or translation that effects expression of genes. For example in some cases gene expression of some gene X1 takes place after knocking out some other gene X2. This would mean that X2 regulates the expression of X1. But it may not be directly - perhaps X2 codes for a kinase that has to phosphorylate some third protein (a transcription factor) to allow this protein to regulate the expression of X1. Or perhaps X2 phosphorylates yet another protein, which when phosphorylated, is capable of binding to the transcription factor (regulating the expression of X1) and thereby enable/prevent it from binding to the DNA.

For this reason, and for reasons of simplicity, we decided against trying to extract how the regulation of expression takes place. We did, however, strictly require that the

extracted relations provide information about a protein (the regulator, **R**) regulating the expression of a gene (the target, **X**), for which reason the following three requirements must be fulfilled:

1. It must be ascertained that **the sentence mentions gene expression**. *The protein R activates X* fails this requirement, as **R** might instead activate **X** post-translationally. This means that it is not sure whether the semantic type of **X** is a *gene* and not a *protein*. Whether the relation should be extracted or not thus depends on the type of the accusative object **X** – the type should be *gene* and not any kind of *gene product* – and whether it thus satisfies the selectional restrictions for the verb. Without a head noun specifying the type, **X** remains ambiguous and thus the whole relation remains underspecified and should not be extracted. It should be noted that roughly two thirds of the gene/protein names mentioned in our corpus are ambiguous for this reason.

(4.8) Conversely, NIK activates NF-kappaB and this is blocked by kinase-dead Akt.

(4.9) A mediator role of Src is supported by the ability of c-Src and v-Src to activate STATs and ... (induce transcription through APP promoters).

(4.10) The oncoprotein MDM2 binds and inactivates p53.

(4.11) These results, together with our previous results, suggest that NEPP11 activates the expression of HO-1 ...

The examples (4.8) and (4.9) show sentences mentioning activation but not in the sense of activation of gene expression. In contrast to that example (4.10) mentions inactivation. Nonetheless the type of the arguments can not be determined from the syntactic structure and the head nouns specifying the type of *p53*. Hence this sentence has also to be excluded. Example (4.11) would meet the requirement since it is explicitly mentions that NEPP11 *activates the expression* of HO-1.

2. The identity of the regulator (**R**) must be appropriate. A statement like *The X promoter activates X expression* fails this requirement, as it can not be seen from this phrase which transcription factor activates the expression when binding to the **X** promoter. From a linguistic point of view this implies that the semantic type of the subject argument (*i.e.* *agens*) has to be appropriate and informative.

(4.12) ... the *bvgR* promoter activates the expression of *bvgR*.

(4.13) This led to the hypothesis that the binding of phosphorylated BvgA to the *bvgR* promoter activates the expression of *bvgR*.

Regarding the phrase in example 4.12 it is not expressed which transcription factor regulates the expression of *bvgR*. But, regarding the complete context, as shown in example 4.13, a different fact is reported. The binding of a phosphorylated BvgA to the *bvgR* promoter activates expression. These two cases have to be distinguished. It should be noted that in 4.13 there is the definite nominal phrase *the bvgR promoter*, which describes *BvgA*. The fact that *bvgR promoter* is responsible for the *bvgR gene* is well-known for a biologist.

3. The identity of the target/regulatee (**X**) must be known. “The transcription factor **R** activates **R** dependent expression” fails this requirement, as it is not known which gene’s expression is dependent on **R**. The semantic types allowed for the patients arguments should thus also be restricted.

(4.14) In the absence of Mak2p or Mak3p, Sty1p fails to phosphorylate the Atf1p transcription factor or induce Atf1p-dependent gene expression.

The example 4.14 does neither specify the transcription factor of *Atf1p* nor does it specify the product that depends on *Atf1p*. It has to be noted that it would be tautological to extract that the transcription factor of *Atf1p* regulates the *Atf1p*-dependent gene expression.

The two last requirements are important to avoid extraction from seemingly non-informative phrases. It should be noted that in some cases this information can be retrieved through resolving co-referring expressions. But this is beyond the scope of this work.

With the colouring of the entities in figure 4.2 I intend to help discern entities that are relevant for regulation (boxes with white background colour) from entities which are relevant as regulatees (boxes with black background colour). The entities shown with grey boxes can appear as both regulators and regulated ones.

Before going into the details of the system another complication for extraction of gene expression relations has to be mentioned. It arises from the fact that experiments are carried out in laboratories sometimes manipulate the genetic information of the organisms. This means that biological scientific texts often mention what takes place when the genetic material of an organism is artificially modified in a particular way. As a consequence for the extraction of relations two cases have to be distinguished. The modification can (i) reverse part of the meaning of the verb or (ii) lose the type of the relation completely:

As an example for case (i), from the sentence “Deletion of **R** increased **X** expression” a biologist concludes that **R** represses expression of **X**. The key point is to identify that “*deletion* of **R**” implies that the sentence talks about an experiment in which **R** has been removed, but that **R** would normally be present and that the biological impact of **R** is thus the opposite of what the verb *increased* alone would suggest.

In the case of (ii) the verb will lose part of its meaning. The example sentence “Mutation of **R** increased **X** expression” implies – for a biologist – that **R** regulates expression **X**, but a biologist can not infer whether **R** is an activator or a repressor. In this case *mutation* is treated different than *deletion* in the previous example.

Finally, there are those relations that should be completely avoided as they exist only because they have been artificially introduced through genetic engineering. In the extraction method that is described in the following chapter all three cases are addressed.

### **4.3 Conclusions for the system architecture**

After a series of discussions taking into account both the biological and linguistic challenges we have opted for a rule based approach – implemented as cascaded finite state automaton – for the extraction of gene expression relations for mainly two reasons. The first reason is, that a rule based approach allows directly ensuring that the requirements stated above are fulfilled for the extracted relations. This means that rules specifying these requirements can be used. We ensure thus to attain high precision on the extracted relations. Since the setting for this experiment was mainly biologically motivated – and the results were intended to be used in a biological scenario – the precision is of high importance. As a consequence we focus in our evaluation on the semantic correctness of our method rather than on the linguistic correctness. As long as linguistic or grammatical errors do not result in semantic errors, we did not consider them be errors. Conversely, even a grammatically correct extraction is considered an error if it is semantically incorrect.

The second reason for choosing a rule based approach is that this approach is theory-driven and highly interdisciplinary, involving computational linguists, bioinformaticians, and biologists. The rule based approach benefits more from the interplay of scientists with different backgrounds, since known biological constraints can be explicitly incorporated in the extraction rules. As we have seen in this chapter the simplified ontology reflects the biological facts and thus it allows transforming this knowledge explicitly into templates and rules.

# Chapter 5

## Summary

The first part of this thesis gives an overview of standard NLP techniques that can be utilised for effective extraction of information from biomedical scientific publications. These techniques comprise Information and Passage Retrieval, Information Extraction, and Question Answering. In addition, an overview of some standard competitions is given demonstrating how an evaluation of such systems can be carried out. The intent of these sections is not to replace an introduction to any of these NLP-subfields but rather to give an overview of possible methods for extraction and retrieval of knowledge from documents and to show their capabilities and limitations.

The methods and techniques introduced in the general background section are independent of a specific domain. However, the main obstacle for biomedical text processing is the detection, classification and identification of technical terminology. The main terminological hurdles (like synonymic, homonymic, orthographical, paragrammatical and other types of variance) are described and discussed in detail. The same section gives an overview of related work in the field of biomedical Information Extraction. Mainly the key issues that have been addressed within the *BioCreAtIvE (Critical Assessment of Information Extraction Systems in Biology)* competition 2004 are reported. The *BioCreAtIvE* was an assessment for standardising NLP tasks for biomedical text processing.

The first chapter concludes with a description of the setting for a pilot study on biomedical information extraction. This study was carried out in collaboration with the Bork group at the European Molecular Biology Laboratory (EMBL) in Heidelberg. The main part of the chapter explains the details of the questions that the biologists at EMBL needed to have answered by such a system. It is a major achievement in this interdisciplinary setting that we managed to work out the details that such a system had to account for. On the one hand it was vital to adjust to and specify the needs of the biologists. At the same time this question had to fit an appropriate natural language processing system. It is important to mention that the precise questions we worked with explain – at least partly – the success and good results that were achieved with our system.



## **Part II**

# **Extracting Gene Regulatory Networks**



# Chapter 6

## System overview (Methods)

This chapter describes the overall architecture of the information extraction system that was discussed in the previous section. The system was implemented as a CASS grammar (see Abney (1996) for details about CASS) and applied to PubMed abstracts extracting gene regulatory networks. Preliminary work for the development of such an IE system has been presented within Reyle and Šarić (2002) for catalysing reactions, where the reactants have been extracted from database comment lines. Parts of the system that is being described in the succeeding sections as well as parts of the results can be found in Šarić *et al.* (2004a). A further development and adaptation of the system to related questions is presented in Šarić *et al.* (2005a).

---

**Level 0 - 2 Processing on word level**

The levels 0 - 2 comprise the tokenising step, the part-of-speech tagging step and the semantic labelling step. All three levels work on the level of words.

**Level 3 - 5 Syntactic and semantic processing**

These processing levels take the word-processed corpus from the previous levels as input and recognise certain syntactic structure and annotate them with semantic labels. The last step comprises output generation and visualisation.

---

Table 6.1: **Overall System Architecture:** Overview of the overall system components. A distinction between processing on word level and processing on phrasal and sentential level is shown in the table.

To give a rough overview of the systems architecture the simplified picture in 6.1 shows the two main building blocks of the system. The first one refers to processing on

word level. The second refers to processing on sentential level. Each of these two blocks is shown more detailed in table 6.2 and table 6.3, where sub-levels are presented and briefly described.

<b>Level 0 Tokenising and multiwords</b> This level comprises the detection of word and sentence boundaries. The text is transformed in a one word per line format. In the succeeding step multiwords are recognized and recomposed to one multiword per line.
<b>Level 1 Pos-Tagging</b> This processing level takes the tokenised corpus as input and assigns to each word (or multiword) a part-of-speech tag (and the lemma in addition). The lexicon is based on common English words with a biology-related extension.
<b>Level 2 Semantic labelling</b> A semi-automatically generated list of words and multiwords like gene names, e.g. <i>GAL4</i> , ..., as well as cue words relevant for gene transcription, e.g. <i>transcription factor</i> , <i>binding site</i> , <i>gene</i> , ... get annotated with a semantic label.

Table 6.2: **Word processing architecture:** Overview of the system components that are arranged in cascades for the processing on word level. Each level (*i.e.* level 0 - 2) represents a system component. The output of one level  $i$  is fed into the next level  $i + 1$ .

Each of the mentioned processing levels in table 6.2 as well as the processing levels in table 6.3 will be described within the following sections in detail providing examples for all relevant details. It should be noted that within the section on tokenisation the corpus to which the whole system is applied is described as well.

## 6.1 Corpus and Filtering

In this section I describe the development corpus, the evaluation corpus that has been used for the development and evaluation of our information extraction system. In addition I describe the filtering procedure that has been applied to obtain organism-specific parts of the corpus.

**Level 3 Named entity chunking**

Based on the pos-tags and the semantic labels, a cascaded chunk grammar recognises noun chunks, which are relevant for the gene transcription domain, e.g. [*nxgene* The GAL4 gene].

**Level 4 Relation chunking**

This level of processing recognises relations either through verbs and their arguments or relational nouns and their arguments. This processing takes place on top of the chunked named entities. An example is:

(6.1) [*controlrel* [*nxevregul* The expression of [*nxgenes* the cytochrome genes CYC1 and CYC7 ]]] is controlled by [*nxgene* HAP1].

Here a control relation between CYC1 (CYC7) and HAP1 is recognised.

**Level 5 Output and Visualisation of Data**

From the annotated relations pairs of transcription factors and corresponding genes are extracted. From the above given example two regulations are extracted: regulates (HAP1, CYC1) and regulates (HAP1, CYC7). For the visualisation of the chunked data TIGERSearch is used. The visualisation of the gene regulatory networks a map is produced showing the interactions with bullets and arrows.

Table 6.3: **Sentential processing architecture:** Overview of the system components that are arranged in cascades for the processing of sentential information. Each level (*i.e.* level 3 - 5) represents a system component. The output of one level  $i$  is fed into the next level  $i + 1$ .

The PubMed resource was downloaded on January 19, 2004 through <http://www.ncbi.nlm.nih.gov/pubmed/>. 58,664 abstracts<sup>1</sup> related to the yeast *Saccharomyces cerevisiae* were extracted by looking for occurrences of the terms “*Saccharomyces cerevisiae*”, “*S. cerevisiae*”, “Baker’s yeast”, “Brewer’s yeast”, and “Budding yeast” in the title/abstract or as head of a MeSH term<sup>2</sup>. These abstracts were filtered to obtain a reduced version with 15,777 abstracts that mention at least two gene or protein names. Note that the details for the detection of protein or gene names within PubMed abstracts have been carried out before and will be described within section 6.4. The details of the protein and gene name detection are described in section 6.4. This reduced corpus was subsequently divided into a development (referred to as “*S. cerevisiae*<sub>dev</sub>” in the following) and an evaluation set (referred to as “*S. cerevisiae*<sub>eval</sub>” in the following) of 9137 and 6640 abstracts respectively. More details on the number of sentences or tokens in

<sup>1</sup>It should be noted that PubMed consists of abstracts mainly and some full-text articles.

<sup>2</sup>MeSH (Medical Subject Headings) is a controlled vocabulary that is used to manually annotate PubMed abstracts or articles. This annotation comprises more than 19,000 terms to describe the subject of each PubMed article. It helps provide uniformity in the indexing and classification of articles. Typically, 10 to 12 MeSH terms are assigned to each article.

each organism specific corpus part are shown in the overview table table 6.4.

Organism	NCBI ID	Abstracts	Sentences	Tokens
<i>S. cerevisiae</i>	4932	58.664	430.553	9.447.237
<i>S. cerevisiae</i> <sub>test</sub>	4932	9.137	72.793	1.837.004
<i>S. cerevisiae</i> <sub>eval</sub>	4932	6.640	44.353	1.286.701
<i>M. musculus</i>	10090	688.937	4.715.893	106.027.447
<i>B. subtilis</i>	1423	16.270	92.603	2.022.852
<i>E. coli</i> K12	83333	195.492	1.284.491	28.568.983

Source	Papers	Tokens
PubMed Central	5.075	19.199.318

Table 6.4: Overview of the used Corpora: The first column shows the name of the organism. Each of them corresponds to a set of PubMed abstracts referring to this organism. The second column shows the organism specific NCBI ID which is used frequently as unique identifier. The following columns show the corpus size for each of these organism specific sets of abstracts in terms of *abstracts*, *sentences* and *tokens*.

Analogously to the evaluation corpus for yeast, we created corpora for other model organisms, *i.e.* *Escherichia coli*, *Bacillus subtilis*, and *Mus musculus*, in a later stage. These were extracted by looking for both the full and abbreviated genus name (*e.g.* *E. coli*). In the case of *M. musculus* we further checked for occurrences of the words “mouse” and “mice”. The details for each corpus are presented in table 6.4 as well. Note that categorising the PubMed corpus according to the presented criteria can lead to incomplete classifications. This is the case since not all organisms that are mentioned within each article are also to be found within the list of MeSH terms that have been used for the annotation of the abstract.

In order to test the extraction rules on full-text articles as well, the open-access part of PubMed Central was downloaded on March 16, 2004. Again, details on the corpus size are described within table 6.4. For the preliminary tests presented here, we did not separate between different sections of a paper (like “Introduction and Motivation”, “Materials and Methods”, etc.). Although it is case that the introduction section of a paper tends to list many established facts (in contrast to for example the results section) as reported in Shah *et al.* (2003).

## 6.2 Tokenisation influences parsing

Roughly speaking, tokenisation is about identification of word and sentence boundaries. According to Grefenstette and Tapanainen (1994) the process of tokenisation consists of two steps. A first step concerns the segmentation of the input text into a sequence of tokens, the detection of word boundaries. The second step is about the detection

of sentential boundaries. When reading a given text it is clear (for the human reader) which punctuation mark identifies a sentence boundary. However, the same does not hold for word boundaries in English. As an example, is “bus station” one word or a term consisting of the two words “bus” and “station”? The succeeding section 6.2.1 goes into the details of these two tokenisation steps (*i.e.* word and sentence segmentation). It shows common issues that have to be addressed within this processing step. The following section 6.2.2 shows that tokenisation, part-of-speech tagging and parsing are highly interrelated.

I decided for our information extraction experiment to use a tokeniser which was developed and implemented by Helmut Schmid (Schmid (2000)) at IMS (University of Stuttgart). This decision was guided by mainly two reasons. One is that the tokeniser achieves a high accuracy for the identification of sentence boundaries. In Schmid (2000) the author reports an accuracy of 99.56% on the Brown corpus (Francis and Kucera (1979)). The second reason is based on the fact that the tokeniser acquires his knowledge for this decision by unsupervised learning. This means that for the learning procedure (*i.e.* the generation of a parameter file) no corpus with manually labelled sentence boundaries is needed. As can be seen from table 6.4 PubMed offers enough data for the training procedure.

### 6.2.1 Tokenisation

In natural languages like English or German a period can have different functions at the end of a word. It can either mark ordinal numbers (an example is *the 2. Workshop on e-Learning*), it can be part of an abbreviation (for example in *Dr. Jekyll and Mr. Hyde*) or it can serve both as sentence marker and be part of an abbreviation (*e.g. The product was delivered by Pioneer Shipping Ltd.*). As already mentioned in section 3.1, abbreviations are rather frequent in scientific literature and it turned out that to distinguish between abbreviations and sentence boundaries was a main challenge for the tokeniser to be met. To achieve a high and organism independent accuracy the punctuation cutoff was trained on approximately  $10^6$  PubMed abstracts ( $10^8$  tokens) selected randomly from the complete PubMed corpus.

The second problem, the determination of token boundaries<sup>3</sup> in technical or scientific texts is a major challenge for information extraction or information retrieval systems. On the one hand, technical terms contain special characters (like brackets, colons, hyphens, slashes, etc.). On the other hand to prevent overtokenisation (the term overtokenisation is explained with the succeeding subsection in detail) multiword expressions have to be identified as a coherent sequence of words. Technical terms (like *cyclin dependent kinase*) occur rather frequent within scientific literature like PubMed abstracts. The main problem is to determine the left and the right boundaries of the multiword expressions. Examples like (6.2) show that two technical terms (for example "CDC2-related kinase" and "PITALRE") can succeed each other (appositions are already mentioned in section 3.1.3) thus making it even harder to detected the boundaries.

---

<sup>3</sup>It should be noted that I regard detection of term boundaries or multiwords as part of the tokenisation step.

(6.2) The CDC2-related kinase PITALRE is the catalytic subunit of active multimeric protein complexes.

Although a lot of work has been invested in the detection of technical terms within biology related texts (see Nenadić *et al.* (2003) or Yamamoto *et al.* (2003) for representative results) this task is not yet solved to a satisfying extent. As I was interested in a fixed set of terms and high precision results were of importance I opted for a dictionary-based multiword detection. The acquisition of the dictionary was based on semi-automatic methods. The dictionary and the methods are described in sections 6.4 and 6.5. To put it in a nutshell: The basic idea was to collect a series of multiword terms, partly from databases, partly in a manual fashion, and to join the entries in a multiword dictionary. This dictionary was expanded with orthographic variations (described in 6.4) and then used with simple string matching techniques for the multiword recognition within the tokenisation process.

## 6.2.2 (Over-)tokenisation and parsing

This subsection shows the influence of tokenisation, especially the multiword detection on the succeeding processing steps of part-of-speech tagging, syntactic analysis and the semantic construction. The following demonstrates these inter-dependencies by going through the example (6.3). This example contains a series of multiwords. These are used to illustrate the effects that different multiword segmentation has on syntactic analysis and semantic construction. For matters of presentation the example consists of a nominal phrase only.

(6.3) ... the putative gene *saccharomyces cerevisiae* riboflavin synthase beta chain.

To be able to better understand the given example I'll explain roughly the biological background of the mentioned technical terms:

- *Saccharomyces cerevisiae* is the official Latin name used in scientific publications for the yeast organism.
- The official recommended enzyme name *riboflavin synthase*<sup>4</sup> is used to refer to the enzyme with EC number 2.5.1.9. It should be noted that usually proteins or complexes of proteins act as enzymes.
- The expression *beta chain* denotes a subunit of a protein complex (in this case here of the given enzyme). Thus *riboflavin synthase beta chain* refers to beta subunit of the enzyme with the official EC number 2.5.1.9.

I present in what follows three different types of tokenising this input string. For each of these three tokenisation possibilities I show the corresponding syntactic analysis is

---

<sup>4</sup>The official systematic name is *6,7-dimethyl-8-(1-D-ribityl)lumazine:6,7-dimethyl-8-(1-D-ribityl)lumazine 2,3-butanediyltransferase*.

shown. These syntactic tree structures are each shown in table 6.5, table 6.6 and table 6.7.

The first possible analysis is shown in table (6.5) which depicts a syntactic analysis deriving from what I call overtokenisation. This analysis follows the one-string-one-token hypothesis. This means that each blank or punctuation mark acts as a word boundary. The consequence from this approach is that more work has to be invested within the succeeding steps of parsing and especially in the step of the semantic construction, where the meaning has to be computed from the parts of the sentence and their syntactic composition. For each of the tokens within the noun chunk (**nx**) the relation to the other tokens has to be computed in order to determine their relation and thus the meaning of the whole word. In a scenario like the previously described information extraction scenario it is not desired to determine the meaning of *saccharomyces cerevisiae* in a compositional way (*i.e.* from the words composing the whole word), for example that *cerevisiae* originates from the Latin *cerevisia* and thus denotes beer. For extracting gene expression data it is enough to recognise the organism names within the abstracts and to map them to a unique identifier. Without lexical information – like *saccharomyces cerevisiae* being a multiword – a chunker might easily group the sequence of nouns together with the *sym* POS-annotated token to one huge noun phrase neglecting the internal structure.

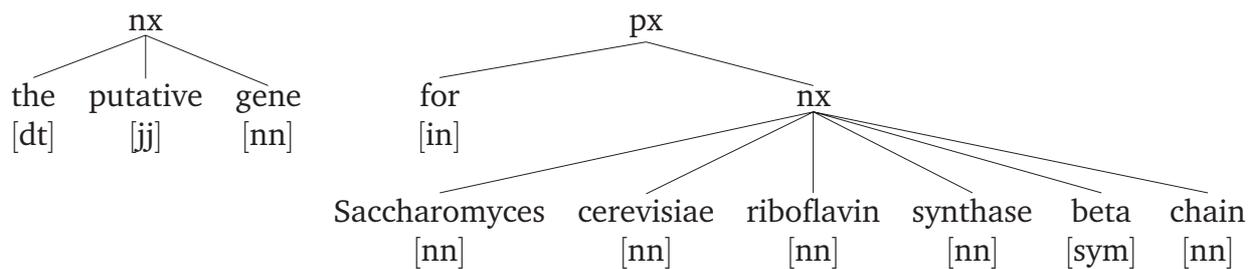


Table 6.5: **Flat representation** of a noun chunk analysis where a noun chunk is succeeded by a prepositional chunk. It should be noted that no multiwords have been recognised during the preprocessing and thus the noun chunk within the prepositional chunk has a flat internal structure.

The second tree, which is shown in table 6.6, derives from a tokenisation where (i) the organism name is tokenised as one coherent token and (ii) the enzyme name is recognised as one multiword entity as well. Technically this can be achieved through a dictionary of enzyme and organism names, which are both available through various sources like the BRENDA database for enzyme names (Schomburg *et al.* (2002)) and the NCBI taxonomy for organism names (Wheeler *et al.* (2004)). It needs to be mentioned that these dictionaries are not (and can never be) complete. A manual extension and curation of this dictionary is recommended. For the recognition of multiwords a longest match approach is necessary since parts of multiword terms can be multiword terms as well. More details on our approach are described in section 6.5, *Semantic Tagging*.

The third possibility for processing multiwords is shown in table 6.7. It is a variant of the second analysis, where a more syntactic structure is added. It can be described

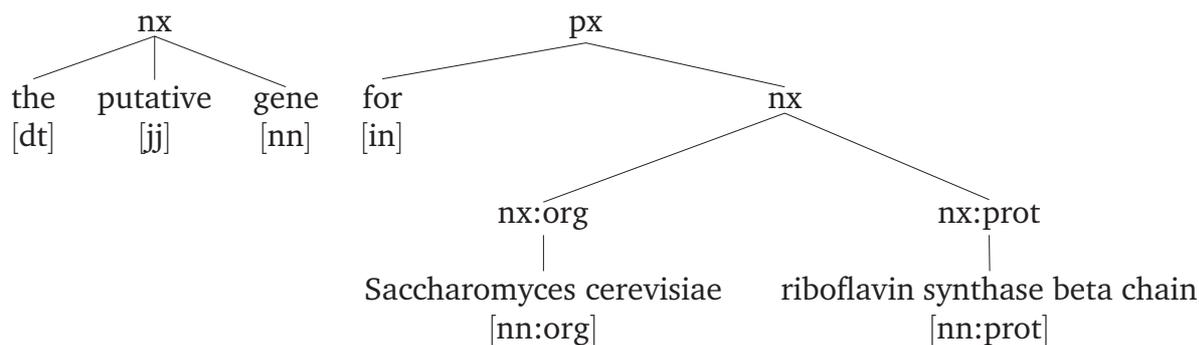


Table 6.6: The picture shows a syntactic tree for a noun chunk with a prepositional chunk attached, where the noun chunk part of the prepositional chunk is composed of two succeeding multiword terms with both multiwords recognised. Note that each part-of-speech tag is followed by a colon and a semantic label.

as finer grained with respect to the compositional structure of the enzyme name. Given that a lexicon provides the necessary entries it would be possible to distinguish between the subunits like *alpha chain* and *beta chain*. Depending on the dictionary, other subunits of such complexes could be identified analogously. This analysis provides more information as the term *riboflavin synthase* is recognised as referring to an enzyme name (with EC number 2.5.1.9) and *beta chain* specifies that the enzyme is a complex and this expression refers to the *beta chain* part of it.

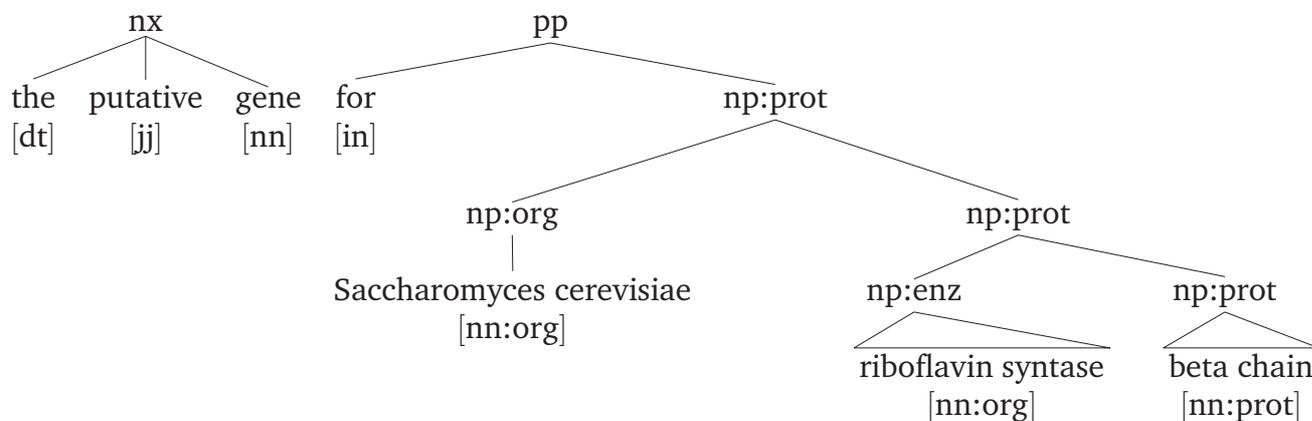


Table 6.7: **Deep structure:** Syntactic tree representation showing three succeeding multiword terms with the relations between them. All three multiwords recognised within the prepositional chunk.

In principle all three analysis are reasonable always depending on the granularity of the information that the system should produce. As already mentioned we opted for the second analysis within our IE system. This made it necessary to have a multiword recognition component in our system to treat for example organism names or protein/gene names. Details on that are described in the section 6.5 *Semantic tagging*.

## 6.3 Part-of-Speech Tagging

According to the system architecture that is shown in table 6.3 the next processing step on word level is part-of-speech (POS) tagging.<sup>5</sup> POS-tagging is the process of labelling each token with an appropriate part-of-speech. At the moment there are various POS-taggers available. A general distinction about tagging approaches which can be made concerns the degree of automation of the tagger training and tagging process itself. As for the training procedure a distinction between *supervised* and *unsupervised* learning approach can be made. Using supervised taggers means training a tagger (*i.e.* calculating POS-tag sequence probabilities) on a manually POS-tagged corpus and combine that with additional information like a tagger lexicon. In contrast to that unsupervised models do not require a (labour intensive manually) annotated/pre-tagged corpus but use sophisticated computational methods to automatically induce word and POS-tag groupings. Another unsupervised method is a rule-based approach. Hand-crafted rules that consider left and right POS context determine POS tags for the next word in case it is ambiguous. As an example a rule might say: if a determiner is followed by an ambiguous word (*i.e.* possibly being an adjective or an adverb) and this one is followed by a noun, then the ambiguous word is annotated as an adjective. An implementation and its application is described within Brill (1992).

Within the information extraction system I decided to use the TreeTagger developed by Helmut Schmid (see Schmid (1994)). This tagger estimates transition probabilities based on a Markow model in combination with decision trees that was trained on labelled data. The reason for this decision is straightforward. According to Coden *et al.* (2004) a large annotated common English corpus is not sufficient for building a tagger model which is adequate for tagging documents from the medical domain. By adding a rather small domain-specific corpus to a large common English corpus the performance rises from 87% to over 92% accuracy. Luckily, the GENIA project – which is hosted by the University of Tokyo<sup>6</sup> – offers a manually POS-tagged set of Medline articles. Although some adaptations – which are reported further down – had to be made, it was a comfortable approach promising high precision results. The TreeTagger reached after training on the GENIA corpus an accuracy of 96.36% on test data.

Two resources have to be provided for successful use of the TreeTagger: (i) a lexicon (ii) a tagset that contains all possible POS-tags. Ad (i): The lexicon contains frequent tokens. Each of them has to come with a list of possibly assignable part-of-speech tags and the corresponding word stems. To give an example, the entry for *age* has to list that this word can occur as a noun (*i.e.* *NN*), as a verbal infinitive *VV* etc. Concerning (ii) I used the Penn Treebank tagset, which is described in Santorini (1990) in detail, as underlying tagset. Underlying means that some minor adaptations have been carried

---

<sup>5</sup>It has to be noted that frequently stemming is carried out within the part-of-speech processing step. The stem information is used within the POS tagging process, as for example the stem can already tell whether the word is used as a nominal or verbal derivate. In the following I neglect this information because we did not need to use the stems for our information extraction system.

<sup>6</sup>Details on the GENIA corpus and the corpus itself are available through: <http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/>.

Token	POS-tag	Stem
RME1	nn	RME1
is	bez	be
repressed	vvn	repress
by	by	by
a	dt-a	a
complex	nn	complex
of	of	of
MATa1	nn	<unknown>
and	cc	and
MAT	nn	MAT
alpha	sym	alpha
2	cd	2
gene	nn	gene
products	nns	product
.	sent	.

Table 6.8: **Part-of-speech tagging:** An example showing the output of the TreeTagger with three columns. The first column shows one token per line with the corresponding POS-tag in the second column and the stem information in the third column.

out. These are reported further down. To train and analyse the improvement of the accuracy of POS-tagging on PubMed abstracts, TreeTagger was trained on three different corpora. The first experiment was carried out using a parameter file generated by training on a common English corpus with a common English dictionary. The parameter file has been generated by training on the Penn Treebank. The second experiment was carried out using a parameter file generated by training on GENIA 3.0 corpus (details are described in Kim *et al.* (2003)) with a dictionary for common English. The third experiment was conducted using a parameter file generated by training on GENIA 3.0 corpus with extensions to the tagger lexicon relevant for the PubMed abstracts. These extensions include mainly gene and protein names and other cue words, both including multiwords. The sections 6.4 (on detecting gene and protein names) and 6.5 (on semantic tagging) describe how large lists of relevant nouns were generated. These lists were fed into the tagger lexicon in the third experiment.

For the experiments with the GENIA 3.0 corpus – it consists of PubMed abstracts and has 466,179 manually annotated tokens – I have adapted the POS tagset and corrected some POS tags in the corpus. Of course, one might object that by changing a free online-available corpus it reduces the chances to compare results between different systems. However since some of the changes were inevitable (due to tagging errors for example or due to incompatibility with our tagset) I have adapted the tags in such a way that it met all requirements for the further processing.<sup>7</sup> In addition, the main goal was not to

<sup>7</sup>The adapted corpus is made available upon request.

set up a system that achieves best POS-tagging results on one particular corpus but to extract information with a certain quality for a certain purpose. Thus the tagger had to perform in an optimal way for the succeeding components. Three types of revisions or corrections have been carried out:

1. The first type of changes was responsible for removing inconsistencies and POS-tags that had not been assigned correctly. To give an example, a seemingly undecidable case is the POS-tag for *acetyl* in *a chloramphenicol acetyl transferase* was annotated with `jj|nn`. I corrected it to `nn` since an adjective can not occur between two nouns. I would judge this ambiguity rather as false – in a linguistic sense – than as undecidable.  
More obvious examples were: the token *in* was annotated in some occurrences as a determiner (tagged as `dt`), which has been corrected to `in` (*i.e.* the general tag for prepositions). In other cases the same token was annotated as `nn`. Other examples include the token *were*, which was annotated in one case as `nn`. The annotation was changed to `vbd`. All in all more than 300 occurrences were identified to be falsely annotated and corrected.
2. The second set of changes was applied to seemingly undecidable cases which were disambiguated. As an example the token *in/or* was annotated with two POS-tags, *i.e.* `in|cc`. As TreeTagger annotates only one tag per token the simplest solution was to split this token into three tokens, *i.e.* *in*, */*, and *or*. Each of them was annotated with its own tag, `in`, `:`, and `cc` respectively. In contrast to that I did not decompose *+/-* into three tokens. I treated this as one mathematical symbol. All in all more than 110 previously undecidable cases have been disambiguated or corrected.
3. The refinement of the GENIA tagset was the third type of revisions. It concerned auxiliary verbs and greek letters. For different auxiliary verbs we used different POS-tags. Thus we distinguished between *be* verbs – annotated in the new version as `vb...` –, *have* verbs – annotated in the new version as `vh...` –, and other verbs – annotated as `vv...` in the new version –. Other adaptations comprise: the POS-tag for *+/-* was changed from `cc` to `sym`, */* was changed from `/` to `:`, *same for* – which was changed from `-` to `:`, and *=* was annotated as `sym` instead of `jj`. Sentence markers *.* were annotated as `sent`. Additionally, some POS-tags were simply changed wrt. their naming, *i.e.* `np` instead of `nnp`, `PP` instead of `prp`, and `pp$` instead of `prp$`. Some POS-tags occurred only once or twice in the whole corpus. Obviously these were typing errors, like `xt` instead of `dt`, `pp` instead of `our/PRP$` for the token *ours*, or `n` instead of `nn` for the token *CD80* (a protein name).

All in all more than 14.000 changes to the tagset have been made. The results that have been achieved (yielding different parameter files) are described in the evaluation section 7.3.

## 6.4 Recognising Gene and Protein Names

Within the molecular biology domain names of genes, proteins, chemical substances and the like are of crucial importance. As already mentioned in section 3.1 there exist several approaches for automatic detection of biomedical technical terms. I already cited Collier *et al.* (2002) and Nenadić *et al.* (2003). Both pursue an approach that generally aims at detecting terminological expressions in biomedical texts and classifying them according to a pre-defined structure. This is a less specific but a fairly broad approach. The results are good compared to the bandwidth of the task but are not enough for a highly specific question that the already described scenario of this work has. In contrast to that other approaches have more specific questions guiding their terminological work. For the detection of protein names in biomedical publications for example there are several systems available. The probably three most promising are the Kex system (see Fukuda *et al.* (1998) for details), the Yapex system (details are reported in Franzén *et al.* (2002)) and the Yamamoto approach (according to Yamamoto *et al.* (2003)). Kex and Yapex are both based on a hand-crafted rule-based approach. In contrast to that Yamamoto is a statistical approach based on a morphological analysis composing the morphemes to chunks.

All three systems were evaluated on the Yapex corpus. It consists of 200 Medline abstracts. 99 of these were drawn randomly to form a reference or training corpus. The corpus used for testing consists of 101 Medline abstracts. These were annotated by domain experts. The comparison of the results for the so-called *strict* evaluation shows that the Yamamoto system has the best results with a precision of 73,8% and a f-score of 63,5%, Yapex reached – depending on which reference you choose – up to 67,8% precision and 67,1% f-score, Kex reached 40,4% and 40,7% respectively.

Although these results are promising we decided not to follow any of these approaches. Mainly three reasons were responsible for this decision. First, it is not obvious that comparable results can be reached for gene name recognition without too much effort. Second, the precision rates are still not high enough compared with the MUC scenario to guarantee good results for the following system components. And last, the task in our application seemed to be more specific than the one that has been chosen for the evaluation. We were not interested in any protein or gene name but in particular names, *i.e.* we were interested mainly in recognising protein names and recognising gene names separately. In addition these names should be recognised for the four mentioned model organisms.

We have chosen to follow a dictionary-based approach together with some hand-crafted rules. To be able to recognise gene/protein names for the four organisms as such, and to map them on to the appropriate database identifiers<sup>8</sup>, a list of synonymous names and identifiers for six selected eucaryotic model organisms was compiled from several sources<sup>9</sup>. These sources comprise names and identifiers from SWISS-PROT (Boeckmann *et al.* (2003), which has the largest collection of protein and gene names),

---

<sup>8</sup>This task is sometimes referred to as normalisation of protein/gene names.

<sup>9</sup>These lists are available through <http://www.bork.embl.de/synonyms/>. Note that the versions change through time since the databases are updated and the lists are updated as well.

supplemented by names for yeast from the Saccharomyces Genome Database (SGD) (Dwight *et al.* (2002)). Additionally, WormBase (Stein *et al.* (2001)) and FlyBase (The FlyBase Consortium (2003)) were consulted. The numbers on the gene and protein names can be found in table 6.9. At the moment of writing this thesis of a yeast gene or protein had more than 8 synonymous names in average.

Organism	NCBI ID	Unique gene names	Expanded gene names
<i>S. cerevisiae</i>	4932	51.640	518.252
<i>M. musculus</i>	10090	134.646	540.694
<i>B. subtilis</i>	1423	20.014	519.939
<i>E. coliK12</i>	83333	41.057	524.155

Table 6.9: The table presents numbers on the organism specific protein and gene names that were use for our system. The first column mentions the organism. The second column names the corresponding unique NCBI ID. The third column shows the numbers indicating the number of protein and gene names extracted from databases. The last column indicates the amount of generated orthographic variants. This expanded list was used for string matching as described in .

Before matching these names against the POS-annotated corpus, the list of names was expanded to include different orthographic variants of each name. Firstly, the names were allowed to have various combinations of uppercase and lowercase letters: all uppercase, all lowercase, first letter uppercase, and (for multiword names) first letter of each word uppercase. From each of these variants new variants were generated by replacing whitespaces by hyphens and the other way round. In addition, from each gene name a possible protein name was generated by appending the letter p. The following strings within the generated dictionary remained unchanged, since they are commonly used case sensitive: ssDNA, cDNA, mtDNA, hDNA, rRNA, mRNA, tRNA, RNase. A change that has been carried out on the corpus was to replace tokens within the corpus containing a '/' by a whitespace. It is common usage to refer to the compositional nature of protein complexes by concatenating the participating proteins by '/'. As an example *HAP2/HAP3/HAP4 protein complex* refers to the protein complex consisting of *HAP2*, *HAP3* and *HAP4*. All in all the mentioned orthographic variations expanded the name lists roughly by three to ten-fold. The table 6.9 shows for all four organisms the amount of uniquely resolvable names<sup>10</sup> and the amount of expanded names.

The orthographically expanded name list was fed into the the POS-tagger lexicon. And, those consisting of more than one word were fed into the multiword recognition. In a subsequent step the list was run against the POS-tagged corpus to retag gene/protein names as such. The new tag (*i.e.* nnp<sub>g</sub>) was a combination of its part-of-speech *nn* and its semantic property assuming to be either protein *p* or a gene *g*. At this level it was not possible to distinguish between genes and proteins. This is due to the already mentioned problem of a protein sometimes having the same name as the corresponding gene. To

<sup>10</sup>This refers to gene/protein names that can be mapped to a unique identifier in a database.

reduce the problem of homonymy, where a protein or gene name is the same as a verb or conjunction, only matches to words tagged as common nouns (nn) were accepted.

## 6.5 Semantic Tagging

The step of semantic tagging is usually carried out on a part-of-speech tagged corpus. Its main objective is to determine a word sense in a specific context. It is common to reduce this task to the annotation of words with one (or more) semantic tag(s) out of a pre-defined set of tags. This step is frequently referred to as semantic labelling, too. The previously mentioned recognition of protein and gene names already means a semantic labelling. Additionally, it was necessary for our approach to tag several other semantically relevant terms, too. In particular these are nouns and verbs, some prepositions and some adjectives. Out of each of these syntactic categories a relevant set of terms was classified and annotated. In the following I will go through each of these syntactic categories to present the semantic classes that were associated with these terms.

It has to be mentioned that instead of annotating the semantic label additionally to the part-of-speech tag we replaced the part-of-speech tag. The main reason is that within the following processing step we used only one feature for recognising entities and relations. It would be easy to change the annotation such that both the POS-tag and the semantic label are available in case needed. Nonetheless, the arguments for this decision will become even clearer after going through the following subsections, where the extraction rules are explained in detail.

Within the semantic labelling step three categories of words were semantically annotated. The first main group of terms are nouns, the second class are verbs and the third class are adjectives and prepositions. The nouns are annotated and classified as follows:

- A first class of nouns terms represent generally relevant concepts. These comprise mainly the concepts shown in Figure 4.2. Some of the terms have each its own semantic tag, others could be clustered and synonymous terms bear the same label. The term *gene* is thus annotated as *gene* instead of *nn*. Other nominal terms that were treated analogously comprise: *protein*, *promoter*, *binding site*, *transcription factor*, etc. Groups of nominal terms that were annotated with the same semantic tag are for example: *activating protein*, *activator*, *positive activator*, *transactivator*. All in all there were 153 nominal entries in this part of the nominal lexicon.
- A second class of nominal terms that are semantically annotated as nouns triggering an experimental or artificial context. As an example artificial context comprise artificially changed genetic material. This class comprises nouns like *mutation*, *deletion*, *fusion*, *defect*. All were annotated with the same semantic tag *mution*, abbreviating the term *mutation*. This class has 11 distinct entries.
- A third class of nominal terms that were semantically labelled are names referring to enzymes or enzymatic activities like *elongase*, *hexokinase*, etc. The set of terms

belonging to this class could easily be extracted from the corpora by looking for the suffix *ase* and some manual inspection of the list, which comprised 569 entries.

- The fourth class of terms are species and organism names. A basic part was extracted from the NCBI taxonomy of organisms Wheeler *et al.* (2004) (20,746 entries), which lists the common names, alternative names and synonyms. Additionally, through manual inspection (by using CQP<sup>11</sup>) variants that were not within the taxonomy were extracted from the corpora. These mainly include orthographic and paragrammatical variants as described in the chapter before.
- The fifth class of nominal terms are relational nouns, mainly nominalisations. This class deals with nouns which express relational information between entities and can be treated similar to verbs. We thus have three classes, (i) nouns of activation, (ii) nouns of repression, and (iii) nouns of regulation.
  - i . Nouns of activation are *derepression, positive regulation, enhancement, activating, increasing* etc.
  - ii . Nouns of repression are *suppression, blocking, negatively regulating and negative regulation, suppression* etc.
  - iii . Nouns of regulation, which are neutral with respect to up- or down-regulation comprise examples like *affecting, control, controlling, regulating* etc.

All in all there were 69 entries in the class of relational nouns labelled with one of these three labels.

The set of verbs contains 50 entries plus their corresponding inflectional variants<sup>12</sup>. Verbs are crucial for the extraction of relations between entities. The classes I present for the verbs are analogous to the relational nouns. Verbs behave different with respect to their syntactic valence (, or subcategorisation) why they receive a different tag than the nouns. This is necessary to be able to treat them different in the following relation detection components. The following shows the classification according to the type of interaction with respect to gene transcription:

- The first class comprises verbs of activation, for example *enhance, increase, induce, and positively regulate*. All these verbs (and their inflectional variants) were annotated with the same semantic tag, *i.e.* actv.
- The second class consists of verbs of repression, *e.g.* *block, decrease, downregulate, and down regulate*. The corresponding semantic tag is inhv.
- The verbs of regulation like *affect* and *control* were annotated with the semantic tag regv.

---

<sup>11</sup>Details about CQP and an example using scenario are shown in the appendix .

<sup>12</sup>For a verb like *activate* the additional inflectional variants are *activated* and *activates*.

- Other selected verbs that are relevant for gene expression like *code* (or *encode*) and *contain* each received its own semantic tag. Further verbs that were also annotated – each with a separate semantic tag – comprise *cause*, *contain*, *detect*, *display*, *find*, *reveal* etc.

The remaining other categories consist of (i) prepositions, (ii) adjectives and adverbs, and (iii) words expressing negation.

- To be able to distinguish the semantic role of some entities it was necessary to distinguish the prepositions on the level of semantic labelling, which is not possible on the level of POS-tags, since all prepositions get annotated as *in*. As an example it makes a difference between *binding site for* and *binding site of* (more on that will follow within the succeeding sections). The prepositions that were semantically labelled comprise *for*, *from*, *in*, *with*, *by*.
- Although most adjectives and adverbs were not distinguished some were labelled semantically. Among them are examples like *in vivo*, *in vitro*, *pre*, *not only*, *on the other hand*, etc.
- Negations are obviously important to detect false positive interactions. False positive interactions are interactions that are reported to not take place. This class comprises *not*, *nor*, *neither*, *cannot*, *unable* etc.

## 6.6 Recognising Entities and Relations with CASS grammars

In the preceding sections I describe the segmentation of the corpora into tokens, the annotation with part-of-speech tags as well as the annotation of these tokens with semantic labels. The next processing step changes now from word level to phrasal or sentential level, *i.e.* the step of syntactic analysis. At this point a decision had to be made whether to chose a chunking approach or a full parsing approach. In general it can be said that full parsing is expensive, and less robust. In contrast chunking (also called partial parsing) is much faster, more robust, and is frequently sufficient for many applications like information extraction or Question/Answering. But there have been two additional reasons to chose the chunking approach. First, an existing chunk grammar is usually easier adapted to new corpora and new domains than a full-parser grammar is. At the moment of writing this thesis full parsers have a limited coverage far beyond that of chunkers, which is mainly due to the massive amount of technical terminology within PubMed. The second argument for choosing the chunking approach was Steven Abney's CASS chunker Abney (1996) that offers the possibility of combining syntactic and semantic features. This possibility saved us from building a separate semantic construction on top of the syntactic analysis. The main drawback of this approach is the combinatorial explosion of rules. Although this item will become clearer within the following sections

it should be noted that each semantic feature was built in the syntactic rule, which lead to the mentioned explosion of rules.

On the other hand the templates that had to be filled by this system were very limited. This also makes clear why there was no need for a grammar with a broad coverage and semantic features for all possible phenomena. The questions to be answered made it possible to follow this approach with manageable efforts. This combination of syntactic and semantic rules, or simply syntacto-semantic chunking, was implemented as a CASS grammar. The main properties of a CASS grammar can be characterised as follows:

- A CASS grammar consists of a series of finite-state automata (or, transducers), that operate on different levels arranged in cascades. Thus the output of one level serves as input for the succeeding level.
- The transducers are coded as regular expression grammars, with a left-hand side and a right-hand side working on the output of the previous levels.
- Level 0 takes the part-of-speech tag sequence as input. In our case, the semantic labels, which have replaced the POS-tags, or the POS-tags, that have not been replaced are the features which are used as input.
- If the input from one level *i*, *i.e.* a series of POS-tags and semantic labels, match the pattern on the right-hand side of a rule, then these are chunked and annotated with the label specified on the left-hand side. Thus a new level of annotation is generated, which can serve as input for the next level.
- Any input sequence that does not match a rule is passed to the next level without being modified.

To illustrate how the CASS chunker works table 6.10 shows on its left-hand side the output for example (6.4). In contrast to that on its left-hand side the same sentence is shown after being processed by our system.

(6.4) Binding of Sp1 and Sp3 transcription factors to the Oct-4 gene promoter.

As already mentioned the input for level L0 for the CASS chunker are the POS-tags. In table 6.10 the tokens are in the right-most position in brackets with the corresponding POS-tags as superscripts of the corresponding opening brackets. As an example the POS-tag for *and* is *cc* and for *Sp1* it is *nn*. To demonstrate how the rules and the cascaded architecture work I chose three rules, *i.e.* (6.5), (6.6), and (6.7). Example (6.5) shows the first level (*i.e.* Level 0) and examples (6.6) and (6.7) shows the second level (Level 1).

### Level 0

(6.5)  $nx \rightarrow dt? (jj | rb)^* h=(nn|nns)+;$

## Level 1

(6.6)  $ng \rightarrow h=nx \text{ (of } nx \text{ (of } nx \text{)}^*)? \text{ (cc } nx \text{)}?;$

(6.7)  $pp \rightarrow \text{(to | in) } h=nx;$

The rule in example (6.5) can be interpreted as follows. Collect a potential determiner (*i.e.*  $dt?$  represents either one or no  $dt$ ), followed by a possibly empty set of adjectives or adverbs (*i.e.*  $(jj | rb)^*$ , where  $|$  represents a disjunction and  $*$  the Kleene star), followed by a series of singular or plural nouns (*i.e.*  $(nn | nns)^+$ , marked as head of the constituent *i.e.*  $h=$ ) and group it as a noun chunk (*i.e.*  $nx$ ). All other sequences of POS annotated tokens are passed unchanged to the next level L1. Thus, applying this rule to example in table 6.10  $[_{to} \textit{to}]$  or  $[_{cc} \textit{and}]$  both remain unchanged.

Within the second Level L1 two rules within the same cascade are shown in example (6.6) and example (6.7). The rule in (6.6) collects a noun chunk (*i.e.* the  $nx$  from the previous level) which might be succeeded by a prepositional phrase where the prepositional phrase has *of* as the preposition attaching a noun chunk. This noun chunk can be followed by a conjunct with an addition noun chunk (*i.e.*  $(cc \ nx)?$ ). The whole chunk is annotated as a noun group (*i.e.*  $ng$ ). This tag is passed to the next level L2. Analogously, the rule in example (6.7) chunks the preposition (*i.e.*  $(to | in)$ ) and a succeeding noun chunk (*i.e.*  $nx$ ) as a prepositional phrase (*i.e.*  $pp$ ) passing it to the next level L2, too.

Since no other rules apply the analysis remains unchanged.<sup>13</sup> This stage of analysis already shows common problems with this approach. To determine from this structure that both  $Sp1$  and  $Sp3$  are transcription factors one has to reanalyse the noun group *Binding of Sp1 and Sp3 transcription factors* to recognise the semantic type of each  $Sp1$  and  $Sp3$ .

In contrast to that the right-hand side of table 6.10 shows for the same example (6.4) the output that is generated by our set of rules applied to the semantically labelled and POS-tagged input. On level L0 the tokens are either annotated with the appropriate POS-tag. For example  $dt$  marks the word *the* as being a determiner and the semantic label  $nnp_g$  is assigned for the tokens denoting a gene or protein name. The details of the grammar on entity level and on relational level will be described in the following subsections 6.6.1 and 6.6.2. But, without knowing the details of the grammar it is easy to recognise that much more structure is annotated in the syntacto-semantic chunking output. Additionally the structure is still understandable with some explanation. For example  $nxtf$  marks a noun chunk denoting one or more transcription factors. Or,  $nxprom$  refers to a noun chunk denoting a promoter. The extraction component has to extract that each  $Sp1$  and  $Sp3$  bind to the *Oct-4* promoter and thus regulate (*i.e.*  $ev\_reg\_expr\_rx$ ) the *Oct-4* gene.

Although the number of rules increases drastically it is less work intensive than applying a separate component to re-analyse a chunked output and interpret it semantically. Comparing the amount of levels or cascades implemented in the e8.reg grammar

---

<sup>13</sup>These rules are a reduced version from the CASS e8.reg grammar, which is within the CASS software package.



changes have to be made. It is simply not easy to oversee the consequences in such cascaded approach. Bug fixing as well as improvement of performance forced us to change and adapt the grammar steadily. And in the end it is still incomplete and produces still errors. Thus it is always possible to add rules that recognise a yet unanalysed or not correct analysed syntactic constructions. The grammar that I present in the following sections is thus a trade-off between the time invested and the quality of results that the grammar produced on the chosen corpus.

One version of the grammar is to be found within the appendix section VI. The sections B.1 to B.30 each shown one of the 30 levels of the grammar.

### 6.6.1 Recognising Entities

This section describes the cascaded set of rules for the detection of noun phrases which are relevant for gene expression. In some way it is comparable to named entity extraction that was carried out within the Message Understanding Conferences (see section 2.2.1, *The message understanding conferences* for details). In comparison the amount of different types of entities has been smaller within the MUC named entities task than in this experiment. I present in what follows the named-entities rules by giving examples for every semantic type of entity that was implemented. The examples show the annotation which is a result of the cascaded rules. Each example shows a bracketed structure with categories or labels annotated through subscripts. The whole set of rules can be found in the appendix section VI in section B.4.

The entities have been grouped as follows. The first package shows bracketed structures for entities denoting proteins and related entities, *i.e. protein, transcription factor, gene product, parts of protein* and the like. The second package presents the bracketed structures for genes and gene related entities like *gene, promoter, upstream activating sequence* etc. The third part reports on ambiguities between proteins or genes. The last part reports about other protein and gene related entities that consists at the moment of *RNA* only.

To give an overview of the amount of detected named entities that in the corpus a block chart was included, see figure 6.1. It shows all named entities with a frequency of occurrence of more than 100.

#### Recognising proteins and related entities

- **Protein names – nxprot:** One of the most important entities within gene expression relations are the protein related noun phrases since proteins act as agents in gene expression. As all other now following named-entity patterns they are based on the syntacto-semantic tag *nnp<sub>g</sub>*.

1. A bracketed structure that shows a *nnp<sub>g</sub>* followed by the head noun *protein* and thus makes makes clear that *nnp<sub>g</sub>* refers to a protein.

```

[nxprot
  [nxpg
    [nnpqx
      [nnpq ARL-6]]]
  [prot protein]]

```

2. This pattern shows a twofold extension of the previous pattern. First, the *nnpq* is preceded by an organism name (*i.e. yeast*). Second, the whole noun phrase is followed by an abbreviation in brackets. There's additional information in the bracket after *the yeast POP2 protein*. It is important to recognise this bracketing as belonging to the noun chunk. Mainly this avoids blocking (or coming in between) the recognition of a gene expression relation, which can for example be followed by a gene expression interaction verb.

```

[nxprot
  [nxpg
    [dt The]
    [org yeast]
    [nnpqx
      [nnpq POP2]]]
  [prot protein]
  [( (]
  [nxpg
    [nnpqx
      [nnpq Pop2p]]]
  [) )]]

```

3. In this pattern here the fact that mentioned *nnpqs Cdc42* and *Rac* can be identified as being proteins rather than genes depends on the key word *GTPases*<sup>14</sup>, which is semantically labelled as *enzyme*. It should be clear that from the label *nxprot* the mentioned enzymatic function is not derivable. Other terms triggering the same rules are for example *kinase*, *synthase*, *lipase* as well as combinations like *protein kinase*. All in all there are 569 terms in this *enzyme* class.

<sup>14</sup>GTPases refers to a family of enzymes that can bind and hydrolyse GTP. GTP abbreviates *guanine 5'-triphosphate* and takes part in a set of energy-requiring processes.

```
[nxprot
  [enzyme GTPases ]
  [nxpg
    [nnpgx
      [nnpg Cdc42 ]]
    [cc and ]
      [nnpgx
        [nnpg Rac ]]]]
```

- **Transcription factors - *n<sub>x<sub>tf</sub></sub>***: As already introduced before transcription factors are proteins that steer the expression of genes. A few examples of different bracketed structure recognising *n<sub>npg</sub>*'s as being transcription factors are shown here:

1. This example shows a protein or gene name (*i.e.* *n<sub>npg</sub>*) preceded by the key word *transcription factor* thus the semantic type *tf* is assigned to the noun phrase. The corresponding rule can be found in the appendix section VI in section B.4.

```
[nxtf
  [dt The]
  [jj basic]
  [nm helix-loop-helix]
  [tf transcription factor]
  [nxpg
    [nnpgx
      [nnpg c-Myc]]]]]
```

2. This example shows that we also recognise plural mentions of *n<sub>npg</sub>*'s. This conjunct of two *n<sub>npg</sub>*'s is preceded by *transcription factor* assigning thus its semantic type. Since it is not decideable without taking larger context into consideration whether cumulative or distributive reading is intended we chose distributive reading for coordinated *n<sub>npg</sub>*'s. This means that each *Rel* and *RelA* are treated as seperately acting transcription factors.

```

[nxtf
  [dt the]
  [tf transcription factors]
  [nppg
    [nppgx
      [nppg Rel]]
    [cc and]
    [nppgx
      [nppg RelA]]]]]]

```

3. In this case here the term *transcription factor* occurs as subject complements – also commonly referred to as copula-construction – is recognised. Copula constructions occur within other noun phrase patterns as well and are used to identify the type of the *nppg*. Here, *nppg Cbfa1* is thus assigned the semantic type transcription factor.

```

[nxtf
  [nppg
    [nppgx
      [nppg Cbfa1]]]]
  [bez is]
  [dt an]
  [jj essential]
  [tf transcription factor]]

```

- **Transcriptional activator/repressor/regulator agents** - **nextractr\_ag, nxtr-repr\_ag, nxtrregr\_ag**: Transcriptional activators and repressors are specific types of transcription regulators. They're also called transcription factors. Transcriptional regulators are treated as being underspecified with respect to the type of regulation. The following bracketed structures show examples for agentive *transcriptional activators*, *repressors* and *regulators*.

1. The example shows an agentive transcriptional activator with the key word *activator of transcription*. Other key words for similar patterns are *transcriptional activator*, *positive transcriptional regulator* etc.

```

[nxtractr_ag
  [nxtractr_ag
    [tractr activator of transcription]
    [nxpg
      [nnpgx
        [nnpj protein 6]]]]
  [nxbrack
    [( (]
    [nxpg
      [nnpgx
        [nnpj STAT6]]]]
    [) )]]]

```

2. This example shows a noun chunk referring to an agentive transcriptional repressor. Other key words that are used as head nouns for this type of noun chunk are *negative transcriptional regulator*, *repressor of transcription* etc.

```

[nxtrrepr_ag
  [nxpg
    [nnpgx
      [nnpj TEL]]]]
  [bez is]
  [dt-a a]
  [jj sequence-specific]
  [trrepr transcriptional repressor]]

```

3. In this example I present a noun chunk referring to the agentive transcriptional regulator *Tat* with a series of nominal modifications that are accepted.

```

[nxtrregr_ag
  [dt The]
  [org human]
  [nn immunodeficiency]
  [nn virus]
  [nn type 1]
  [trregr transcriptional regulator]
  [nxpg
    [nnpgx
      [nnpj Tat]]]]]

```

- **Transcriptional activator/repressor/regulator patiens** - **nextractr\_pt**, **nxtrrepr\_pt**, **nxtregr\_pt**: Again, transcriptional regulators are detected. However the following bracketed structures show examples for patiens variants of *transcriptional activators*, *repressors* and *regulators*.

1. The example shows a patiens transcriptional activator. Other head nouns are *activator of transcription*, *positive transcriptional regulator*, *activator*, *activating protein*, *positive activator*, *transactivator* and *positive regulator*.

```
[nextractr_pt
  [dt-a a]
  [tractr transcriptional activator]
  [of of]
  [nxgene
    [nxpg
      [nnpqx
        [nnpq Cyp1a1]]]
    [gene gene]]]
```

2. Other patterns besides the one shown in the example (*i.e. suppressor of gene expression*) are *transcriptional repressor of nxgene*, *negative regulator of nxgene transcription*, etc.

```
[nxtrrepr_pt
  [dt-a a]
  [repr suppressor]
  [of of]
  [nxgene
    [nxpg
      [nnpqx
        [nnpq CYP1B1]]]
    [gene gene]]
  [expr expression]]]
```

3. Similarly, for the patiens regulators the bracketed structure looks as follows. As well as the head noun *regulator*, *transcriptional regulator* is accepted by the rules.

## System overview (Methods)

```
[nxtrregr_pt
  [dt-a a]
  [regr regulator]
    [of of]
    [nxgene
      [nxpg
        [nnpqx
          [nnpq SPI-1]]]
        [gene genes]]]]
```

- **Activators/repressors/regulators** - **nxactr\_ag**, **nxrepr\_ag**, **nxregr\_ag**: The following examples show other noun chunks which can act as transcription factors. This comprises agentive activators, repressors and regulators. As subjects of the appropriate verb-object tuples they are thus relevant for gene expression regulation.

1. This example shows a bracketed structure for the agentive activator protein *CDC20*.

```
[nxactr_ag
  [dt the]
  [jj mitotic]
  [actr activator]
  [nxpg
    [nnpqx
      [nnpq CDC20]]]]]
```

2. The example shows a bracketed structure for the agentive suppressor protein *p53*.

```
[nxrepr_ag
  [nxpg
    [dt the]
    [nnpqx
      [nnpq p53]]
    [nn tumor]]
  [repr suppressor]
  [prot protein]]]
```

3. A bracketed structure that recognises the agentive transcriptional regulator *Tat*.

```
[nxtrregr_ag
  [dt The]
  [org human]
  [nn immunodeficiency]
  [nn virus]
  [nn type 1]
  [trregr transcriptional regulator]
  [njpg
    [nnpgx
      [nnpg Tat]]]]]
```

- **Activators/repressors/regulators** - *nxactr\_pt*, *nxrepr\_pt*, *nxregr\_pt*: Similarly, as in the preceding examples the following ones show noun chunks related to transcription factors. The crucial difference is that the *nnpg*'s recognised act as patients *activators*, *repressors* and *regulators*.

1. *HSP25* is shown here in a patients position and thus being activated. The activator protein is not mentioned within the pattern.

```
[nxactr_pt
  [actr activator]
  [of of]
  [njpg
    [nnpgx
      [nnpg HSP25]]]]]
```

2. The following bracketing recognises that the gene *SPI-1* is being regulated. The regulator protein is not mentioned.

## System overview (Methods)

```
[nxtrregr_pt
  [dt-a a]
  [regr regulator]
  [of of]
  [nxgene
    [nxpg
      [nmpgx
        [nmpg SPI-1]]]
    [gene genes]]]
```

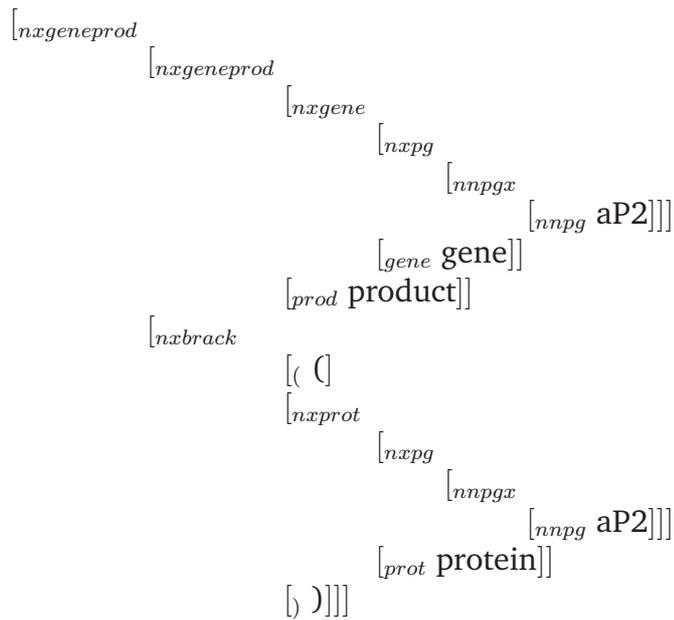
3. *IL-4* is being negatively regulated (*i.e.* repressed) in this example. The repressor remains unknown from this phrase.

```
[nxrepr_pt
  [dt-a a]
  [repr negative regulator]
  [of of]
  [nxpg
    [nmpgx
      [nmpg IL-4]]
    [nn signaling]]]
```

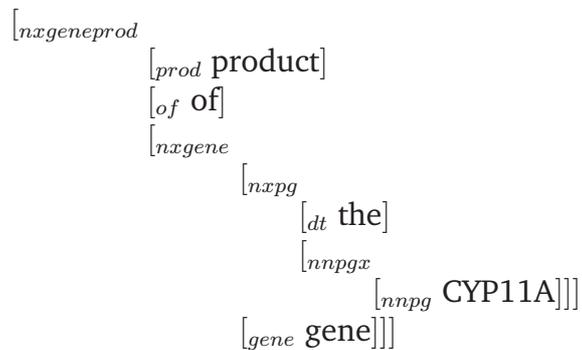
- **Gene products - nxgeneprod:** This category recognises entities referring to gene products. Nominal phrases of this type are highly relevant candidates for identifying products of gene expression process. In contrast to the examples given so far the pattern for *nxgeneprod* works on top of an already chunked entity, *i.e.* *nxgene*. This is the first example for nested structures. The pattern *nxgene* is recognised in a previous level and it is either preceded by term *product of* or followed by the term *product*.

1. Here, *aP2* is recognised as a gene product. Note, that the description in parenthesis disambiguates *aP2* as being a protein.

## 6.6 Recognising Entities and Relations with CASS grammars



- In this example the structure *nxgene* is preceded by *product of*). This determines the semantic category being a product of a gene, since proteins don't generate products.



- A conjunct of two *nnpgs* with an organism name (*i.e. human*) recognised as *nxgene* and followed by *products* changing the category to *nxgeneprod*.

## System overview (Methods)

```
[nxgeneprod
  [nxgene
    [nxbg
      [dt the]
      [org human]
      [nnpqx
        [nnpq CD80]]
      [cc and]
      [nnpqx
        [nnpq CD86]]]
    [gene gene]]
  [prod products]]
```

- **Protein complexes - *nxcmpx***: These patterns recognise mainly names of protein complexes. Most occurrences do not mention the proteins that the complex is made of. However, in some cases the participating proteins are mentioned.

1. A *nnpq* followed by the head noun complex without mentioning explicitly whether *BCR* is a protein complex or a complex of other entities. Still this entity makes sense for the following relation annotation as it might be disambiguated, for example through the verb that subcategorises this noun chunk. In any case *BCR* can be assumed to be the name of the complex.

```
[nxcmpx
  [nxbg
    [dt the]
    [nnpqx
      [nnpq BCR]]]
  [cmpx complex]]
```

2. This shows an example where all participants of the complex are mentioned explicitly, *i.e.* *PIG-A*, *PIG-H*, *PIG-C* and *GPI1*.

```

[nxcmpx
  [dt-a a]
  [cmpx complex]
  [of of]
  [nxpg
    [nnpqx
      [nnpq PIG-A]]
      [cma ,]
      [nn PIG-H]
      [cma ,]
      [nn PIG-C]
      [cma ,]
      [cc and]
      [nnpqx
        [nnpq GPI1 ]]]]]

```

- **Protein functions - nxfunct:** This category is not invented to determine what type of function a protein has but more to recognise noun chunks that mention the key word *function* like *the function of nnpq*. In a later step – if the function is related to expression – the function can be processed separately.

```

[nxfunct
  [cd one]
  [func function]
  [of of]
  [nxpg
    [nnpqx
      [nnpq ptc1]]]]

```

- **Parts of proteins** - Not all patterns related to proteins refer to complete proteins. Some also refer to parts of proteins. At the moment the only part of proteins that occur in the patterns are protein domains.

– **Protein domains - nxdom:** The head noun for patterns of this type are *domain*, *DNA binding domain*, *binding domain* and *kinase domain*.

```
[nxdom
  [nxpg
    [dt the]
    [nnpqx
      [nnpq PR]]]]
[dom domain]]
```

## Recognising genes and related entities

- **Genes - nxgene:** Besides the noun chunk referring to proteins the other highly relevant type of noun chunks are the ones referring to genes, *nxgene*. The only head noun that is used is *gene*.

1. In this case here a noun chunk referring to a gene is recognised. Additionally, the organism name is mentioned as well.

```
[nxgene
  [nxpg
    [dt-a a]
    [org human]
    [nnpqx
      [nnpq SEL-1L]]]]
[gene gene]]
```

2. Similarly as in the above example this pattern shows gene names with the organism mentioned.

```
[nxgene
  [nxpg
    [dt The]
    [org mouse]
    [nnpqx
      [nnpq Rh11]]
    [cc and]
    [nnpqx
      [nnpq Rhag]]]]
[gene genes]]
```

- **Allele - nxalle:** According to Lackie and Dow (2000) alleles are defined as *differ-*

ent forms or variants of a gene [...] Assumed to arise by mutation. Alternate form of a gene. Noun chunk referring to alleles are not relevant as arguments for the extraction and thus need to be excluded from the relevant results. Entities of this type are recognised so that they cannot interfere by accident with other types of entities.

```
[nxalle
  [dt the]
  [jj resistant]
  [alle allele]
  [of of]
  [nxgene
    [nxxpg
      [dt the]
      [nnpqx
        [nnpq Nramp1]]]
    [gene gene]
    [in in]
    [jj murine]
    [nns macrophages]]]
```

- **Diploids - *nxdiploid***: Similarly as for the alleles the diploids have to be excluded. I will show no example here but the structure is along the lines of the [*nxalle* class shown above.
  
- **Parts of genes** - Analogous to the class *parts of proteins* this item discusses the structures related to parts of genes.
  - **Upstream activating sequences – *nxuas\_ag*, *nxuas\_pt***  
 The upstream activating sequences are binding sites on (or preceding) genes that are responsible for the transcriptional activation of the gene. The example given here is a patients example that refers to the uas of the *DAL7* gene. The agens would refer to the transcription activator which binds to an upstream activating sequence.

## System overview (Methods)

```
[nxuas_pt
  [nxpg
    [nxpg
      [nnpqx
        [nnpq DAL7]]]]]
  [uasx
    [uas UAS]]]
```

- **Upstream repressing sequences** – **nxurs\_ag, nxurs\_pt**. Analogously to the item before on uas the upstream repressing sequence is the binding site on (or preceding) a gene that is responsible for the repression of the expression. The agens vs. patiens distinction is treated analogous.

```
[nxurs_pt
  [jj functional]
  [bs binding sites]
  [for for]
  [nxrepr_ag
    [nxpg
      [nxpg
        [dt the]
        [nnpqx
          [nnpq p53]]
        [nn tumor]]]
    [repr suppressor]]]]]
```

- **Promoter** – **nxprom**: The promoter of a gene is an obligatory sequence. This is the position where the transcription process starts. If a protein binds to this site, then one can infer that it regulates the expression of the corresponding gene and thus it is highly relevant for our task. The patterns for the detection of promoter regions of genes follow in principle the ideas of what has been illustrated in the structures before. A simple example is given here:

```

[nxprom
  [nxgene
    [nixpg
      [dt the]
      [nnpgx
        [nnpg Oct-4]]]
    [gene gene]]
  [prom promoter]]]]

```

- **Open reading frame – *nxorf***: The open reading frame is a part of the DNA. It is a part of genes which *are capable of being translated into proteins* (according to Lackie and Dow (2000)). A noun chunk related to this sequence is *nxorf*. This entity can occur as patients argument of a gene expression relation.

1. One of the simpler patterns shows that the gene name can also be mentioned and thus be used for the extraction.

```

[nxorf
  [nixpg
    [dt the]
    [nnpgx
      [nnpg M25]]]
  [orf open reading frame]]

```

2. Here is an example where the orf occurs explicitly as part of a gene, namely *M2*.

```

[nxorf
  [dt the]
  [jj second]
  [orf ORF]
  [of of]
  [nxgene
    [nixpg
      [dt the]
      [nnpgx
        [nnpg M2]]]
    [gene gene]
    [of of]
    [nns pneumoviruses]]]

```

3. Obvious head nouns are *open reading frame* and *orf*. Another head noun for the detection of orf's is *coding sequence* as can be seen here.

```
[nxorf
  [dt the]
  [coding coding]
  [sequ sequence]
  [of of]
  [nxbg
    [nnpbx
      [nnpb Hoxa13]]]]]
```

- **Binding sites** – *nxbs*, *nxbs\_ag*, *nxbs\_pt*: The binding site is a part of a gene. It contains as a necessary part the promoter and as an optional component the enhancer and/or silencer part. These two are the site where transcriptional activator or repressor usually bind to.

1. This category refers to bindings sites where it is not detectable from the structure whether they are used as agens or patiens. This can be determined in a later step. One clue for the determination can derive from the verb.

```
[nxbs
  [dt the]
  [bs binding site]
  [of of]
  [nxbg
    [nnpbx
      [nnpb Cbfa1]]]]]
```

2. This example shows a patiens binding site that serves as a binding site for a certain transcription factor. From this structure one can determine which protein binds to the binding site but not to which gene this binding site belongs.

```

[nxbs_pt
  [dt-a a]
  [jj high]
  [nn affinity]
  [bs binding site]
  [for for]
  [nextf
  [dt the]
  [tf transcription factor]
  [nxpg
    [nnpqx
      [nnpq Sp1]]]]]]

```

3. In contrast to the example before this structure shows a binding site of a gene. But it does not say which transcription factor binds to it.

```

[nxbs_ag
  [dt the]
  [jj SAF]
  [bs binding site]
  [in in]
  [nxpg
    [dt the]
    [nnpqx
      [nnpq CD4]]
    [nn silencer]]]]

```

- **Mutations – nxmut:** A series of terms trigger so-called **artificial contexts**, which indicate that some kind of manipulation has been carried out. This list comprises *mutant* (as well as the derivatives *mutation* and *mutating*), *recombinant*, *fusion*, *synthetic*, *modified*, *defects* (and the derivate *defective*). The plural derivatives are not listed here. I will not give examples for all of these terms since the construction of the noun chunks is similar to the following or according to the ones given before. Another similar category is *nxdel*, which refers to deletions. A simple example showing a chunked output. Other patterns are similar to the already given ones including of-PP's and and adjectival modifications.

```
[nxmut
  [mutation mutation]
  [of of]
  [nxpg
    [nnpgx
      [nnpg Pax3]]]]]
```

### Ambiguities between proteins or genes

For slightly more than 50% of the *nnp<sub>gs</sub>* it was not possible to judge from the noun chunk whether they are referring to a gene or a protein name. The reason is that there was no linguistic material (*i.e.* head noun) to make a decision. This means that the *nnp<sub>gx</sub>* chunk is not embedded or nested within any other nominal chunk assigning a specific category thus leaving it ambiguous. Depending on the context the selectional restrictions of the verb (or a relational noun) have been used for disambiguation in a later step.

- **nnp<sub>gx</sub>**: This example shows *nnp<sub>gs</sub>* without head nouns or verbs disambiguating between protein or gene name. In principle it can be used if disambiguated in a later step (for example through co-reference resolution). Another similar under-specified category is *n<sub>xpg</sub>* or *n<sub>xpg</sub>\_dep*.

```
[invn involvement]
[of of]
[nxpg
  [nnpgx
    [nnpg PAR-1]]
  [in in]
  [ntp this]
  [nn process]]
[sent .]
```

- A last class of nominal chunks are nominal relations, where only one argument is specified through an attached prepositional phrase. They occur as arguments of other relational words in the following steps. This class comprises examples like:
  - Overproduction - *n<sub>xopr</sub>d*, like *overproduction of IL-10*
  - (Over-)Expression - *n<sub>xexpr</sub>*, like *FasL gene expression*

### Other protein or gene-related entities

- **RNA - nxrna**: An entity which is mentioned frequently as well, is RNA. It can either refer to stable RNA or to RNA as a intermediate product that gets translated to a protein. In most cases this is not derivable from the information in the sentence. However in some cases it makes sense to access information from genome databases as well. In this case here it could be retrieved whether the gene codes for a protein and the transcription regulator might be missing. In a series of cases bioinformaticians can use data, even if it is only partial.

Besides the head noun *RNA* we also use the noun *transcript*. The patterns for *nxrna* are similar to the others shown before. The following shows just one possible outcome.

$$\begin{array}{l}
 [nxrna \\
 \quad [nxpg \\
 \quad \quad [nmpgx \\
 \quad \quad \quad [nmpg \text{ IL-6}]]] \\
 \quad [rna \text{ mRNA}]]
 \end{array}$$

### 6.6.2 Recognising Relations

This section describes the detection of relations between the biological entities that have been described in the previous sections. To get an insight in the relation recognition component I have divided this section into four parts. The first three are related to the different types of regulation, these are (i) the activation-related part, (ii) the repression-related part and (iii) the regulation-related part. The fourth part (iv) is related to artificial contexts. In the first three of these parts I describe in the (a) part the nominal relations and in the (b) part the different types of verbal relations. These comprise active and passive voice constructions as well as negated structures. The reason to treat the the relations on artificial contexts separately was that the artificial contexts (*i.e.* mutation or deletion of genes) force a re-interpretation of the relations. This means that the detected relation have to be mapped to other semantic types of relations while for some other cases it forces the semantic type of the relation to remain underspecified. This process will be expemplified later.

For each of the mentioned categories the following shows representative examples that are generated by the running the CASS grammar for relation detection on the output of the named entity recognition output. The relation detection grammar is arranged in cascades so that the output of one level is fed as input to the next level. It should be noted that an additional level has been introduced to normalise this output. This means that the type of interaction as well as the direction should be easily understandable from the annotation of the last level. In table 6.11 all six possible top-level annotations for the relation chunks are shown. The six combinations result from the three different

types of relations (*i.e.* activation, repression and underspecified regulation) and the two possible directions (*i.e.*  $xr$  versus  $rx$ , where  $x$  is the target, the entity regulated by the regulator  $r$ ).

	Activation	Repression	Regulation
Regulator - Target	ev_act_expr_rx	ev_rep_expr_rx	ev_reg_expr_rx
Target - Regulator	ev_act_expr_xr	ev_rep_expr_xr	ev_reg_expr_xr

Table 6.11: **Top-level categorisation of relations:** The overview shows the names of the categories from the top-level cascade of the relation processing. From the name the type of relation and the direction of the relation can be read. As an example,  $ev_{act}expr_{r,x}$  encodes an expression activation event, where the first argument ( $r$ ) is the regulator and the second argument ( $x$ ) is the target. This is the input for the script that extracts the relations from the bracketed structures.

To figure out which of the recognised relations belongs to which of the top-level categories one simply has to look in the appendix section B.30. For each of the top-level categories the corresponding relational categories are listed as disjunct on the right-hand-side of the rules. As can be seen from the appendix section some very few relations are not categorised according to these top-level concepts but treated separately. The top-level annotation has proven to be very helpful to normalise the semantic annotation for the extraction of the information-triples from the bracketed structures. This is carried out by a perl script which takes these normalised structures as input.

## (i) Activation relations

### (a) Nominal activation relations

The example here shows an activation relation that is expressed through the relational noun *induction*. Given that the first entity is tagged as gene expression – attached by *of* – and the second one is a protein or gene name – attached by *by* – with the appropriate biological knowledge we can infer that *Tat* activates *NOS-3* expression. This inference is expressed through the rule for  $nxev\_act\_expr$ . Other nouns which are also tagged as relational nouns of activation and thus can appear in similar subcategorising contexts are:

*activation, derepression, enhancement, positive regulation, stimulation, up regulation, upregulation, overexpression* etc. All in all 19 different terms were used as key words for this pattern.

```

[nxev_act_expr
  [dt the]
  [activation induction]
  [of of]
  [nxexpr
    [nxgene
      [nxpg
        [nnpgx
          [nnpj NOS-3]]]]
      [gene gene]]
    [expr expression]]
  [by by]
  [nxpg
    [nnpgx
      [nnpj Tat]]]]]

```

The other nominal activation relations are *nxev\_expr*, *nxev\_actr\_of\_expr*. The principles for the structure are the same and the details of the rules can be found in the appendix section.

#### (b) Verbal activation relations

Here I demonstrate the verbal activation relations comprising active voice constructions, passive voice constructions and negation.

1. The following example shows an active voice variant of a verbal activation relation. The key word *increase* has as synonyms *activate*, *derepress*, *enhance*, *increase*, *positively regulate*, *promote*, *stimulate*, *support*, *up regulate* as well as *upregulate*. All these verbs are allowed to occur in the same position within this pattern.

```

[ev_act_expr_rx
  [ev_act_expr_va
    [nxpg
      [nnpgx
        [nnpj IL-4]]]]
    [actv increased]
    [nxexpr
      [nxpg
        [nnpgx
          [nnpj VCAM-1]]]]
      [expr expression]]]]]

```

- Here an example is shown for a passive voice variant of a verbal activation relation. The key words are the same as in the active voice variant. The passive activation pattern is triggered through the *vx-actv-by* pattern. It should be noted that in contrast to the active voice variant here the top-level concept/category differs in the last two characters. Instead of *rx* we have *xr* here. *r* represents the regulator and *x* the target. This results from the fact that passive voice changes the order of the arguments.

```

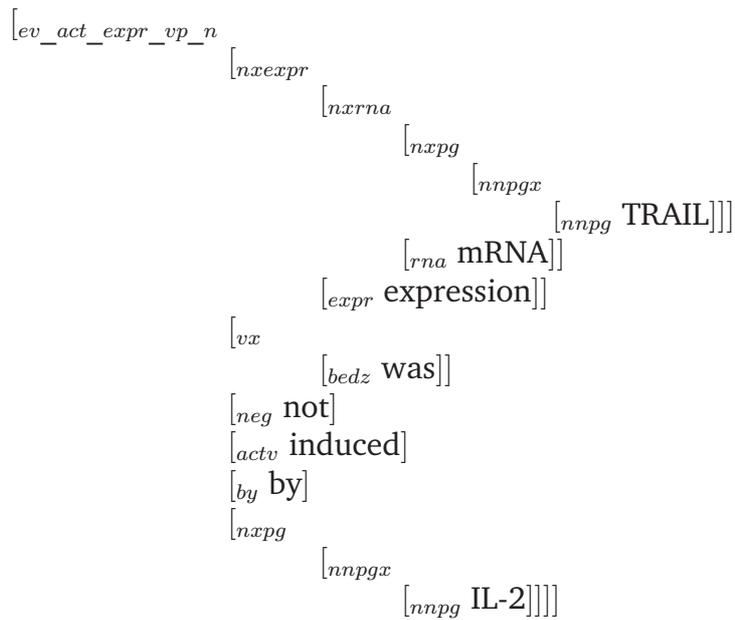
[ev_act_expr_xr
  [ev_act_expr_vp
    [nxprom
      [nxpg
        [dt The]
        [jjr TATA-less]
        [org rat]
        [nnpGx
          [nnpG GAD65]]]
        [prom promoter]]
    [vx
      [md can]
      [be be]]
      [actv activated]
      [by by]
      [nxpg
        [nnpGx
          [nnpG Sp1]]]]]]]]

```

- The third example for verbal activation is a combination of negation and passive voice construction. Of course, corresponding rules have to recognise the combination of negation and active voice constructions as well. Other key words triggering negation are *unable*, and *cannot*.<sup>15</sup>

---

<sup>15</sup>In a later version of this grammar we also implemented rules to properly deal with constructions like *A but not B*, which are not processed by this grammar here.



**(ii) Repression relations**

**(a) Nominal repression relations**

The pattern for the nominal repression relation is similar to the nominal activation pattern shown before. In contrast it has *inhibition* as head noun. Other head nouns for this pattern are *down regulation*, *downregulation*, *inactivation*, *loss*, *negative regulation*, *repression*, *suppression*.

## System overview (Methods)

[*nxev\_rep\_expr*  
[*dt* The]  
[*inh* inhibition]  
[*of* of]  
[*nexpr*  
[*ngene*  
[*npg*  
[*nnpqx*  
[*nnpq* StAR]]]  
[*gene* gene]]  
[*transc* transcription]]  
[*by* by]  
[*npg*  
[*nnpqx*  
[*nnpq* DAX-1]]]]]

### (b) Verbal repression relations

The verbal repression relations are constructed analogously to the verbal activation relations. Hence I will give only one active voice example. In this case here we recognise a infinitive construction, which has not been mentioned before.

```

[ev_rep_expr_rx
  [ev_rep_expr_va
    [nx_funct
      [cd one]
      [func function]
      [of of]
      [nxpg
        [nnp_gx
          [nnp_g ptc1]]]]]]
  [vx
    [bez is]]
  [to to]
  [repv repress]
  [nxexpr
    [nxpg
      [nnp_gx
        [nnp_g Shh]]]]
    [expr expression]
    [in in]
    [dt the]
    [jj anterior]
    [nn limb]
    [nn bud]]]]]

```

### (iii) Regulation relations

#### (a) Nominal regulation relations

The example for the underspecified regulation relation is analogous to the activation and repression examples immediately before. The key words are *expression*, *expressing*, *control*, *regulation*, *translation*, *transcription*.

## System overview (Methods)

```
[nxev_reg_expr
  [dt-a a]
  [regul regulation]
  [of of]
  [nxepr
    [njpg
      [nnpgx
        [nnpg NCAM]]]
    [expr expression]]
  [by by]
  [njpg
    [nnpgx
      [nnpg OTX2]]]]]
```

### (b) Verbal regulation relations

Here an example illustrating verbal regulation relations is presented. It demonstrates a passive voice construction. Same as before, active voice, negations and infinitive constructions are recognised. The verbal key words are: *affect*, *control* and *influence*.

```
[ev_reg_expr_xr
  [ev_reg_expr_vp
    [nxprom
      [njpg
        [dt the]
        [org rat]
        [nnpgx
          [nnpg StAR]]]
      [prom promoter]]
    [vx
      [bez is]]
    [regv regulated]
    [by by]
    [njpg
      [nnpgx
        [nnpg DAX-1]]]]]]]
```



or decreases. From that a biologist knows that this protein has been working as an enhancer or repressor. As a further consequence if the interaction between a deleted gene or a deleted protein and another protein is activation one can conclude that if it was present the relation type would have been repression. In case of the deleted gene or protein forces repression the argument can be made that if it wasn't deleted the expression would have been activated. The following two examples illustrate this principle.

1. The following shows an example where a deletion changes activation to repression.

```
[ev_act_expr_val_r
  [nxdel
    [deletion deletion]
    [of of]
    [nxgene
      [nxpg
        [dt the]
        [nnpqx
          [nnpq IRF-2]]]
        [gene gene]]]
    [actv increases]
    [nxexpr
      [nxpg
        [nnpqx
          [nnpq IFN-beta]]]
        [expr expression]]]
```

2. In this example here the absence of a protein or gene changes activation to repression.

```

[ev_act_expr_var
  [nxdel
    [dt the]
    [abse absence]
    [of of]
    [nxpg
      [nnpqx
        [nnpq IL-2]]
      [cc and]
      [nnpqx
        [nnpq IL-3]]]]]
  [actv induced]
  [nxepr
    [expr expression]
    [of of]
    [nxpg
      [dt both]
      [nnpqx
        [nnpq c-Myc]]
      [cc and]
      [nnpqx
        [nnpq Bax]]]]]]

```

### 6.6.3 Summary and numbers on entities and relations

#### Summary and numbers of occurrences of named entities

To give an overview of the distribution of the amount of retrieved entities the figure 6.1 shows a block chart with the corresponding figures. The chart shows the numbers for named entities retrieved within in the *M. musculus* corpus. Some concepts like protein or gene names occur very frequent, where other concepts like nominal phrases identifying binding sites occur comparably rare.

#### Summary and numbers on occurrences of relations

The pie chart in figure 6.2 gives an overview over the types of extracted relations for the *M. musculus* corpus. The class of activation relations is the most frequent type of relations (46%). This class is followed by the regulation relations (28%), and the repression relations represent the smallest class (21%). The class with negated relations are comparably rare (5%).

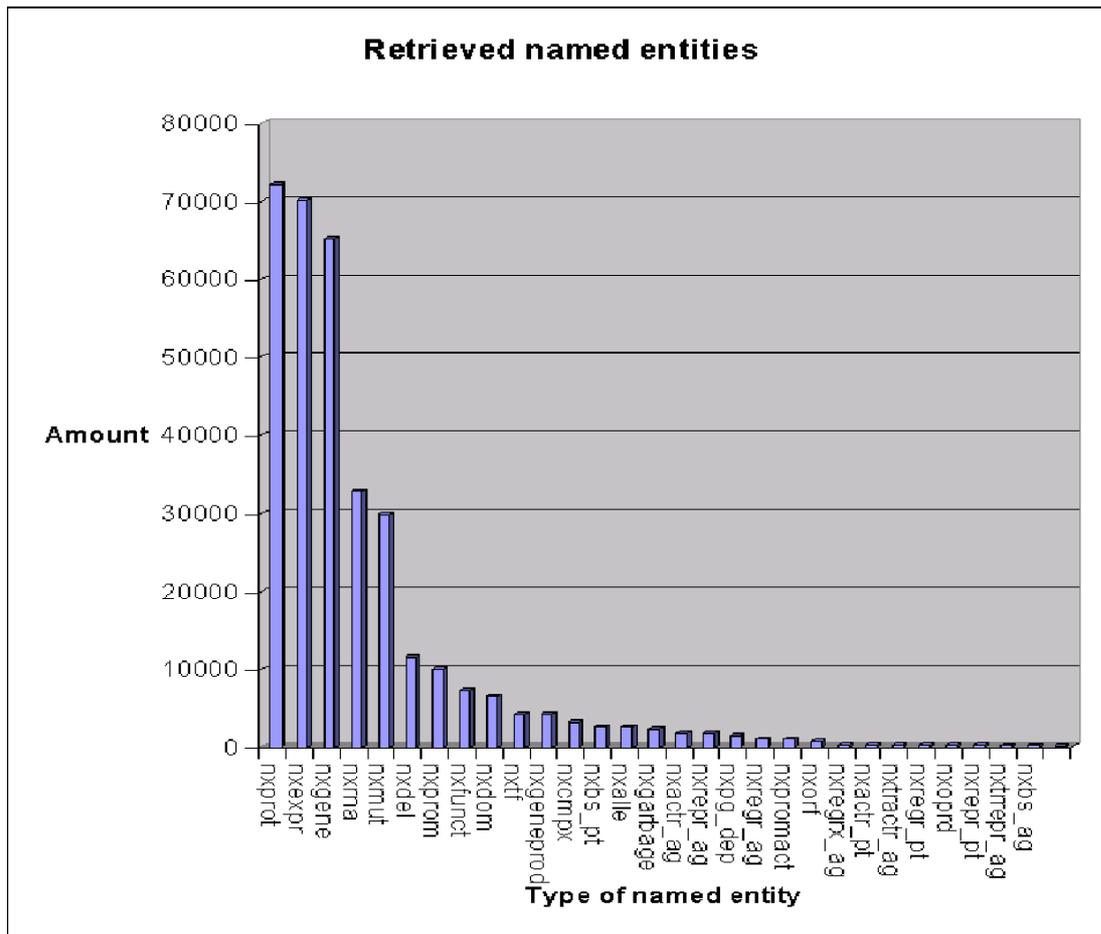


Figure 6.1: The block chart shows the distribution of the named entities for our experiment. The chart shows on the x-axis the frequency and on the y-axis the concepts that occur more than 100 times. As one can see the concepts *nxprot* and *nxgene* occur very frequent.

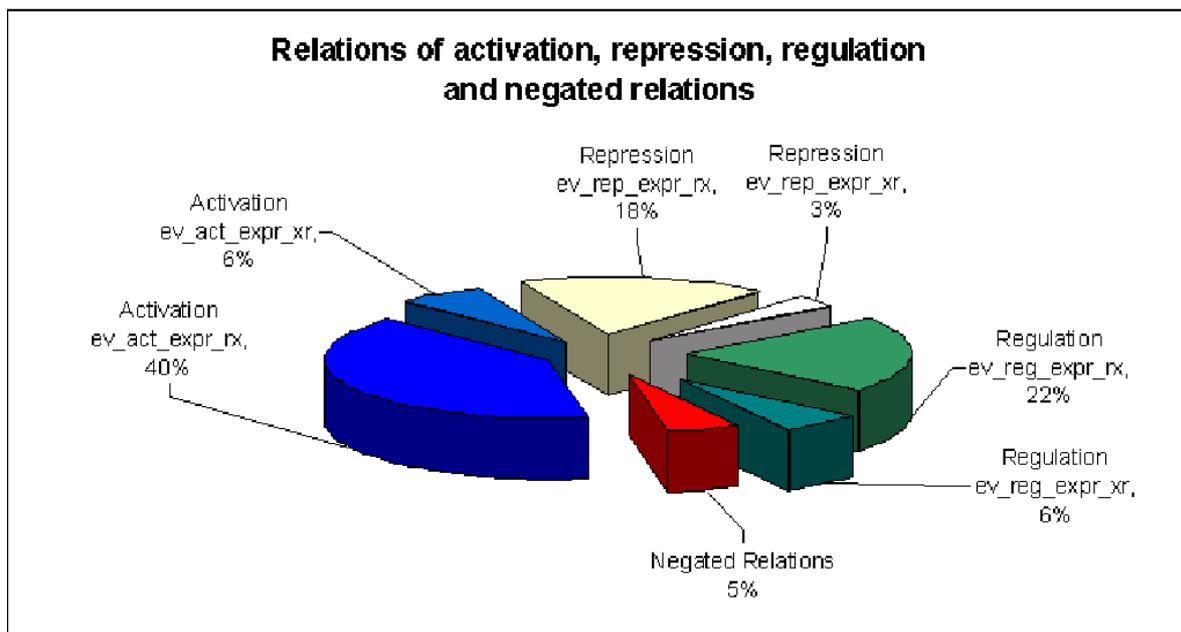


Figure 6.2: The pie chart shows the distribution of the relations that were extracted. The activation relations have the prefix *ev\_act*, the repression relations *ev\_rep* and the regulation relations have *ev\_reg*. The direction is indicated by *rx* (regulator - target) vs. *xr* (target - regulator).



# Chapter 7

## Results

### 7.1 Evaluation of Relation Extraction

#### 7.1.1 Evaluating the relation extraction for yeast

Using our relation extraction rules, we were able to extract 422 relation chunks from the yeast corpus. As one entity chunk can mention several different named entities, these correspond to a total of 597 extracted pairwise relations. However, as several relation chunks mention the same pairwise relations, this reduces to 441 unique pairwise relations comprising 126 up-regulations, 90 down-regulations, and 225 regulations of unknown direction and/or type.

Figure 7.1 displays these 441 relations as a regulatory network in which the nodes represent genes or proteins and the arcs are expression regulation relations. Known transcription factors according to the Saccharomyces Genome Database (SGD) Dwight *et al.* (2002) are denoted by black nodes. From a biological point of view, it is reassuring that the proteins serving as regulators in our relations tend to correspond to known *S. cerevisiae* transcription factors.

To evaluate the precision of the extracted relations, it was necessary to manually inspect all relations extracted from the evaluation corpus. This was done by using the TIGERSearch tool (Lezius (2002b)) for visualisation and easy retrieval of the results.<sup>1</sup> The main reason for the manual evaluation is that no labelled data is available for automatic evaluation. There are several efforts at the moment in building up such data, like the GENIA corpus (Kim *et al.* (2003)). But all these data have very few if at all annotated with gene expression relations.

The precision of the relations was evaluated at the semantic rather than the grammatical level. We thus carried out the evaluation in such a way that relations were

---

<sup>1</sup>It was little effort to convert the bracketed structures to TIGERSearch representation since TIGERSearch offers a import filter for this type of bracketing.

Table 7.1: Corpus statistics and evaluation of extraction results.

Corpus	Papers	Tokens	Gene/protein matches	Event chunks	Unique RX-pairs	Accuracy
<i>E. coli</i>	195,492	28,568,983	380,362	321	265	<b>85%</b>
<i>B. subtilis</i>	16,270	2,022,852	580,654	89	76	<b>90%</b>
<i>S. cerevisiae</i>	58,664	9,447,237	67,758	422	441	<b>83%</b>
<i>M. musculus</i>	688,937	106,027,447	3,599,912	1636	1276	<b>84%</b>
PubMed Cent.	5,075	19,199,318	558,941	158	N/A	<b>84%</b>

counted as correct if they extracted the correct biological conclusion, even if the analysis of the sentence was not as to be desired from a linguistic point of view. Conversely, a relation was counted as an error if the biological conclusion was wrong. Linguistic correctness of the structure was not evaluated. Given that only one person has been evaluating the biological correctness of the extracted data, it has to be mentioned that other biologists might have come to a slightly different result. We hope that the possible difference is minor and can be neglected.

In the yeast evaluation corpus 75 of 90 relation chunks (83%) extracted from the evaluation corpus were entirely correct, meaning that the relation corresponded to expression regulation, the regulator (**R**) and the regulatee (**X**) were correctly identified, and the direction of regulation (up or down) was correct if extracted. A further 6 relation chunks extracted the wrong direction of regulation but were otherwise correct; our accuracy increases to 90% if allowing for this minor type of error what we would call a sloppy evaluation. Approximately half of the errors made by our method stem from genetic modifications being overlooked—the relation being extracted is actually mentioned in the sentence, however it is not biologically relevant.

To estimate the coverage (or recall) of our method, we looked through 250 of 44,354 sentences that contain at least two gene/protein names. These contained only 8 relation chunks of the desired type. This would correspond to an estimate of 1419 in total. Since 422 of these were successfully extracted by our method on this part of the corpus, we estimate the coverage of our method to be around 30%. This corresponds to an F-score in the order of 44%, which is respectable by IE standards.

### 7.1.2 Evaluating the relation extraction for other organisms

As already described in the previous section, a system was developed for the extraction of gene expression networks. The system was first run on a yeast-specific corpus. To test the overall performance of the system it was applied to other organism specific corpora, *i.e.* *E.coli*, *B.subtilis* and *M.musculus*. The results for the relation extraction are described here.

Using our relation extraction rules, we were able to extract 2429 relation chunks for four organisms from PubMed abstracts (see table 7.1). As one entity chunk can mention

several different named entities, these corresponded to a total of 2,980 extracted pairwise relations. However, as several relation chunks mention the same pairwise relations, this reduces to 2,012 unique pairwise relations (RX-pairs).

## 7.2 Evaluation of Entity Recognition

For consistency, the ability of the system to correctly identify named entities at the level of semantic rather than grammatical correctness was evaluated as well. Again, through manual inspection of 500 named entities from the yeast evaluation corpus revealed 14 errors, which corresponds to an estimated accuracy of just over 97%. Surprisingly, many of these errors were committed when recognising *proteins*, for which our accuracy was only 95%. Phrases such as “telomerase associated protein” (which got confused with “telomerase protein” itself) were responsible for about half of these errors.

Among the 153 entities involved in relations no errors were detected, which is fewer than should be expected from our estimated accuracy on entity recognition (99% confidence according to hypergeometric test<sup>2</sup>). This suggests that the templates used for relation extraction are unlikely to match those sentence constructs on which the entity recognition goes wrong. False identification of named entities are thus unlikely to have an impact on the accuracy of our relation extraction.

## 7.3 Evaluation of POS-Tagging

We compared the POS-tagging accuracy of three parameter files on 24,798 held-out tokens from the GENIA 3.02 corpus. The best result was achieved using the parameters trained on the revised GENIA corpus, which correctly tagged 96.4% of tokens. This compares favourably to the 93.6% and 85.7% correct tokens achieved using the parameter file for the standard GENIA corpus and the standard English parameter file respectively.

## 7.4 Evaluation of Tokenisation

To estimate the precision of the sentence splitter 1,068 periods were manually checked. From these 1,068 periods 995 sentence boundaries and all 68 abbreviations were correctly identified. This results in an overall precision of 99.5%. This is comparable to the results in reported in *Schmid2000* for common English newswire texts.

---

<sup>2</sup>Hypergeometric test is used to determine whether a sample is representative or not in a certain scenario. For example, given an urn with  $N$  balls in it, where  $w$  are white and  $b$  are black. After a sample with  $K$  balls were drawn from the urn and  $x$  are observed to be white. The hypergeometric test can answer whether we would have there are too many white balls in the sample. It simply compares the fraction of  $\frac{w}{N}$  and  $\frac{x}{K}$

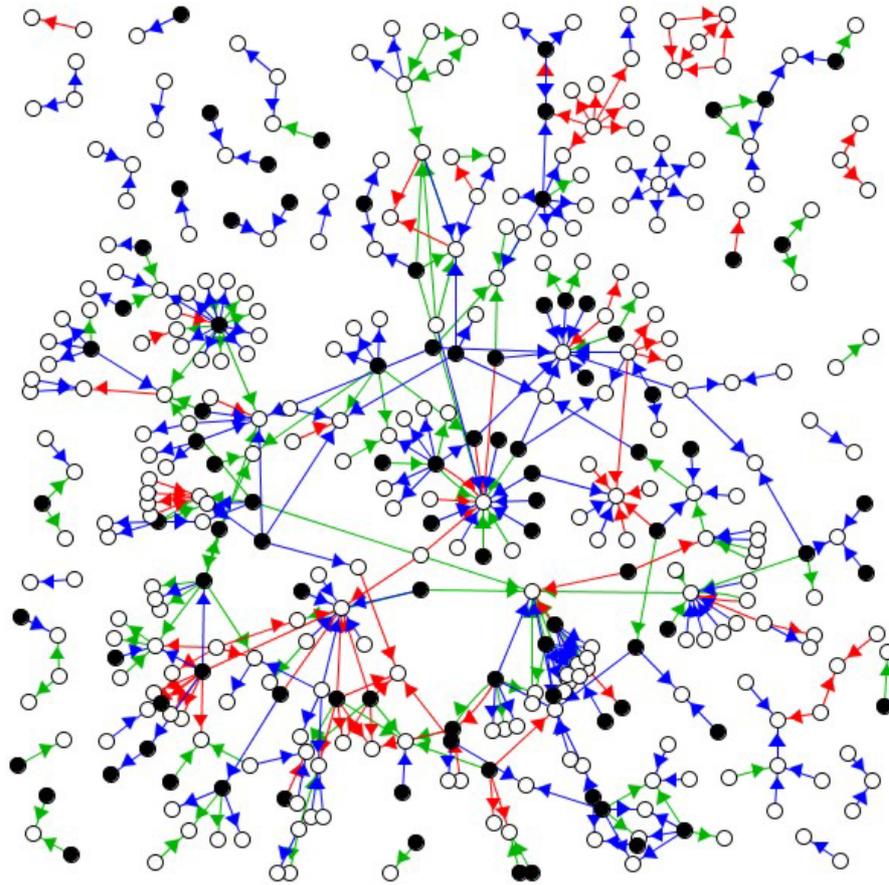


Figure 7.1: **The extracted network of gene regulation** The extracted relations are shown as a directed graph, in which each node corresponds to a gene or protein and each arc represents a pairwise relation. Arcs point from the regulator to the target and the type of regulation is specified by the color: up-regulation (green), down-regulation (red), and underspecified regulation (blue). Known transcription factors are highlighted as black nodes.

# Chapter 8

## Summary

Information extraction (IE) is a linguistically motivated engineering method for identification and extraction of factual knowledge from (large) text collections fitting into predefined information templates. In this part I describe a pilot study that uses IE methods to extract large-scale data with high precision from biomedical literature (PubMed abstracts). The system produces structured information that can be used for populating databases or for spreadsheet analysis. This implementation was a highly interdisciplinary piece of work and was carried out in collaboration with biologists and bioinformaticians. I worked for the SDBV (Scientific DataBases and Visualisation) group of EML Research gGmbH in close collaboration with the Bork group at the EMBL (European Molecular Biology Laboratory) to build this system.

The overall question that the system had to answer was which proteins are responsible for regulating the expression (*i.e.* transcription or translation) of which genes. As a literature resource we used PubMed abstracts. This differs from most comparable efforts. To give an insight into current research in this field I present the results that were achieved within the BioCreAtIvE assessment (see Yeh *et al.* (2005) and Hirschman *et al.* (2005) for details). BioCreAtIvE had as its main goal identifying gene-related entities in text and classifying these genes (or the corresponding gene products) according to the Gene Ontology (GO). GO was developed for the large-scale annotation of gene functions in databases (see Ashburner *et al.* (2000) for further details on Gene Ontology). Two things should be mentioned with respect to the BioCreAtIvE setting. One is that the identification of gene-related entities is extremely useful within biomedical text extraction systems and a comparison through shared tasks like BioCreAtIvE is useful. On the other hand the identification of corresponding GO terms seems to be too ambitious for NLP systems available at the moment. Nonetheless, the results show what is possible with current technology and which topics are of interest for further research.

The resulting system is able to extract a satisfactory number of relations (for yeast) comprising 422 relation chunks. Since one relation chunk can contain several binary relations we had 597 binary relations and after excluding duplicates, we retrieved (in a first version) 441 unique binary relations. The extracted relations can be represented as a directed graph in which each node corresponds to a gene or protein and each arc represents a pairwise relation. The "arrows" point from (known) regulators to targets.

## Summary

The system was able to distinguish three types of regulation, *i.e.* regulation, repression and activation. Such a graph clearly shows the distribution of the relation types of, for example, activation or repression assigned for 50% of the relations, the rest remains underspecified.

A domain expert – a biologist – evaluated the results for the yeast organism manually. The accuracy of the relations was evaluated at the semantic rather than the grammatical level. We thus carried out the evaluation in such a way that relations were counted as correct if they extracted the correct biological conclusion, even if the analysis of the sentence was not optimal from a linguistic point of view. Conversely, a relation was counted as an error if the biological conclusion was incorrect.

It should be noted that the evaluation showed that 75 of the 90 relation chunks (83%) within the pilot study extracted from the evaluation corpus were entirely correct, meaning that the relation corresponded to expression regulation, the regulator (R) and the regulated (X) were correctly identified, and the direction of regulation (up or down) was correct if extracted. Further 6 relation chunks extracted the wrong direction of regulation but were otherwise correct; our accuracy increases to 90% if we allow for this minor type of error. Approximately half of the errors made by our method stem from overlooked genetic modifications-although mentioned in the sentence, the extracted relation is not biologically relevant.

The two most important aspects of this work are the reusability of the system and the relevance of the results for the bioinformaticians. As regards reusability, it was possible to adapt the system to related questions. This means that with a series of modifications such as adaptation of the lexicon and manual extension of the set of rules, protein modifications could be extracted successfully. Results were presented at the SMBM 2005 symposium (Semantic Mining in Biomedicine) in Hinxton. This work is summarised in Šarić *et al.* (2005b) and was selected to appear in Bioinformatics 2005. The second and more important issue is that the IE system will be used in a biological scenario, which means that the results make sense biologically. The system developed will be used within a project called STRING (Search Tool for the Retrieval of Interacting Genes/Proteins). It will be integrated into the next version of STRING and will then be available for inspection at <http://string.embl.de/>.

## **Part III**

# **The Language of Biological Databases**



# Chapter 9

## The Swiss-Prot Database and its FT-lines

### 9.1 Introduction

Swiss-Prot (Boeckmann *et al.* (2003)) is an annotated protein sequence database established in 1986. It maintains curated protein sequence entries with a wide range of protein related information. Because of its high level of integration it is one of the main databases for protein sequences. It is maintained in a collaboration between the Swiss Institute for Bioinformatics (SIB) and the European Bioinformatics Institute (EBI).

The release 44.5 of 13-Sep-04 of Swiss-Prot contained 158,316 sequence entries, comprising 5,8238,578 amino acids abstracted from 119,969 references. Each sequence entry is organism specific. The three most frequent organisms are: (1) *Homo sapiens* with 11,461 entries (2) *Mus musculus*, or mouse, with 8,225 entries, and (3) *Saccharomyces cerevisiae*, or yeast, comprising 4,982 entries. To give an impression of the growth of the Swiss-Prot database the chart in picture (9.1) shows how the number of entries has changed over the years. Besides the sheer amount of entries and entry-related information there is another obstacle that makes the query and retrieval process so hard, namely the structural complexity and distribution of information through the Swiss-Prot entries. This problem is not only Swiss-Prot related but common to all biological databases, in particular protein databases. The amount in combination with the structural complexity of the data makes retrieval and exploitation of this information that is stored in these databases sometimes a hard task. Generally these resources are either exploited in tedious manual work or only partially within large-scale experiments where applications retrieved the information in an automated fashion possibly losing a part of it.

First, I give a rough overview of the structural organisation of a Swiss-Prot entry to illustrate this problem. Then I focus on a specific type of Swiss-Prot information, the FT-lines. This chapter finishes with a discussion of the limitations of standard tools for querying Swiss-Prot. The following chapter introduces an alternative for querying Swiss-Prot FT-lines to overcome the described obstacles, which finishes with a discussion on establishing criteria that enable us to organise the data in a more consistent way.

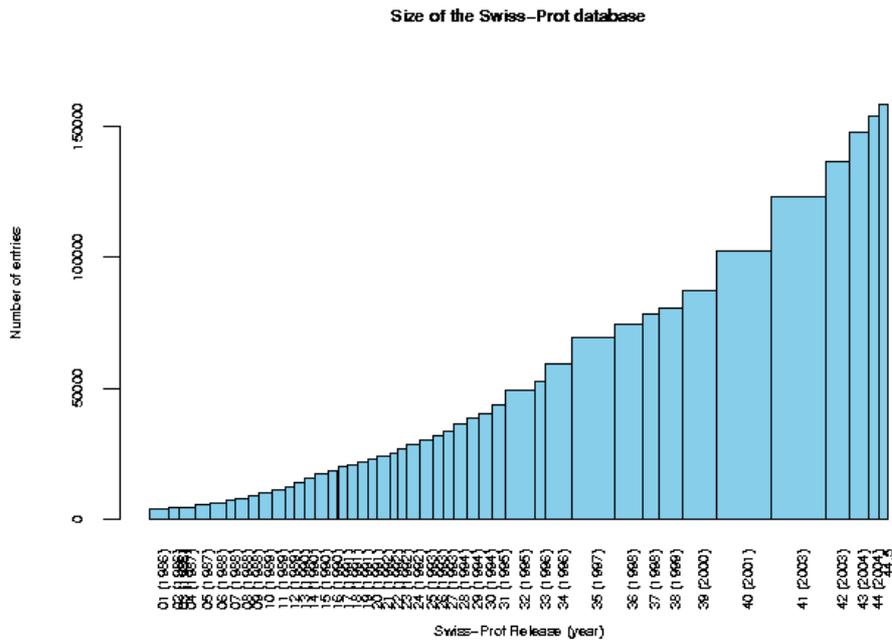


Figure 9.1: The picture illustrates the enormous growth of the Swiss-Prot database from 1986 to 2004. Each bar in the bar chart refers to a single release.

## 9.2 What Swiss-Prot covers

Each sequence entry in Swiss-Prot is composed of different lines each bearing a different type of information. The format differs from line-type to line-type according to differences in the type of information. Swiss-Prot in general - as most other sequence databases - distinguishes between two types of data, the core data and the annotation data:

1. The core data mainly consists of the following:
  - The sequence, also referred to as peptide sequence or amino acid sequence. This is a chain of amino acid residues, which are connected by peptide bonds. It is generally listed from the N-terminal to the C-terminal. Peptide sequences are referred to as protein sequences thus characterising the primary structure of a protein.
  - The bibliographical references are information about the bibliographic source that reports from which scientific publication the data has been recovered.
  - The taxonomic data describes the biological source of the protein.
2. The annotation data comprises a series of entries. It shows the both the functions of the protein as well as structural properties. These are categorised as follows:
  - The function(s) of the protein is (are) annotated.

- Post-translational modification(s) like carbohydrates, phosphorylation, acetylation, GPI-anchor are annotated.
- Domains and sites are annotated. For example calcium binding regions, ATP-binding sites, zinc fingers, homeobox, kringle, etc.
- The secondary structure of the protein is annotated. The secondary structure of a protein refers to certain common repeating structure. There are two types of secondary structures, *i.e.* alpha-helix and beta-pleated sheet.
- The quaternary structure of the protein is annotated, for example homodimerisation or heterotrimerisation, etc., indicating the complex structures to which this sequence can be part of.
- Similarities to other proteins are annotated.
- Known disease(s) associated with deficiency(ies) in the protein are listed.
- Sequence conflicts, variants, etc. are described.

The EBI (European Bioinformatics Institute) websites report about Swiss-Prot<sup>1</sup>:

The Swiss-Prot Protein Knowledgebase is a curated protein sequence database that provides a high level of annotation, a minimal level of redundancy and high level of integration with other databases.

This means that for each entry in the Swiss-Prot database the data has been collected through manual literature research. It has to be mentioned that there is no one-to-one correspondence between single references and sequence entries. It can either be the case that one entry is assembled from different references or from one reference different sequence entries for different organisms can be extracted. Since literature on proteomics, genomics and life sciences in general is steadily growing, the manual extraction is a steadily ongoing process. This merging of data goes hand in hand with the process of minimising the redundancies in Swiss-Prot as the user-manual reports in Bairoch (2003):

In Swiss-Prot we try as much as possible to merge all these data so as to minimise the redundancy of the database. If conflicts exist between various sequencing reports, they are indicated in the feature table of the corresponding entry.

### 9.2.1 The Swiss-Prot lines - heterogeneous information

Each sequence entry in Swiss-Prot comes with a set of lines describing properties of the peptide sequence. Each line has a separate format specifying the relevant information. The lines are indicated by a two-character line code, specifying the type of data. Some are found for each entry, some are optional. As the sample entry in the appendix (see appendix section A.1) shows these entries can easily span more than 180 lines and

<sup>1</sup>See: <http://www.ebi.ac.uk/swissprot/index.html>

contain free text comments on the entry over several lines as well as cross-referring identifiers. The table 9.1 (taken from the Swiss-Prot documentation) gives an overview of the different types of lines that can occur within each entry.

Line Code	Content	Occurrence in an Entry
The ID line	Identification	Once; starts the entry
The AC line	Accession numer(s)	Once or more
The DT line	Date	Three times
The DE line	Description	Once or more
The GN line	Gene name(s)	Optional
The OS line	Organism species	Once or more
The OG line	Organelle	Optional
The OC line	Organism classification	Once or more
The OX line	Taxonomy cross-reference(s)	Once or more
The RN line	Reference number	Once or more
The RP line	Reference position	Once or more
The RC line	Reference comment(s)	Optional
The RX line	Reference cross-references	Optional
The RA line	Reference authors	Once or more
The RG line	Reference group	Once or more (Optional)
The RT line	Reference title	Once or more (Optional)
The RL line	Reference location	Once or more
The CC line	Comments or notes	Optional
The DR line	Database cross-references	Optional
The KW line	Keywords	Optional
The FT line	Feature table data	Optional
The SQ line	Sequence header	Once
The sequence data line	Sequence data	Once
The // line	Termination line	Once; ends the entry

Table 9.1: **The Swiss-Prot entries:** The table shows the different types of entries within the Swiss-Prot database. The first column shows the official character code to identify the type of information. The second column shows a short information about the content of the specific line type. And, the last column indicates whether this type of information is mandatory or optional. It can be seen from this table that various types of information are packed in the database, like unique identifier, comment lines (free text), sequence information etc.

The amount of different types of information in table 9.1 shows the heterogeneity of information provided by the database<sup>2</sup>. All in all it can be said that it is nearly impossible to retrieve the complete information in an automated fashion, which is mainly due to two reasons:

- (i) Most of the knowledge contained in an entry is not presented in a satisfactorily

---

<sup>2</sup>It should be noted that Amos Bairoch *the head of* Swiss-Prot states that Swiss-Prot is designed for humans not for computers. In contrast to that Swiss-Prot is referred to as.

structured way. For example the *CC*-lines (*i.e.* commentary lines) contain information about the function or structure of a protein in free-text form.

- (ii) Each entry has information scattered over different parts of lines without explicit cross-references connecting the information bits. There is no indication of their collective information potential. Only users using the database regularly and thus knowing the structure and its shortcomings, can retrieve manually the desired information briquettes.

The consequence of these facts is that Swiss-Prot – and similar arguments can be made for other biological databases as well – is an underutilised source of knowledge.

I do not discuss the details of all Swiss-Prot lines, but – as already mentioned – I focus on the *FT*-lines. The main reason to pick this type of information is that it looks highly structured and provides detailed information about domain structure of the peptide sequence and thus on its functional properties. On the other hand it can be seen from the examples given that retrieving certain information across several lines requires a level of consistency which is hard to achieve.

### 9.2.2 The Swiss-Prot *FT*-lines

The Swiss-Prot *FT*-lines associate particular features to regions and other sites of interest in the sequence of proteins. In particular they list post-translational modifications, binding sites, enzyme active sites, local secondary structures and non-covalent bindings between amino acid residues that may be located at any distance in the sequence. The information is presented by triples consisting of a key word, an interval of the sequence and a short (possibly empty) natural language description providing additional information about the feature (see table 9.2 for an example).

Key word	Start	End	Length	Description
dna_bind	1	40	40	copper-fist.
domain	1	108	108	binds copper and dna.
domain	109	225	117	required for transcriptional activation.
metal	11	11	1	zinc (by similarity).
metal	14	14	1	zinc (by similarity).
metal	23	23	1	zinc (by similarity).
metal	25	25	1	zinc (by similarity).

Table 9.2: Swiss-Prot *FT*-lines for peptide sequence entry *ACE1\_YEAST*. The *FT*-lines already shows structured information (*i.e.* key words, start and end position and description). It should be noted that the description field contains unstructured data, *i.e.* textual description of properties of the domains.

The Swiss-Prot user manual (Bairoch (2003)) reports five different classes of feature annotations, *i.e.* five different types of key word classes. These are:

**Change indicators:** The class of change indicator annotations subsumes a series of categories that indicate that a certain span of the peptides sequence can occur in a modified way. The classes are: *CONFLICT*, this class reports about differences between different sources of reports (*i.e.* scientific literature), *VARIANT* comprises the class where one author reports that variants exist, *VARSPLIC*, reports about variants that are produced through splicing variants and *MUTAGEN* is about experimentally altered sites.

**Amino-acid modifications:** The class of amino-acid modifications subsumes the following classes of modifications that can be found for amino acids, where some of them have again a sub-classification. For example *MOD\_RES* lists post-translational modifications of residues (like acetylation, methylation, etc.) *LIPID*, reports on covalent binding of a lipid moiety, *DISULFID*, indicates existing disulfide bonds, where the *FROM* and *TO* endpoints indicate the two residues that are linked by this type of binding. Other classes are *CROSSLNK*, *CARBOHYD*, *SE\_CYS*, *METAL*, and *BINDING*.

**Regions:** The class of regions has various types of information collected. All refer to a region of the sequence. *SIGNAL*, for example indicates the so-called prepeptide (or, signal sequence). *TRANSIT* gives the extent of a transit peptide, like for example chloroplast. The other subclasses are: *PROPEP*, *CHAIN*, *PEPTIDE*, *DOMAIN*, *CA\_BIND*, *DNA\_BIND*, *NP\_BIND*, *TRANSMEM*, *ZN\_FING*, and *REPEAT*.

**Secondary structure:** The class of secondary structure comprises *HELIX*, reporting the types of helices, *STRAND*, indicating beta-strands and *TURN*, indicating turns.

**Others:** All information on regions within the peptide sequence that have a specific annotation which does not fit in the previous given occurs in this class. As an example *ACT\_SITE* indicates amino acids that are involved in the activity of an enzyme. In contrast to that *SITE* subsumes other types of *interesting* sites. Other subclasses are: *INIT\_MET*, *NON\_TER*, *NON\_CONS*, and *UNSURE*.

### 9.3 Querying Swiss-Prot – state of the art

To retrieve knowledge from Swiss-Prot various possibilities are offered. The most obvious one is through the Swiss-Prot web-site<sup>3</sup>. It offers free-text search. The whole database is treated as a set of text-files and can be accessed with a simple bag of word approach. Another query system, which is one of the most elaborated system at the moment to retrieve knowledge from Swiss-Prot (and other biological databases as well), is SRS (Sequence Retrieval System)<sup>4</sup> It is developed by Lion Bioscience AG<sup>5</sup> and its main

---

<sup>3</sup>[www.expasy.org/sprot/](http://www.expasy.org/sprot/)

<sup>4</sup>I only consider the querying of Swiss-Prot through tools that have a user interface and do not need much additional knowledge. Alternatives that allow for detailed parsing of Swiss-Prot, like the *swissknife* perl modules ( see Hermjakob *et al.* (1999) for details), are disregarded.

<sup>5</sup>Several versions of SRS are on the market. The version 7 is online accessible through: <http://srs.csc.fi>.

advantages are that it's easy to use and has a broad coverage. SRS supports for example the retrieval of proteins that are characterised by a particular set of features (as given in the Swiss-Prot FT-lines). It allows for querying feature values by means of regular expressions and Boolean combinations of the retrieved results. In order to find all phosphorylated proteins in a cytoplasmic location, for example, one may first ask for proteins with value “cytoplasmic” for any of its keywords, and then ask for those proteins among them that are “phosphorylated”. However it is not guaranteed that there has to be a sequence overlap between both subsequences. In addition the relevant keywords may be indicated by further conjunctive constraints and the term cytoplasmic may be part of a regular expression. The expressivity of the query-language underlying SRS is, however, not powerful enough to allow to retrieve all the information that is represented in the FT-lines. And to my knowledge this is also true for other search tools. The reason is twofold:

- (i) The information that a feature value is associated with a particular subsequence (a chain, domain etc.) of the protein is lost, when queries are combined by boolean expressions. This means that it is not possible to link a subsequent query to the previous one, for example by using variables for expressing the coreference.
- (ii) The part-whole information between subsequences (hence modification sites, motifs, domains and chains) that is encoded by the start and end positions of these sequences is not accessible for the query engine. It is, therefore only possible to ask for phosphorylated proteins with cytoplasmic tails. It is not possible to ask in addition whether the phosphorylation site is actually part of the cytoplasmic tail. To illustrate this example figure 9.2 shows a screenshot of the SRS query interface. It shows that the two query items have to occur within the same FT-line description column. But it is not possible to express that a part-whole relation has to hold between the two sequence parts.

Another problem with which existing retrieval systems cannot deal with is the interpretation of terminological items depending on the context in which they occur. For example, the following two tables (*i.e.* table 9.3 and 9.4) show Swiss-Prot FT-lines of two different peptide sequences. The fact that the represented subsequence has the property of being calcium-binding is expressed in two different ways. The first table (table 9.3) shows the calcium-binding information within the key word column; the second table (table 9.4) shows the same information embedded in a more complex term given in the description column. In addition, the same information (*i.e.* the calcium-binding capacity of a subsequence) is expressed using different terms, *i.e.* *CA\_BIND* vs. *CALCIUM-BINDING*. To be able to retrieve this information the user has to query for the same information at different positions. This is just one example on scatteredness of information through an entry. To assure that information in the database is made completely available to the user two possible approaches can be taken. One possibility is that all possible positions where bits of information can be stored have to be made explicit. A second possibility is to guarantee transparency, which can be achieved by analysing and translating complex terms into a terminologically standardised complete and sound representation.

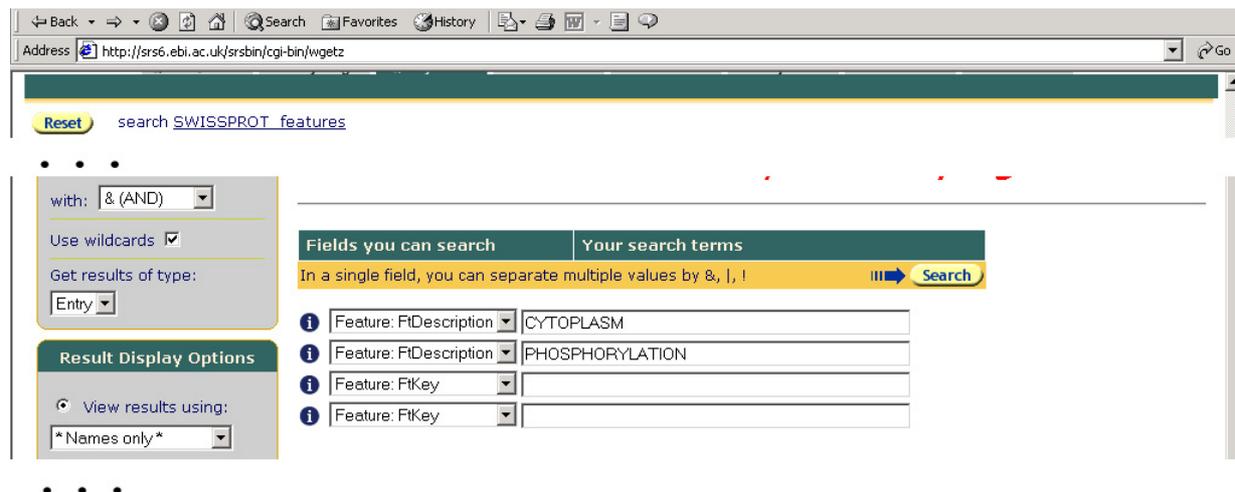


Figure 9.2: The figure shows a screenshot of a SRS query on the web. The description column of the FT-lines is queried for *CYTOPLASMIC* and *PHOSPHORYLATION*. It should be noted that it is not possible to establish a relation between the two queries, like for example a part-whole or overlap relation.

	Key word	Start	End	Description
FT	CA_BIND	700	711	EF-HAND 1 (POTENTIAL).

Table 9.3: The table shows one of the FT-lines copied from the entry for *AD60D\_DROME* showing information on calcium-binding properties in the key word column.

To summarise, what makes the extraction and exploitation of Swiss-Prot information so hard – and this holds for most of the comparable databases – is that on the one hand the query tools, that are available for this task at the moment, lack the expressive power to retrieve all information that Swiss-Prot captures. On the other hand the knowledge that Swiss-Prot covers is not yet so well-structured and transparent that it could be retrieved in a user-friendly way. In the following section I introduce an approach that overcomes the expressivity problem of the querying tools for the FT-lines. In addition, I discuss how the data could be re-structured in a semantically consistent way.

	Key word	Start	End	Description
FT	CHAIN	21	695	CALCIUM-BINDING ACIDIC-REPEAT PROTEIN.

Table 9.4: The table shows one of the FT lines copied from the entry for *ARP\_EUGGR* showing information on calcium-binding properties in the description column.



# Chapter 10

## Adapting TIGERSearch for Cell Biology

### 10.1 Introducing TIGERSearch

TIGERSearch (see Lezius *et al.* (2002) and Lezius (2002a) for details) is a search engine for the retrieval of information from graph structures. It was originally designed for querying linguistic hierarchical structures, *i.e.* syntactically annotated sentences in a text corpus. This means that the text collection which is to be queried by TIGERSearch must have been annotated syntactically beforehand. TIGERSearch is meant to give linguists an easy and intuitive access to such "linguistically annotated databases". There are three important issues summing up the key features of TIGERSearch:

- TIGERSearch is a software-package with a carefully designed **graphical user interface** that can be downloaded through [www.tigersearch.de](http://www.tigersearch.de). It offers to formulate queries in an intuitive manner either graphical, *i.e.* by *drawing* partial graphs, or through a textual query input, which is supported by syntax highlighting and various menus for choosing the correct feature names etc. for a given tree database. Results (*i.e.* sets of tree-structures) are displayed graphically by the TIGERSearch GraphViewer, which allows for convenient browsing of results. An example of a tree structure in TIGERSearch is shown in 10.1.
- The **TIGERSearch query language** supports elements which are well known from grammar formalisms in computational linguistics. A restricted variant of attribute-value structures is used to represent the annotated trees. Type definitions can be used to structure the corpus nomenclature. Template definitions can be used to break down complex queries. The use of variables and regular expressions makes the query language very powerful. Some sample queries both in TIGERSearch query language as well as in natural language description are presented here to illustrate the expressive power of this language.

(10.1) Find all sentences containing the word *run*, where this word is annotated with the part-of-speech *NN*:

```
[word="run" & pos="NN"]
```

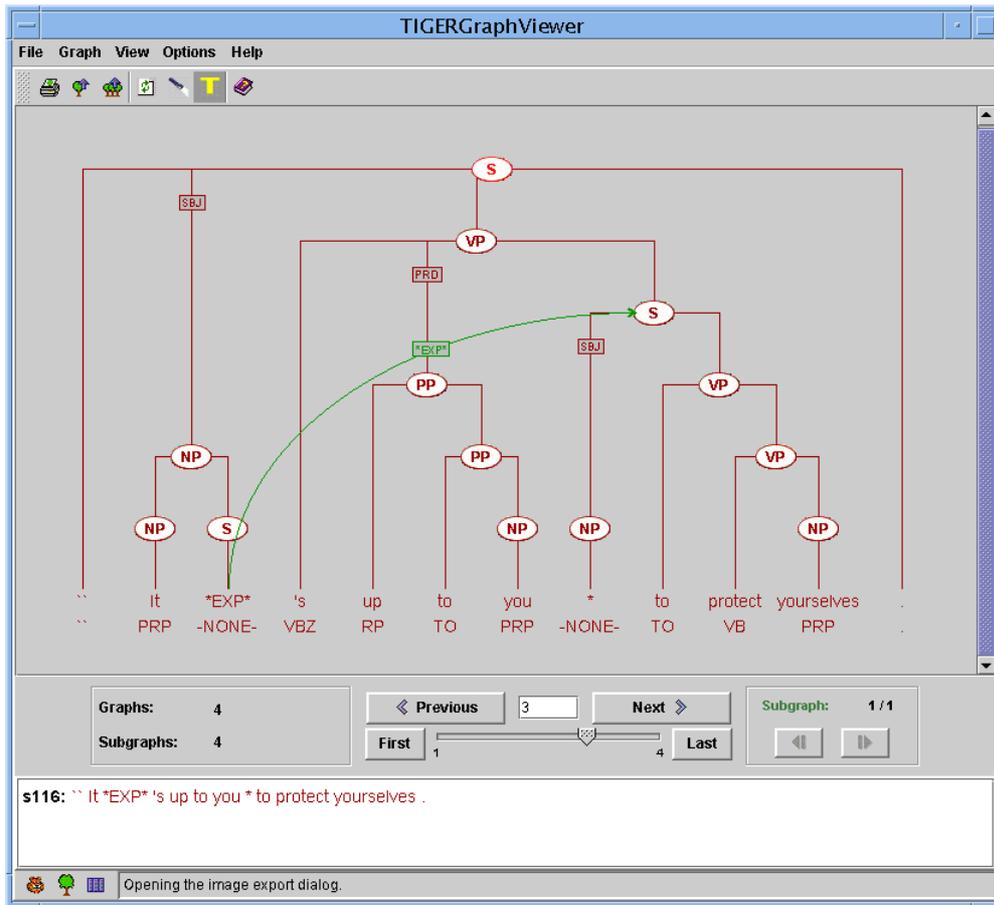


Figure 10.1: A screen shot showing a syntactically annotated sentence within TIGERSearch graphically represented. The tree has at its leaves the words of the sentence with the corresponding POS-tags below. The tree itself is a graph that has labels at its nodes and labels at its arcs. It should be noted that the green arc indicates a certain type of long distance dependency.

(10.2) Find all words starting with the letters *Ma* and are annotated either as common noun or as named entities the following query retrieves the desired results:

```
[word=/Ma.* / & pos= ("NN"|"NE")]
```

- TIGERSearch is based on XML. **TIGERSearch-XML** is a specific XML version developed for annotation of corpora. It mainly serves for the import and for the export of results. This state-of-the-art technology can be used to manipulate and aggregate query results.

## 10.2 Representing and querying FT-lines in TIGERSearch

The preceding sections illustrate some shortcomings of common Swiss-Prot query tools, especially SRS. This section presents an alternative approach for overcoming these shortcomings. The overall idea is to transform the Swiss-Prot FT-lines representation into TIGERSearch-XML. As shown above, this makes it possible to use TIGERSearch for querying the FT-lines. The architecture for such an approach can be described by the two following major components:

**Translation Component:** The information that is only implicitly contained in the database is made explicit, by translating the database contents into more structured representations, *i.e.* TIGERSearch-XML. The additional structure is based on a well defined ontological classification. The so enriched database is therefore semantically sound and accessible to intelligent search engines like TIGERSearch.

**TIGERSearch Search Engine:** This component refers to the TIGERSearch search facility that allows the user to query the re-structured database. The query language as well as the deductive search engine incorporates the same ontological knowledge that is used in the translation component. This guarantees that a maximal amount of information can be retrieved.

The advantage of TIGERSearch is that it has a representation and query language that is powerful enough to deal with arbitrarily complex feature combinations and it is supported by the underlying ontological features of the tables, in particular part-whole and precedence relations between sequence elements as well as between any subsequences of interest.<sup>1</sup>

Before going into the details of the translation component I want to show a TIGERSearch query allowing to retrieve proteins that have a phosphorylated cytoplasmic location. According to the example in previous section this shows that TIGERSearch has the necessary expressive power to express this query:

```
(10.3)[location="CYTOPLASMIC"] >*
      [aa\-mod="PHOSPHORYLATION"]
```

Given that a concept or attribute like *location* has been annotated the first bracket specifies that the location need be *cytoplasmic*. The second bracket specifies that all nodes that are annotated with *aa-mod* should be retrieved that are annotated with *phosphorylation*. To express that the phosphorylated residue has to be within the cytoplasmic location the ">\*" operator is used. It expresses the (transitive) part-whole relation between the two concepts or regions.

---

<sup>1</sup>For the time being the focus lies on the sophisticated retrieval of information with respect to one particular database, here Swiss-Prot. Accessing and exploiting all relevant databases in the way it is done by SRS is in principle possible, but requires for each such database a specific translation module.

The following example illustrates once more the expressive power of TIGERSearch. In contrast to SRS it is possible to refer from one sub-query to another. In order to find all subsequences of interest (*i.e.* motifs, domains etc.) that are connected by a disulfide bond, one conjunctively combines the following four queries: (a) find all non-trivial sub-sequences (NT) that precede each other (note that the relation of precedence is expressed by ".\*"); (b) and (c) find amino acid residues (T) that are part of these sequences, respectively; (d) show that there is a disulfide bridge between these residues. To guarantee that subqueries (a) and (b), (a) and (c) as well as (b) and (d), and (c) and (d) talk about the same subsequences we simply use the variables #v, #w, #vt, and #wt for each of the subqueries. Through a conjunction, &, we can then relate these sub-queries. Putting this all together we can use the following query:

```
#v:[NT] .* #w:[NT] & #v >
#vt:[T] & #w > #wt:[T] & #vt >~DISULFID #wt
```

How these variables will be instantiated and how the whole query will be resolved in detail by TIGERSearch will be described in more detail below. The section before will now describe the translation module that takes the FT-lines and makes use of our ontological knowledge about proteins and creates the database on which the TIGERSearch engine operates. It is important to note that in this approach there is no crucial difference between the language that is used to represent the data and the query language that allows the user to express his queries to the database.<sup>2</sup> TIGERSearch avoids this separation by taking the query language as a simple but powerful extension of the representation language. The languages only differ in the level of specification (totally specified vs. underspecified). As an advantage the user has to learn just one language. The query language is based on regular expressions, and it is supported by pull-down-menus for key word search. It is transparent and user-friendly. Furthermore TIGERSearch is equipped with a graphical user interface that displays the results in a tree-like graph structure (see Table 10.5).

## 10.3 Database Construction

### 10.3.1 General Idea

In the following I illustrate the general principles for the automatic transformation of Swiss-Prot feature table entries into a TIGERSearch-XML representation. In a prototype version I have implemented the conversion to TIGERSearch-XML according to the described features. This conversion procedure allowed to convert more than 95 % of the FT-lines<sup>3</sup>. The result of the conversion is imported to TIGERSearch and it is thus accessible for the TIGERSearch engine. It should be noted that not only the FT-lines are converted but additional information from the Swiss-Prot entries is encoded as well. This

---

<sup>2</sup>The difference is considerable and obvious if you consider SQL as query language for relational databases.

<sup>3</sup>There are few entries that can not be converted due to inconsistencies in the Swiss-Prot format. In addition there are few other exceptions that are not covered by the described procedure.

comprises relevant parts of the sequence (*i.e.* amino acids), the identifier, the protein name etc. There are five items that describe the guiding principles for the conversion.

1. Relevant regions of a peptide sequence are represented as nodes. The start and end points that are given in the Swiss-Prot representation are represented as the leaves of the nodes. They include the corresponding amino acids. As an example, a relevant region is a *transmembrane region* which can be represented in TIGERSearch as depicted in table 10.1.

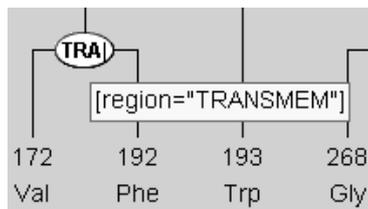


Table 10.1: A *transmembrane domain* is represented in TIGERSearch as spanning the region between sequence position 172 and 192.

2. Additional features for single amino acids may be associated to the terminal nodes (as well to non-terminal nodes) of the tree by means of feature-value matrices. As an example *Metal binding site*, or a *Phosphorylation site* can thus be encoded. This can be seen from the picture in table 10.2.

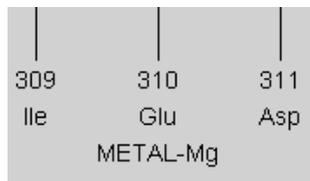


Table 10.2: An amino acid modification (*METAL-Mg*) is annotated below the sequence position 310 and its corresponding amino acid residue (*Glu*).

3. Precedence relations between domains are encoded through the linear order of the domains. This means that if a domain is located in the sequence before another it is also shown before in the TIGERSearch representation. This information can thus be retrieved through TIGERSearch.
4. Inclusion of domains is represented through dominating nodes. Thus if two regions *A* and *B* include a third region *C*, the corresponding nodes *A'* and *B'* dominate the corresponding *C'*.
5. Relations between single amino acids (and thus between the corresponding domains) are represented through secondary edges (*e.g.* *Disulfide bonds*), see Table 10.3.

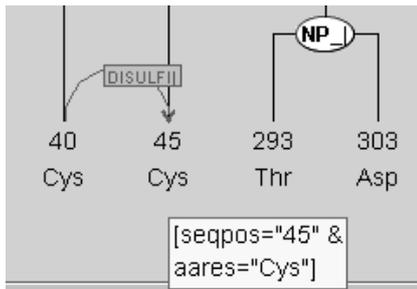


Table 10.3: A relation between single amino acids *i.e.* a *disulfide bridge* is represented between sequence position 40 and 45.

### 10.3.2 Going through an example

To illustrate the details of this transformation process I go over a complete example in this section. I use an FT-line entry from Swiss-Prot for the peptide sequence 11SB\_CUCMA (shown in Table 10.4). Each step of the conversion towards the TIGERSearch representation is documented. The result of the conversion can be seen from table 10.5.

Starting from Table 10.4 the first step consists of categorising the given information according to the three schemata introduced above in table 10.1, table 10.2, and table 10.3. A distinction is made between regions/domains of peptide sequences, relations between single amino acids or regions and single amino acids. The domains in this example are given by the key words *DOMAIN*, *CHAIN*, *SIGNAL*. The binding between single residues or corresponding regions or domains refer to *DISULFID* (disulfide bridge). And the third category comprises modifications of single amino acids, *i.e.* *MOD\_RES*.<sup>4</sup>

Key word	Start	End	Description
SIGNAL	1	21	
CHAIN	22	480	11S GLOBULIN BETA SUBUNIT
CHAIN	22	296	GAMMA CHAIN (ACIDIC)
CHAIN	297	480	DELTA CHAIN (BASIC)
MOD_RES	22	22	PYRROLIDONE CARBOXYLIC ACID
DISULFID	124	303	INTERCHAIN (GAMMA-DELTA) (POTENTIAL)
CONFLICT	27	27	S -> E (IN REF. 2)
CONFLICT	30	30	E -> S (IN REF. 2)

Table 10.4: The table shows a sample entry of Swiss-Prot FT-lines for the protein 11SB\_CUCMA with the *key word*, the *start* and *end*-position as well as the *description* column.

The leaves of the tree are linearly ordered and consist of the sequence of amino

<sup>4</sup>Comments on uncertainties like CONFLICT, SIMILAR or UNSURE are not represented for the moment. They will be captured in some later version.

The screenshot displays two windows from the TIGERSearch software. The background window, TIGERSearch, is in 'Graphical mode' and shows a query: `#v:[NT] .* #w:[NT] & [CHAIN="GAMMA\ CHAIN\ \ (ACIDIC\)] > #vt:[T] & [CHAIN="DELTA\ CHAIN\ \ (BASIC\)] > #wt:[T] & #vt > ~DISULFID #wt`. The foreground window, TIGERGraphViewer, shows a hierarchical graph for protein 11SB\_CUCMA. The root node is 11S, which branches into Met (1) and Ser (21). Node 21 branches into Gln (22) and Cys (124). Node 124 branches into Asn (296) and Gly (297). Node 297 branches into Cys (303) and Glu (480). A DISULFID bond is shown between Cys (124) and Cys (303). The graph is labeled PYRROLIDONE CARBOXYLIC |. The interface includes navigation buttons for Previous, Next, First, and Last, and a search bar.

Table 10.5: The rear window shows the query window with features that can be used for the queries. On the left-hand side of the rear window the *terminal features*, *Nonterminal features*, and *edge labels* can be seen. The right-hand side of the query window shows the query interface. The front window shows the resulting graph with the hierarchical structure for the protein 11SB\_CUCMA. The detailed information for this entry is shown in table 10.4.

acid residues<sup>5</sup>. Table 10.5 shows the boundaries of the sequence intervals with their corresponding position number.

Most of the feature records associated with leaf nodes are not shown in the graph of Table 10.5. Only the values of the “main feature”, *i.e.* protein ID for the root node, type of region for the non-terminal and sequence position for the terminal nodes, are depicted. Note that this is dependent on the settings of TIGERSearch and that there exists the option to visualise the complete feature records.

Keywords that qualify single amino-acid residues introduce terminal nodes of the tree and are translated as features having the corresponding description on the keywords as values. To give an example, for node 22 the complete feature record is the following.

<sup>5</sup>The amino acid can easily be accessed via the position number from the Swiss-Prot sequence line.

SEQ_POS	22
RESIDUE	Gln
MOD_RES	PYRROLIDONE CARBOXYLIC ACID

Keywords that qualify regions (DOMAIN, SIGNAL, and CHAIN and others as listed in Appendix A.3 of the Swiss-Prot user-manual) are treated in a similar way. They introduce the non-terminal nodes of the tree dominating the corresponding sequence. In case the domain has another domain as part of it, the part-of relation is represented in the tree by means of dominance. This means the embracing domain dominates the subsumed. The limiting nodes of the domains are node depicted.

The keywords are translated to feature names like DOMAIN, or CHAIN etc. The description is added as value. The CHAIN domains are thus represented as nodes in the tree. The key CHAIN (see Table 10.4) with its value "GAMMA CHAIN (ACIDIC)" spans thus from position 22 to 296. The position 121 being necessary for the disulfide bridge is dominated by this node. Hence, it is represented within the borders of this CHAIN. The root node carries in addition the the protein ID.

The disulfide bridge (or any other kind of non-covalent binding between residues) between the two "CHAIN" domains is represented as a "secondary edge" between two amino acids. The amino acids are part of the two peptide regions "GAMMA CHAIN" and "DELTA CHAIN". The relation between these two domains can now easily be queried and represented in the graphical representation of the protein. Note that it is not trivial to retrieve this information from the Swiss-Prot FT-lines without TIGERSearch.

In the above example the two regions (from 22 to 296 and from 297 to 480) are contiguous and completely partition the whole region of the globulin beta subunit. Cases in which (amino-acid residues of) far distant and hence disconnected regions are connected (by a secondary edge) are also captured by the TIGERSearch language.

The previous paragraphs present a translation schema for all descriptions in the Swiss-Prot feature tables as strings. TIGERSearch supports the search for information contained in these strings by allowing regular expressions in the query language (see section 10.4 for details).

### 10.3.3 Future extensions

In addition to the given transformation procedure some more extensions would be of great value. As an example a particular feature-value pair could be inherited along the dominance relation. An amino-acid modification which is encoded through the feature MOD\_RES in the FT-lines should be conceived not only as being a property of a single amino-acid. But, it should be treated as a feature of the region that contains the modified residue as well as of the whole protein. A possibility to represent this property is to inherit this feature bottom-up the tree. A question that arises with this approach is whether all features that are assigned at the bottom level can be propagated up while some features might have only a local scope.

A further continuation of this work includes the development of a parser and an formal ontology that allows structuring this information. This would allow to make the

data accessible to queries that use concepts from this ontology and hence abstract over the arbitrariness of the free text form. Descriptions associated to regions, like ASP/GLU-RICH, ACIDIC, CYTOPLASMIC TAIL, GOGO-A0101 ALPHA CHAIN might then be translated by feature value pairs like the following.

SEQ_CHAR	ASP/GLU-RICH
???	ACIDIC
LOC	CYTOPLASMIC
NAME	GOGO-A0101 ALPHA CHAIN

In a translation like the proposed a dictionary associating the single lexical items with ontological categories is necessary. As an example it is necessary to know that *cytoplasmic* in the above mentioned example refers to the cytoplasm of a cell. And, that the cytoplasm of a cell is of type location. Along these lines it then would be possible to annotate *extracellular* with the same ontological concept and to retrieve both with just one query.

## 10.4 The Representation and Query language

The following paragraphs give an overview of the search facilities of the TIGERSearch tool to illustrate the expressiveness of the TIGERSearch query language. It can also be used as a brief introduction to the TIGERSearch query language. A more detailed introduction can be found in Lezius (2002a), König and Lezius (2002a), König and Lezius (2002b). Note that a quick reference guide can be found in Lezius *et al.* (2002).

The first paragraph, (i) **Nodes and Terminals** introduces options to search for features associated with nodes and terminals. The second paragraph, (ii) **Node relations** explains how to query relations between nodes. (iii) **Graph descriptions** introduces more complex queries. As an example they show how to specify properties of a graph in the query language. The last paragraph (iv) **Types** sums up the types of variables that can be used within the query language.

### (i) Nodes and Terminals

As previously mentioned any node in the graphical representation of proteins is associated with a feature record, *i.e.* a flat feature structure whose feature values are constants. The query language allows boolean expressions to be used both on the feature value level and on the feature-value-pair level. To give an example, the query to retrieve domain characteristics being either a “SH2” or a “SH3” looks as follows:

(10.4) [DOMAIN = ("SH2"|"SH3")]

To retrieve an “extracellular” domain that has a phosphorylation site at some residue, one simply concatenates the two feature-value pairs using the conjunction operator “&”. The attribute *modRes* is used to represent the Swiss-Prot annotation *MOD\_RES*.

(10.5) [DOMAIN = "EXTRACELLULAR"] & [modRes = "PHOSPHORYLATION"]

Feature values which cannot be enumerated (such as most of the descriptions so far) can be referred to by regular expressions. For example, values containing the string EXTRACELLULAR, like “DOMAIN=EXTRACELLULAR (POTENTIAL)”, will not be recognised by query 10.5. However they can be retrieved by the following query (the "/" symbols mark the beginning and end of a regular expression):

(10.6) 
$$\begin{aligned} &[\text{DOMAIN} = /. * \text{EXTRACELLULAR} . */ ] \& \\ &[\text{modRes} = /. * \text{PHOSPHORYLATION} . */ ] \end{aligned}$$

### (ii) Node relations

Since graphs are two-dimensional objects, one basic node relation for each dimension is needed: direct precedence "." for the horizontal dimension and labelled<sup>6</sup> direct dominance ">L" for the vertical dimension (the precedence of two inner nodes is defined as the precedence of their leftmost terminal successors). There are also several derived relations such as general dominance or a sibling relation. In case both properties from query 10.5 relate to the same region one can easily represent this with help of the dominance relation ">" (instead of "&").

(10.7) 
$$\begin{aligned} &[\text{DOMAIN} = " \text{EXTRACELLULAR} " ] > * \\ &[\text{modRes} = " \text{PHOSPHORYLATION} " ] \end{aligned}$$

Secondary edges with labels L are represented by ">~L". The disulfide bond from position 124 (seqpos = "124") to position 303 (seqpos = "303") in table 10.4 can be identified through the following query (#v and #w are variables and match terminals as well as non-terminals):

(10.8) 
$$[\text{seqpos} = \#v] > \sim \text{DISULFID} [\text{seqpos} = \#w]$$

### (iii) Graph descriptions

A graph description consists of restricted Boolean expressions of node relations. Negation is not allowed. To express identity of two node descriptions variables are used. The following expression describes a disulfide bridge between two residues belonging to two different nodes. I express that the secondary edge should combine two amino acid residues that are bonded by a disulfide bond by restricting the variables to terminal nodes (T) and specify the label of the secondary edge. The variables #v and #w are restricted to non-terminals (NT) and are dominating the terminal nodes:

```
#v:[NT] .* #w:[NT] &
#v > #vt:[T] &
#w > #wt:[T] &
#vt >~DISULFID #wt
```

---

<sup>6</sup>The label L is optional.

**(iv) Types**

Besides the type "T" for feature records of terminal nodes TIGERSearch has the built-in types "NT" for feature records of non terminal nodes and "FREC" for feature records in general.

The user can also define type hierarchies for features. A type definition might include subtypes or constants. The following example illustrates a type hierarchy for amino acids:

```
aminoAcid := aliphatic, aromatic, others.
aliphatic := glycine, alanine, valine, leucine, isoleucyne.
aromatic := phenylalanine, tyrosine, tryptophan.
others := ... .
glycine := GLY.
alanine := ALA.
...
```

Such a hierarchy can be used to formulate queries in a more adequate way:

```
(10.9) [aares = "aromatic"]
```

**10.5 Conclusion**

This chapter presents the application of the TIGERSearch search tool to a database of protein sequences and protein features which are automatically extracted from Swiss-Prot entries. TIGERSearch is implemented in Java and is available for all major platforms.

Current work concentrates on the improvement of query processing efficiency by using more complex search space filters. On a technical level, a client/server architecture is planned to be implemented. Thin clients will communicate with a high-end server that serves incoming query requests.

Furthermore the translation component is planned to be extended. A parser and a more complex ontology of proteins, protein parts and protein properties are under development and will allow to translate the so far non-analysed natural language descriptions into ontologically structured representations that allow for high level TIGERSearch queries.



# Chapter 11

## Summary

In contrast to the sheer amount of computer-readable protein databases, only a small amount of information contained in these databases can be automatically retrieved by any of the existing query systems and data mining techniques. This is so because

1. most of the knowledge contained in these databases is not presented in a completely and satisfactorily structured way, but is expressed in text form, such as comment lines; and
2. important information is often distributed over different parts of a database without explicit cross-references connecting these parts, and hence without indication of their collective information potential.

If at all, only domain experts are capable of identifying, understanding and exploiting this information. Obviously, it is hard to figure out what information they are missing. Their capabilities are, however, constrained due to the considerable number of databases each of which contains a vast amount of entries. Hence, biological databases are a greatly underutilised knowledge source.

The aim of the work presented in this part of the thesis was to show a way of substantially increasing the amount of information that may be reliably derived from biological databases. As a case study this experiment was carried out with a sequence database, the reference protein database Swiss-Prot (for details see Boeckmann *et al.* (2003) or <http://www.expasy.org/sprot/>). I concentrated on structural properties of proteins that are contained in the database only implicitly because the structure of the database doesn't provide a particular (set of) slot(s) for it. This is the case in particular for the information given in the so-called FT-lines (feature table lines). An example of a feature table entry is shown in table 11.1.

To show how advanced searching capabilities can be used in a scenario like the one described above I used the TigerSearch Engine to query the Swiss-Prot FT-lines. I present a system that has been developed to make the information contained in the FT-lines much more transparent to the user in comparison to standard search facilities. In

## Summary

Key word	Start	End	Length	Description
SIGNAL	1	21		
CHAIN	22	480		11S GLOBULIN BETA SUBUNIT
CHAIN	22	296		GAMMA CHAIN (ACIDIC)
CHAIN	297	480		DELTA CHAIN (BASIC)
...	...	...	...	...

Table 11.1: A shortened version of a representative set of FT-lines for a random Swiss-Prot entry. The FT-lines are annotated with keywords, start and end position and a description. It should be noted that the description field contains unstructured data, *i.e.* textual description of properties of the domains.

addition, the use of TigerSearch allows the user to view the knowledge from different perspectives and combinations, which can roughly be described as an ontology-driven perspective on the FT-lines. The system for acquiring the information in the new format comprises two components:

**Translation Component:** Information that is only implicitly contained in the FT-lines is made explicit by translating the contents into a more structured representation, that is the TigerXML representation. The additional structure relies on an implicit ontology. The thus enhanced representation of the FT-lines is therefore semantically sound and accessible to an intelligent search engine like TigerSearch;

**TigerSearch Engine:** The TigerSearch component allows the user to query the restructured databases. The query language as well as the deductive search engine incorporates the same ontological knowledge that is used in the translation component. This guarantees that a maximal amount of information can now be retrieved from the FT-lines.

Sample output of a TigerSearch tree is shown in section 10.3 in table 10.4.

At the moment relations such as dominance and precedence in the ontology-driven representation can be queried. The approach allows addition of further concepts and relations on the basis of a lexicon of biomedical terminology. Such a lexicon contains syntactic and semantic features for its entries. A further developed approach would thus allow the combination of techniques developed in Natural Language Understanding, Theorem Proving and Ontological Engineering. A team to further develop this work would need to be highly interdisciplinary, combining specialists in the fields of biology, computational linguistics and computer science (especially databases and ontologies).

## **Part IV**

# **Modelling Biological Processes**



# Chapter 12

## Ontologies and Bio-Ontologies

There are three main issues addressed within this chapter. First, I motivate the use of ontologies and describe some ontological applications from a general point of view. This comprises an outline of what I regard an ontology to be within this thesis. The second goal is to describe important criteria for building ontologies. The corresponding section presents some guidelines for such an effort. The third goal is to present a rough overview of different ontologies that are in use, where mainly two types are considered. One is about common ontologies for general applications (*i.e.* independent of a specific domain). The other type of ontologies, which are considered, are biology-related ontologies.

### What is the prime interest for building ontologies?

Since Aristotle researchers from various scientific disciplines have published definitions and discussions on what an ontology is. Depending on the scientific context a different answer is given. The ontology-related literature offers various answers to this question. Ancient Greeks defined ontology as the philosophical discipline of being. Questions that occurred in their discussion were:

- What is being?
- What characterises being?
- What are things?
- How can things be classified?

Aristotle was the first to introduce categories like *substance*, *quality*, *quantity*, *relation*, *action*, *passion*, *place* and *time*, which still are fundamental for classifying things. For most of the time during the past 20 centuries the term ontology has been used to name a philosophical discipline. At the end of the last century a new definition of what ontologies refer to has emerged. Ontologies have become an important research issue in computer science. A series of definitions related to machine interpretability have thus evolved. According to Guarino (1997) one motivation for building ontologies is “the

possibility of knowledge sharing and reuse across different applications”. The term *application* shows the trend towards machine interpretability as key. The most frequently cited definition is from Gruber (see Gruber (1993) for details) according to which “an ontology is an explicit specification of a conceptualisation.” A bit more detailed definition is given according Guarino (1998) as “a logical theory, which gives an explicit, partial account of a conceptualisation”. The term *conceptualisation* refers to the idea that a person or a group has of the world. There are two key issues occurring through nearly all of the definitions: (i) the effort to build a model (of a part) of reality, and (ii) this model is always subject to a special perspective that this someone or this group has on (the part of) reality. These are the two key items that guideline the following work. For more details a short but nice overview of the various types of definitions is given in Gomez-Perez *et al.* (2004).

### **Why dealing with ontologies?**

Given that an ontology is a model (of a part) of reality it can be applied in a series of scenarios. Mostly it is intended to use an ontology as a backbone or a guiding principle for achieving consistency through design and implementation of various applications within one domain. The main motivation is thus usually to reduce the costs for the development and maintenance of each application system. The essential advantage of using ontologies is the combination in accessing existing knowledge and having the possibility to infer new knowledge on top of the existing facts.

The justification for making these endeavours is the machine processability for inferencing in order to support humans whenever large-scale inferencing is necessary. Several applications can be thought of to be supported. Within the domain of natural language processing (NLP) systems ontologies are frequently used. To give an example, Cimiano (2002) demonstrates the use of an ontology for the resolution of bridging references.

In general it can be said that ontologies are relevant for NLP systems whenever semantic interpretation is necessary. There exist other applications like knowledge management systems that use ontologies as organisational backbone (examples are given in Abecker and van Elst (2003)). One might object that automatic deduction systems do not perform as flexible as humans do, and that the complexity of information handling is limited. Nonetheless ontologies are needed to reduce manual work, at least to a certain extent. As an example within the past decades the field of molecular biology has developed to a discipline that is being confronted with massive accumulation of data (previous chapters and the following section report on that). The enormous masses of information make it necessary to automate the processes of information generation and analysis. Problems like cleaning and consistency checking of database entries appear on the agenda of bioinformaticians more and more. The problems of semantic consistency of data are of pivotal interest for maintaining a database in the field of molecular biology and biochemistry.

## Why ontologies for molecular biology?

Biologists face a fundamental problem concerning the huge amount of heterogeneous data they have gathered and presumably will gather within the future. Especially this combination of (i) already available data, (ii) the steadily growing efforts to automatically process large-scale data and produce new data, and (iii) the heterogeneity due to the scatteredness of information through hundreds of databases make it nearly impossible to process this automatically in a semantically consistent and transparent way. This is illustrated with some examples in the following.

There exist several scientific groups that build and maintain databases with sequence information, be it gene sequences or protein sequences. Besides the most prominent, Swiss-Prot (see Boeckmann *et al.* (2003)) there exist hundreds of other databases with similar, related information. Unfortunately, nearly each group has developed its own annotation. The consequence is that the same function appears in different databases with different descriptions. Even within the same database different entries are to be found for the same entity. To illustrate this I have chose one example from the KEGG database<sup>1</sup> (a thorough description is given in Kanehisa (1997) and Kanehisa and Goto (2000)). For one and the same compound there are three different entries to be found in KEGG, *i.e.* *NTP*, *Nucleoside triphosphate* and *Ribonucleoside triphosphate*. At least this was the case at the time of writing this thesis. It should be noted that all three are listed with the same sum formula  $C_5H_{12}O_{13}P_3R$ , and, all three compounds are listed with a picture showing the same structure. For a scientist working with several databases in concert it is obviously an extremely time consuming task to achieve consistency through the data without too much undesired redundancy. The crucial problem when merging data from different databases is to have compatible and consistent meta description of the data and to have consistency within the databases that are accessed.

Another example in the field of molecular biology concerns the frequently used term *gene*. When trying to find a definition for the concept *gene* the massive problems for achieving consistency gets obvious. The following citation is taken from Schulze-Kremer (1998). It reports the confusion that can be met following definitions across various biological database regarding this term.

For GDB (Fasman *et al.* (1996)), a gene is a "DNA fragment that can be transcribed and translated into a protein". For Genbank (see Benson *et al.* (2002)) GSDB [Keen *et al.* (1996)], however, a gene is a "DNA region of biological interest with a name and that carries a genetic trait or phenotype" which includes non-structural coding DNA regions like intron, promoter and enhancer. There is a clear semantic distinction between those two notions of gene but both continue to be used thereby adding another level of complexity to data integration.

Following the given examples it should have become clear that avoiding (or at least reducing) this confusion for relevant and frequently used terms by formalising these concepts is of crucial importance. As already mentioned, the cleaning of bio-databases is

---

<sup>1</sup>Available through <http://www.genome.ad.jp/kegg/>.

a motivation for using ontologies as well. Using ontologies and cross-referring database entries to one and the same ontological concept can help to achieve more consistencies through different databases (different databases would thus mean the same thing when referring to a certain concept) and even within one and the same database (different annotators would thus mean the same thing when populating the database with instances).

### **A scenario for using bio-ontologies**

This section describes a possible scenario for using an ontology as the backbone for different applications. In the area of biology the crucial challenge is the enormous amount of heterogeneous, scattered and complex data. On the one hand biologists face the problem of heterogeneous data through several databases. On the other hand, there's the enormous amount of scientific publications in the area of biology which is steadily growing. At the same time biological databases are growing. They contain lots of information in a textual form, usually referred to as comment lines. As an example, Swiss-Prot offers information on subunits of protein complexes and on functions of proteins etc. buried in such comment lines. A popular procedure is to have a group of scientists (so called annotators) that populate the database with this kind of information. It should be obvious that it is highly cost-intensive to maintain such textual information.

When merging information from various databases a relevant question is the identification of identical entries through these databases. Synonyms occur frequently, for example for the naming of proteins or genes<sup>2</sup>. It is getting more and more popular to use so-called GO terms to express identical functions of genes or proteins across various databases. GO abbreviates Gene Ontology. GO terms were invented to relate or classify genes according to their products, for which the code.<sup>3</sup> GO is a taxonomy, which was invented for the annotation of gene functions. If two entries from different databases have the same GO classification scientists suppose them to be the same. This does not mean that they are the same but have the same function.

This is not only being done for database entries as such, but also for free-text annotations of protein functions (like the mentioned comment lines). Since identification in this scenario is based on a taxonomic identification only, it is fairly obvious that a more detailed ontological classification with a sound definition of the concepts can improve the results. Another common approach is to integrate information based on identical names of two objects from different sources. This does not guarantee better results than the first approach since names in molecular biology tend to be highly ambiguous. The probably most popular example in molecular biology is the confusion about gene and protein names. Not only that they have various synonyms for one and the same entity, they make extensive use of homonyms when giving the same name to a special gene

---

<sup>2</sup>Swiss-Prot for example showed on March, 5th 2004 for the entry *Q9BR43* a peptide sequence called *DJ1093G12.6 [Fragment]* (see <http://www.expasy.org/cgi-bin/niceprot.pl?Q9BR43>). The same entry shows the synonym name *A novel protein*, see screenshot of the entry on in the Appendix A.2.

<sup>3</sup>Some details on GO are given in section 12.3.2, or can be obtained through <http://www.geneontology.org>.

and its corresponding gene expression, for example the protein. In case of genes or proteins one could easily define within an ontology them to be same if they have the same sequence and the organisms they occur in.

A second application scenario for ontologies is for the classification of chemical compounds. Several databases (like KEGG, see Kanehisa (1997) and Kanehisa and Goto (2000)) maintain massive amounts of information. Chemical compounds are stored with structural information – mostly in a graphical form – and lists of synonym names. When querying for pentose (a special kind of monosaccharides, see details in section refglycosugsec) only one match is found, *i.e.* deoxyribose (which is correct). However, other pentoses are not listed as synonyms. For example riboses are a type of pentoses and might not contain the name pentose. A classification based on functional groups of chemical compounds reflects ontologically the structure of the compounds. Thus the class of riboses can be classified as pentoses and one could easily detect all relevant entries. In addition, this would help organising the database in a more consistent way (more on that topic can be found in Wittig *et al.* (2004)). A more detailed view with examples on how ontologies can be useful for information extraction systems is shown in 13.1.

In the succeeding sections of this chapter a definition of an ontology is given. The section starts with a terminological distinction between the terms ontologies, controlled vocabularies, thesauri and taxonomies. This is followed by an introduction and explanation of criteria for building ontologies. The final part of this chapter gives an overview of existing ontologies most of which are relevant for the biological domain.

## 12.1 What is an ontology?

The term *ontology* is in use across various disciplines. From the discipline of philosophy to community of database designers controversial definitions are reported. According to Gruber (1993) and Sowa (1999) philosophers regard ontology as a philosophical discipline and thus are interested in a systematic account and characterisation of things that exist or may exist. For others an ontology is an informal (in contrast to formal) system that defines concepts. The following statement from Gruber (1993) is the probably most cited and referred to definition in the field of artificial intelligence, knowledge representation and natural language processing.

An ontology is an explicit specification of a conceptualisation. The term is borrowed from philosophy, where an ontology is a systematic account of Existence.

This means that a *conceptualisation* is viewed as an abstract model that one person or a group of persons have of a domain of interest. The explicit specification of this model is what Gruber then calls ontology. Gruber (1993) reduces the term existence for the area of artificial intelligence to – a rather poor description of – what can be represented:

For Artificial Intelligence systems what exists is that which can be represented.

With Gruber (1993) an ontology can be regarded as an explicit specification representing an abstract conceptualisation of the world or a part of it for some purpose. There has been criticism against this definition – as for example in Guarino and Giaretta (1995) – which can be inspected in detail through the given references.

I've simply taken the above mentioned ideas to illustrate that this field is subject to ongoing discussions. In the following the focus of the discussion is shifted from philosophical perspectives to a more pragmatic discussion questioning how to specify a model and how to apply this to the domain of molecular biology.

To use such a specification of a conceptualisation (or a model) as a resource for an interoperable system one first needs a consistent shared understanding of what the relevant information means. But to be able to share this knowledge between people and transfer it to machines a suitable representation language is needed. Or, to put it in a different way: people and machines need to speak a common language to share their knowledge.

Another important issue is the use of the encoded knowledge beyond the simple exchange of data. This means that when applying a model to a specific domain ontologies offer the possibility to generate new knowledge from existing ontological knowledge. This procedure is usually referred to as inferencing. A crucial issue is the existence of an ontology on the one hand and an inferencing engine on the other hand. Although these are two separate components they have to work together, which means that the inferencing engine has to be tailored to the ontological model. A pure specification of concepts without regarding the inferencing engine can lead to a representation disregarding its usability. This also relates to the fact that ontologies are commonly designed for a certain purpose. This means that the conclusions and inferences that the system should draw have to be adequate for some type of application that the user has in mind.

It should be noted that it is common to distinguish between concepts and instances. In an ontology only concepts are represented. Instances are not supposed to be parts of an ontology. They are usually kept in databases. Systems that merge both type of information are usually referred to a knowledge bases.

In the following a definition of the term *ontology* is given by relating it to other terms, which are frequently used either synonymously with ontology or appear together with the term ontology. These terms are *controlled vocabulary*, *thesaurus*, and *taxonomy*. Here are some criteria that are important to distinguish these terms.

**Controlled vocabulary:** A controlled vocabulary is a list of terms or phrases. It can be seen as a kind of naming convention that is used for a special purpose that users of a certain domain agree upon. It reflects a set of relevant concepts within a domain. Examples for controlled vocabularies are catalogues like the one offered by the well-known search engine yahoo<sup>4</sup>. It is used to cluster text collections according

---

<sup>4</sup>[www.yahoo.com](http://www.yahoo.com)

to predefined terms. A more bio-medicine related example is MeSH (Medical Subject Headings), see 12.3.2 for details. It is used to index the MEDLINE/PubMED® database<sup>5</sup>. The key idea is that everyone that uses the terms of a controlled vocabulary means the same thing.

**Thesaurus:** A controlled vocabulary with additional semantic information like synonymy or antinomy is often referred to as a thesaurus. An example of a thesaurus is the *International Classification of Diseases and Related Health Problems* (ICD)<sup>6</sup>. It is the international standard diagnostic classification of diseases maintained and published by the World Health Organisation (WHO<sup>7</sup>). The classification has its origins in the 1850s and meanwhile its 10th version (ICD-10) with 12,420 codes is available.

**Taxonomy:** A taxonomy (or hierarchical classification) extends the set of thesaurus terms in a way that specifies generalisations and specialisations between these terms, usually by *is-a* and sometimes even by *part-of* relations. Note, that the *is-a* relation is a relation between concepts. In contrast to *instance-of*, which is a relation between concepts and instances. The resulting structure can be displayed graphically as a tree or a graph. The words represent the nodes and the relations connect the nodes.

There are several taxonomies in use. One example is the EC classification of enzymatic reactions in the field of bio-chemistry. For a defined set of reactions each of them can be associated with a special EC number and these EC numbers are ordered in a tree-like structure. The more specific a certain reaction is, the deeper it is embedded in the structure. Another example is the *gene ontology* for the classification of gene functions. GO is described in more detail in section 12.3.2 in the paragraph *Gene Ontology*). The most well-known examples might be biological examples, where living organisms are classified according to their species and kingdom<sup>8</sup>.

**Ontology:** An ontology is the semantically richest of these structures. It gives a semantically consistent specification of concepts, their subconcepts and their properties as well as the relations between them.

To give an example, the specification of the concept *atom* represents all atoms. Specific atoms like a hydrogen or carbon atom in a specific protein are instances of this concept. These are usually not part of the ontology. A concept (sometimes also called class) can have subconcepts being more specific than the superconcept related through *is-a* relations. As an example the periodic table of atoms is divided into metals, non-metals, or semi-metals. A subconcept can be seen as

---

<sup>5</sup>PubMed (PubMed (2001)) is a bibliographic database covering life sciences with a biomedical focus, comprising around  $14 \times 10^6$  articles and is growing with around  $6 \times 10^5$  new articles every year, <http://www.ncbi.nlm.nih.gov/PubMed/>.

<sup>6</sup>It is online available through: <http://www3.who.int/icd/vol11htm2003/fr-icd.htm>

<sup>7</sup><http://www.who.int>

<sup>8</sup>See <http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/> for details and examples.

a concept again and thus have subconcepts. In our example metals are divided into alkali metals, alkaline-earth metals and so on. The semantic specification for each concept assigns properties (sometimes also called roles) that they share and which distinguish them from other concepts. For example the class of metals has the property of being good conductors of heat and electricity. Alkali metals in contrast to other metal are very reactant and do not occur freely in nature. Relations (also called role restrictions) connect these concepts (or sets of entities) where each relation brings about type restrictions for the possible values. An n-ary relation connects n concepts. For example the value of the binary relation called *atomic number* associates for each atom one specific atomic number between 1 and 108. The atomic number for hydrogen is 1 and can thus be represented as *atomic\_number*(H, 1). Other properties that can be formalised analogously are atomic weight, or valence. Relations basically categorise entities and connect them. As already mentioned the insertion of instances turns the ontology into a knowledge base. This is mainly carried out by creating individual instances of the concepts in the hierarchy. Following the above example this could mean collecting experimental data and adding them to a model capturing chemical reactions. Each experiment can be added with a specific ID, a reaction equation with substrates and products. Each of them is a set of instances of the concept atoms or of the concept ion.

According to the preceding paragraphs I will use the term “ontology” such that it fulfils the following four requirements. An ontology:

1. has an underlying taxonomy that reflects a controlled vocabulary, and
2. each term within the controlled vocabulary is a well-defined concept or sub-concept with distinguishing properties, such that
3. these concepts are connected via relations.
4. The given criteria can be extended by use of axioms and inference rule. This might be necessary for generation of new knowledge from existing knowledge.

In order to formally present the ontology in chapter 13 I chose to use first order predicate logic (FOL). The main reason is that predicate logic allows representing complex knowledge and is a well understood formal language with a well-defined syntax and semantics. FOL distinguishes between predicates and individuals which allows distinguishing between individuals and their properties. In addition the use of variables ranging over all individuals in combination with quantifiers we can write axioms and rules allowing to generalise over sets of individuals.

According to the above described requirements one can define an ontology as a pair  $O = (V, F)$  consisting of a set of concept and relation names, the vocabulary  $V$ , and a set of FOL formulas or axioms  $F$  representing the taxonomic hierarchy and the

definitions of the concepts and relations.

To illustrate such a formal description a reduced ontology  $O_{ex}$  is presented using FOL as example in 12.1.  $O_{ex}$  shows the above described properties of atoms containing the distinction into metals, non-metals, and semi-metals as well as the property that each atom has an atomic-weight<sup>9</sup>. Hydrogen is added as an atom, too.

$$(12.1)O_{ex} = (\{atom, hydrogen, metal, non\_metal, semi\_metal, atomic\_weight\}, \\ \{\forall x[atom(x) \leftrightarrow (metal(x) \overset{!}{\underset{\vee}{\vee}} non\_metal(x) \overset{!}{\underset{\vee}{\vee}} semi\_metal(x))], \\ \forall x[hydrogen(x) \rightarrow atom(x)], \\ \forall x\exists y[atom(x) \rightarrow (atomic\_weight(x, y))]\} \\ \})$$

The first line of this example asserts that we have 6 predicate symbols. The first and second formula concern the taxonomic hierarchy and assert that each atom is equal to either a metal, a non-metal or a semi-metal and that hydrogen is an atom. The last formula expresses the fact that each atom has an atomic weight. This formalisation allows deriving that hydrogen is an atom and thus has an atomic weight.

### Types of ontologies

According to Guarino (1998) four different types of ontologies can be distinguished: *Top-level ontology*, *Domain ontology*, *Task ontology*, and *Application Ontology*. The following picture shows how they are proposed to be inter-related. Note that the arrows point from the less general to the more general type of ontology.

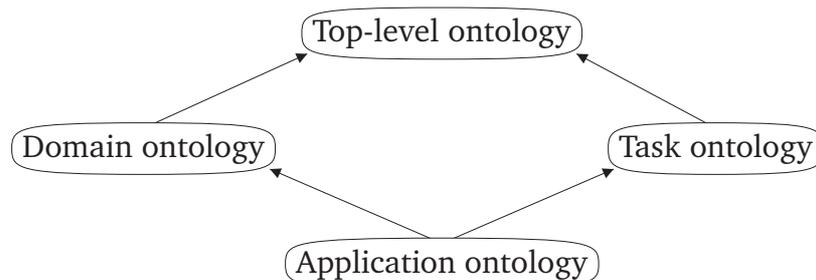


Table 12.1: Different types of ontologies

Following Guarino (1998) the distinguishing criteria for Fig. 12.1 is the level of generality. The most general concepts are to be found at the top of this classification. The

<sup>9</sup>  $\overset{!}{\underset{\vee}{\vee}}$  is used as equivalent for the well-known xor operator:  
 $x \overset{!}{\underset{\vee}{\vee}} y$  is true iff x or y is true, but not both.

**top-level ontology** (the term *upper ontology* is frequently used synonymously) specifies ubiquitous concepts like space, time, event-structure, physical and abstract objects etc. To a great extent they are independent of the domain. Examples are DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering, see Gangemi *et al.* (2002)) as an upper ontology, Benthem (1983) on temporal structure, or James Allen on event structure in Allen and Ferguson (1994), Casati and Varzi (1994) and Casati and Varzi (1997) on spatial structure, or complete top-level ontologies like Sowa (1999) or the Cyc top-level ontology Cycorp (2001a). Note that one and the same top-level ontology can serve for various domain specific ontologies.

The **domain** and **task ontologies** specify the concepts for a domain (like biology, medicine, technical facilities etc.) or a specific task solving model (for large-scale protein experiments, for diagnosis in the medical area, for maintenance of technical facilities etc.). The domain and task ontologies thus specify and use the concepts given in the top-level ontology. An example for a domain ontology is the Unified medical language system, UMLS (see US Dept of Health and Human Services (2002)).

The bottom level is represented by the **application ontology**. The guiding principle for application ontologies is the focus on the special application that they will be used for.

When talking about the details the exact boundaries between domain ontologies and task ontologies are not perfectly clear to me. Each domain ontology has to be designed for a special question or task. Task ontologies and domain ontologies overlap or at least are strongly dependent. The distinction between top-level ontologies and domain- and task-ontologies is more obvious since general concepts like time points, events, processes, states, locations occur through hardly all possible scenarios. Concerning the application ontology a clear cut can be made since this part of the ontology will surely differ from application to application. A database will need a different conceptualisation than a NLP system needs. The ontology which is presented in the following chapter is a domain ontology. Some interactions with top-level ontologies are used but not made explicit as such.

## 12.2 Criteria for building an ontology

Generally there exist few standards for the formal characteristics and design of an ontology. The following presents some criteria that can be important for building an ontology. I do not discuss the question of ontology engineering and how to develop an ontology from scratch. The focus is more on formal considerations for building an ontology. As an example, what has to be considered with respect to the representation language for expressing an ontology.

1. **Expressivity** of the language and coverage of the implementation:  
Each conceptualisation is expressed through a representation language. A representation language should allow to precisely characterise the knowledge about some domain of interest. Depending on the richness of the language – the logic symbols it offers like conjunction, disjunction, negation, variables and quantifiers

– one should be able to answer questions like *To what degree is this language able to represent the relevant concepts and relations?*

At the moment there are various representation languages in use that offer different expressive power. A brief list of representation languages is presented pointing out the different strengths:

Difference between Frame-based systems and Description logic?

*Ontolingua* - developed at Stanford University - is based on the Knowledge Interchange Format (KIF, see citekif). *KIF* is a language which was specifically built for the sharing of knowledge among different ontologies. The semantics of *Ontolingua* are based on the frame knowledge representation systems.

*Cycl*, is maybe the best-known ontology language since it is the underlying language within the development of Cyc (see chapter 12.3.1 for details about Cyc).

*Conceptual Graphs* is a graphical notation and designed to allow conceptualisation in a more readable form. *CGs* have same expressive power as *KIF* and as classical first-order logic.

The surely most popular ontology language at the moment is the description logic based *Web Ontology Language* (OWL). It is mainly developed to publish and share data on the Internet. OWL is a of the *Resource Description Framework* (RDF) and is derived from the DAML+OIL Web Ontology Language. Together with RDF and other components, these tools are the main components of the Semantic Web project. More and more OWL is getting the standard ontology language. Mainly because the OWL specification is maintained by the World Wide Web Consortium (W3C).

2. **Complexity** and **efficiency** issues concern pragmatic issues of system requirements – like processing time and memory capacity – for computing according to the inference algorithms<sup>10</sup>. In Levesque and Brachman (1987) the authors examine the relation between expressive power of a language and its computational tractability. One important conclusions is, that the more expressive a representation language is the more difficult – computationally expensive – it is to reason. As a consequence depending on the problem to solve some representation languages can be more efficient than another.

### 3. Inference and Interpretation:

- What kind of inferences will be necessary? There are three inference methods that can be applied. First, there is **deduction**. It is the traditional logic inference method that derives new knowledge from existing facts. It is an exact inferencing system. The second type is **induction**. The idea behind induction is to learn generalisations from sets of data. It is a method where the premises of an argument support the conclusion but do not ensure it. The third inferencing method is **abduction**. It may be described as finding plausible causes of an effect.

---

<sup>10</sup>The relevant questions are: **How long** does it take in the worst case scenario to infer one or all correct answers? **How much memory** does it require in the worst case to infer one or all correct answers?

- Given the **correctness** (or **soundness**) of an inference system no false conclusions can be drawn. As an example if my system is sound and answers 'yes' I can be sure that the answer is correct. On the other hand a correct inference procedure can not assure that there is no answer in case it fails to find one. Correctness is sometimes also referred to as Coherence as well.
- **Completeness** of the inference system means that all correct conclusions that can be drawn are really inferred by the system, *e.g.* if there is a possibility to answer *yes* my system does so.
- **Decidability** of an inference system is given when I can be sure that there exists an algorithm which ensures that the inferencing terminates with *yes* or *no*.

In this context the *closed world assumption* is relevant, it says that if one cannot prove  $P$  or  $\neg P$  from a given set of rules and axioms, one can add  $\neg P$  to the knowledge base. Two scenarios where it can be reasonable to be used are: First, given a database with a complete set of data – like data about employees with salaries etc. – and a set of rules that can be used for inferencing. In such a case one assumes that one has all facts stored and thus one cannot infer an additional fact  $P$ . Another case is that given that one does not have a complete set of data but one needs to obtain an answer. If one can neither prove  $P$  nor  $\neg P$  than one can assume that it is false ( $\neg P$ ). It is important to note that whenever working with incomplete data in combination with a closed world assumption this can lead to wrong conclusions, *i.e.* if one does not have  $P$  in my database it does not necessarily mean that  $\neg P$  can be added to the database.

The *best* inference system is – of course – sound, complete, decidable and works for an adequate expressive language. But it is not possible to satisfy all these desires. For example it is often a computational expensive procedure to ensure completeness. Details about the formal properties and dependencies between the mentioned issues are presented in Nardi and Brachman (2003).

4. **Readability, clarity and visualisation** as well as **documentation** of the concepts and relations are important for transparency reasons allowing the user to overview the ontology in a comprehensible way. While readability concerns the naming of the concepts and relations<sup>11</sup> documentation concerns natural language definitions of the concepts and relations to avoid misunderstandings. Clarity refers to the consistency of the guiding principles behind the model. And, visualisation can be of help reducing the complexity of the model to enable the user to obtain an overview showing the overall hierarchy. Frequently ontology editors offer a graphical tree-like structure showing the conceptual hierarchy.

5. **Merging and integrating** data (also referred to as scalability)

This paragraph deals with the question of merging data from different concep-

---

<sup>11</sup>A consistency in the naming procedure makes it much easier to "read" the ontology. Related terms should be also related through the name, *e.g.* the concept *DNA-binding-domain* shows already through its name the connection to *DNA*, and to *binding-domain*.

tualisations. Different models may use different names for the same concept and even worse the same name for different concepts. The task of proving that two concepts refer to the same entity is difficult and can usually not be answered by automatic processing. Drawing attention to using standardised representation language and terminology can help avoiding syntactic incompatibilities. However this does not solve the semantic problem when merging ontologies. Most work on integrating different knowledge sources remains to be manual work. As an example in Burgun and Bodenreider (2001) the authors report on the analysis of compatibility between *UMLS* (see further down paragraph *UMLS*) and *Cyc* (see further down paragraph *Cyc*) and *UMLS* and *WordNet* Fellbaum (1998a) the following two points:

- Categories that have similar names in different ontologies may have distinct meanings. For example, Entity, Body Part, Body Region do not have the same meaning in *Cyc* and in the *UMLS*.
  - Moreover, two categories may have similar intensions while neither their respective classes are identical nor is one class totally included in the other, as illustrated by the extensions of Symptom in *WordNet* and in the *UMLS*.
6. Although **extendibility** is related to integration and merging of data it is a bit different. Extendibility deals with the fact that new data (*i.e.* more taxonomical or even more axiomatical information) are added to an existing ontology. However this means that an existing ontology should already have points of contact to the new data. The concepts should not have been defined in such a way that adding of new concepts requires revision and re-definition of the existing concepts.
  7. The ability to conceptualise the world at different granularities and to move from one level of granularity to the other is a major requirement. **Granularity** becomes thus an important issue for ontology building when modelling concepts according to different necessities. The main challenge is to keep the model consistent and to allow to switch from one perspective to another at the same time. Granularity is important as well when merging different ontologies. Different ontology builders may have different perspectives on the same thing and this makes the merging so difficult. To show some different perspectives – or, different levels of granularity – I chose the example of a protein within a signal transduction cascade:

A protein can be regarded as a (component acting as a) signal within a signalling pathway traversing specific states. It can be regarded as a member of a protein family, although being a specific protein, generalising over certain characteristics that each member of the family has. Of course one can regard the specific members of the protein family and thus regard the differences between the members. And, even more in detail, one can regard the specific members of a protein family within different organisms. We can regard a protein as being a chain of peptides having special characteristics like being hydrophobic or not, depending on the individual amino acids that form the peptide chain. Or, we can even look at it as a

macromolecule consisting of atoms having certain domains and thus being able to perform certain chemical reactions, for example according to some enzymatic classification. In what follows I go through an example (*i.e.* the Jak/Stat pathway) to illustrate some of these facets and different levels of granularity.

Regarding the JAK-STAT pathway (see Aaronson and Horvath (2002) for details) one can identify four levels of granularity concerning the participating proteins. The JAK-STAT pathway can in short form be described as follows. First, the binding of a ligand to a cytokine receptor at the outside of the cell causes two of these receptors to form dimers. These dimers activate Janus kinases (*Jak*) through self-phosphorylation. This self-phosphorylation causes certain tyrosine (Tyr) residues on some Stat (*Signal Transducer and Activator of Transcription*) proteins to be phosphorylated. As a consequence the Stats then form dimers. Only dimerised Stats are allowed to enter the nucleus, where by binding to specific binding sites of the DNA strands gene expression is initiated.

- One level of granularity concerns the view which reduces proteins to signals within the signal transporting process. A cascade of chemical reactions is reduced to signals traversing certain states. A state can be described by the localisation of the signal, some modifications (*f.ex.* activation) and the binding relation to other signals. Signals can cross the cell membrane, through the cytoplasm and finally enter the nucleus to initiate gene expression. In principle this signal need not be a protein. It can also be some other type of molecule.
- Another level of granularity concerns generalisation over the Jak and Stat proteins independently from the organism they occur in or independently from the special type of Jak or Stat protein. This level refers to the general protein Jaks and Stats as members of the corresponding protein families.
- A more finer grained view leads to the observation that certain specific members of the mentioned families interact in contrast to other pairs. In mammals for example there are seven different types of Stat proteins (*i.e.* Stat1, Stat2, Stat3, Stat4, Stat5A, Stat5B, and Stat6) and four types of Jak proteins in mammalian cells (*i.e.* Jak1, Jak2, Jak3, and Tyk2). However not each of the Stats interacts with each of the Jaks.  
Analogously, within the dimerisation step of the Stat proteins we can either talk of homodimers – meaning two Stats bind to each other – or we can talk of heterodimer – meaning a heteromer of Stat1 and Stat2 proteins –.
- Another level concerns the differences across organisms. First, there are organism specific Jak and Stat proteins, like Jak-3\_mouse or Jak-3\_human. If two same types of Jak (*e.g.* Jak-3) occur in different organisms that does not mean that they are the same. Jak-3\_human has for example a length of 1124 amino acids and a molecular weight of 125015 Da. In contrast to that Jak-3\_mouse has a length of 1299 amino acids and a molecular weight of 144314 Da.

- Regarding a protein as a polypeptide means looking at a protein as consisting of a chain of peptides. In the above example this means that depending on the STAT protein the length of the chain ranges between 750 and 850 amino acids. The properties of the whole is the sum of the properties of each protein with respect to a kind of composition. This means that if a protein has a significant amount of hydrophilic peptides brought together through folding than this part of the protein will be hydrophilic, too. Another fact which is important in this context is that although proteins in fact are polypeptides not all polypeptides are proteins. This is evident since you can take some repeat sequence from a protein and this is not a protein anymore.
- From the perspective of a more chemistry oriented biologist a protein can be regarded as a special type of chemical compound that has specific chemical properties and is thus responsible for some canonical reactions which are classified through enzyme classification system. Or, looking at the protein domains, with a specific domain a specific chemical reaction is associated. In the case of the Jak proteins, there are seven homology domains identified in the C-terminus to N-terminus direction. While the JH1 domain is well understood (JH1 is the tyrosine kinase domain) the functions of the other domains are not very well understood.

Generally speaking, when building an ontology one has to be clear about which levels need to be represented in an ontology for suiting a specific purpose. If the ontology comprises different levels of granularity the more general levels must be reflected through the most detailed levels.

Following Hobbs (Hobbs (1985)) granularity can be defined by an indistinguishability relation  $\sim$ . Two entities  $x$  and  $y$  are defined indistinguishable iff wrt a specific scenario for all relevant predicates,  $p(x)$  is true iff  $p(y)$  is true. This means that we conceptualise the world (*i.e.* the relevant domain) at different levels of granularity. We can also say that we build equivalence classes for each level of granularity. In a particular scenario we thus only distinguish those things being relevant for that scenario.

8. **Learning ontologies** There exist several benefits that justify developing an ontology. Some have been mentioned. However, the major bottleneck when building ontologies manually is the combination of knowledge acquisition and time-consuming construction of various ontologies for various domains and for various applications. One solution is moving towards automatic generation of ontologies. Although several groups have presented promising results in the past (see Ciarmita *et al.* (2005) and Maedche and Staab (2000) as exmamples), none of the system is able to generate an ontology that fulfils all above mentioned properties. Usually ontology learning systems can provide large-scale conceptualisations that need to be tuned manually to meet the desired requirements. Nonetheless, this semi-automatic approach is promising and can at least reduce the amount of manual work that needs to be invested when building an ontology from scratch.

## 12.3 Existing ontologies and taxonomies

Currently there are more than 100 projects and groups developing ontologies<sup>12</sup>. In the following some representative ontology projects are presented. These are: Cyc, which represents the biggest effort in this domain at all, WordNet, which represents a linguistic ontology and some bio-ontologies. There are mainly two reasons for choosing these ontologies. First, it is important to notice the variety of ontologies and the amount of information that is already implemented. Second, ontologies are built and really in use in a variety of scenarios. Further, the ontologies differ very much. This is mainly motivated by the different application scenarios. This always leaves the possibility of adapting existing implementations to fit a different application scenario.

### 12.3.1 General and linguistic ontologies

This section briefly presents the general knowledge repository Cyc and the linguistic ontology WordNet. Besides these two representatives there exist other approaches as well. However these two can be assumed the most prominent ontologies that have emerged related to natural language processing in the past two decades.

#### Cyc

The Cyc<sup>13</sup> project (see Cycorp (2001a), or Sanguino (2001) for an overview report) is the longest still ongoing project in this field. Within this project the overall goal is to build the basis for large scale symbolic reasoning and to capture human knowledge needed in everyday life (also called consensual knowledge or common sense). The project was running at MCC (Microelectronics and Computer Technology Corporation) in Austin, Texas, since 1984 attempting to build a 'universal' expert system. Cycorp is a spin-off since 1995 of MCC. They represent their knowledge in a form of predicate logic known as CycL (see Cycorp (2001b)).

The Cyc upper-level ontology represents over a decade's work of research on how to represent human common sense. The latest version of the Cyc Upper Ontology is available through <http://www.cyc.com> allowing for browsing or downloading. From the *tens of thousands* ontological concepts approximately 3,000 are available for free download. Extensions of this *upper Cyc* are available as commercial extensions. Besides the Cyc Upper Ontology there exist several sub-ontologies for biology, chemistry, physiology, general medicine, materials, waves etc. Inferencing in Cyc in general subsumes deduction techniques like modus ponens, modus tollens, and universal and existential quantification. Some special modules come with special reasoning techniques, e.g. temporal or mathematical reasoning.

Each concept comes with three features, (i) its name, (ii) an English comment describing the intended meaning, and (iii) a list of links to other concepts to get the concepts hierarchically structured.

---

<sup>12</sup><http://www.cs.utexas.edu/users/mfkb/related.html> offers an overview.

<sup>13</sup>The name Cyc derives from the word encyclopedia.

To give an example the following is taken from the freely available part of the chemistry sub-ontology:

```
#$catalyst : <#$ChemicalReaction> <#$PartiallyTangible>
```

The predicate `#$catalyst` identifies the particular thing that acts as a catalyst in a particular chemical reaction. (`#$catalyst R X`) means that the `#$ChemicalReaction R` has the particular quantity of substance `X` as a catalyst. For example, every instance of `#$Photosynthesis` has some portion of `#$Chlorophyll` as a catalyst; an amount of `#$Water` may be a `#$catalyst` in some `#$OxidationProcess` of a `#$Metal`.

```
isa: #$ActorSlot #$AsymmetricBinaryPredicate #$IrreflexiveBinaryPredicate
genlPreds: #$unchangedActors
```

The Cyc project comprises NLP efforts within the Cyc-NL system. The Cyc-NL system consists mainly of three components: a lexicon, a syntactic parser and the semantic interpretation component. A lot of effort is currently spent on broadening the coverage of these three components.

## Wordnet

WordNet (see Fellbaum (1998b) for details) is perhaps the most used natural language processing resource at the moment. Since 1985 it has been steadily extended at the Cognitive Science Laboratory of Princeton University. WordNet is a lexical database with semantic relations between the so-called synsets. A synset is a set of synonyms like the set { motor vehicle, automotive vehicle }, which is described as “a self-propelled wheeled vehicle that does not run on rails”. Each synset is composed of English nouns, verbs, adjectives, and adverbs and represents one underlying lexicalised concept. To give an impression of the size of WordNet the following table shows the amount of tokens (*i.e.* the lexical entries), synsets (*i.e.* the synonyms set) and senses (for example, WordNet has three senses for vehicle listed, *i.e.* (i) a conveyance that transports people or objects, (ii) a medium for the expression or achievement of something, and (iii) any inanimate object (as a towel or money or clothing or dishes or books or toys etc.) that can transmit infectious agents from one person to another) for nouns, verbs, adjectives and adverbs.

The semantic relations that are captured comprise hypernymy/hyponymy, meronymy/holonymy, antonymy, synonymy, homonymy, and polysemy.

The lexical resource together with some tools is free available through <http://wordnet.princeton.edu/>.

Token type	no. of token	no. of synsets	no. of senses
Noun	114,648	79,689	141,690
Verb	11,306	13,508	24,632
Adjective	21,436	18,563	31,015
Adverb	4,669	3,664	5,808
Totals	152,059	115,424	203,145

Table 12.2: **WordNet 2.0 statistics:** Overview of the token counts, synsets counts and senses counts for nouns, verbs, adjectives and adverbs.

### 12.3.2 Biomedical ontologies

#### Gene Ontology

The Gene Ontology (GO) project (<http://www.geneontology.org>) organises a network of over 10,000 terms which are related to gene products. One of the guiding ideas was that the use of a taxonomy (a consistent vocabulary) allows comparing genes from different organisms based on their GO annotations. This means that database annotation is the main purpose of GO. Meanwhile GO has developed as an import backbone for annotation through several databases (16 database, on March, 3rd 2004) like FlyBase (a database of the drosophila genome, see The FlyBase Consortium (2003)), Swiss-Prot(Boeckmann *et al.* (2003)), Sanger GeneDB (Hertz-Fowler *et al.* (2004)), etc. The institutions behind these databases collaborate mostly in three ways, which are linking of databases entries or objects to GO terms, support of querying based on GO terms, and by contributing to the development of GO terms by suggesting new entries or refinements of existing entries in GO. However, the criteria which are used to classify GO terms are not always perfectly clear (see Kumar and Smith (2003) as an example).

Since GO offers only a taxonomy it lacks any upper-level ontology. The network is organised in a tree-like structure, where the nodes are connected via graphs. The tree is a directed acyclic graph (DAG) and has three top nodes: molecular function, biological process, and cellular component. Molecular function describes activities like for example catalytic or binding activities, at the molecular level. In contrast to that a biological process is a composition of molecular functions, for example cell growth or signal transduction. Cellular components refer – as the name tells – to components of the cell like nucleus, ribosome or a protein dimer. These top nodes dominate all other nodes through two relations, *i.e.* *part-of* and *is-a*. Most relations between concepts in GO are thus not annotated and remain implicit in the term names. What *implicit* means is described in Ogren *et al.* (2004). The following shows two of the examples from the article:

“... the term *positive regulation of cell migration* (GO:0030335) contains the GO term *regulation of cell migration* (GO:0030334) as a proper substring.”

“... the term *regulation of cell proliferation* (GO:0042127) is derived from the term *cell proliferation* (GO:0008283) by addition of the phrase *regulation of*.”

The authors also describe how this regularity can be used to bring more semantics to GO and use this for NLP

A major advantage is that GO offers the DAG-Editor (DAG = directed acyclic graph). The DAG-Editor is graphical user interface that shows relations between the concept in a transparent and clear way. In addition it allows querying for concept names and to inspect the dependencies by clicking on the concept names.

The GO website offers possibilities to search for concepts and to display the *is\_a* and *part\_of* relations. As screenshot is shown in Figure 12.1.



Figure 12.1: HTML search facility with display of concepts for GO.

### Open Biology Ontologies - OBO

A project highly related to the above mentioned gene ontology is the the OBO (Open Biology Ontologies) initiative, initiated by Michael Ashburner. It is aimed to encourage research organisations to develop ontologies in other biology related fields and make them available for other researchers. This is an ongoing project. A website (<http://obo.sourceforge.net/>) and and FTP server is open for downloads of the OBO ontologies.

The five criteria for being listed in OBO are given on the OBO-website as follows:

1. The ontologies must be **open** and can be used by all without any constraint other than that their origin must be acknowledged and they cannot be altered and re-distributed under the same name.
2. The ontologies are in, or can be instantiated in, a common shared syntax. This may be either the OBO syntax, extensions of this syntax, or OWL.

3. The ontologies are orthogonal to other ontologies already lodged within OBO.
4. The ontologies share a unique identifier space.
5. The ontologies include textual definitions of their terms.

At the time of writing this thesis (31st of May, 2005) the OBO website lodges more than 50 ontologies. The most prominent ones are MeSH (Medical Subject Headings, which are introduced further down within this section) and gene ontology (see above).

## **BioPAX**

BioPAX is an abbreviation for Biopathway Exchange Language and it is a collaborative effort between people from different biology related areas building a data exchange format for biological pathways. The group was established at the Fourth BioPathways Consortium Meeting, a satellite of the ISMB'02 Conference held in Edmonton, Canada in August 2002. One of the main objectives is to facilitate the sharing of pathways information between existing databases. The main benefit for the user is that given a common exchange format the heterogeneity of data through various databases is reduced and communication can be facilitated through this common exchange language. The pathway types that are supported are metabolic pathways, signalling pathways, protein-protein interactions, and genetic regulatory pathways. The envisaged users are tool developers, database curators or researchers in the biological domain. The Biopax group offers their results for download through <http://www.biopax.org/> and has installed a mailing list for discussion of the results as well. Figure 12.2 shows the main BioPAX ontology with a root element *entity* (level 0) that has three complementary entities as subparts. The subsumes on level 1 are either *physical entities*, *interactions*, or *pathway*. The following level shows for *physical entities* small molecules, proteins, RNA or complexes. *Interactions* are defined as a set of entities and relations between them. This corresponds to the linguistic notion of relational words, like verbs and some nominal phrases.

The work of the collaboration is divided into three subgroups:

- The *Examples Subgroup*, which gathers sample data from various sources to illustrate use cases and promote practical development.
- The *Small Molecule Subgroup* is a subgroup that are transferring small molecule information in BioPAX. This objective was established as a consequent from the evaluation of the CKL 2.0 version.
- The *State Subgroup* is working on methods to represent biological states (e.g. post-translational modifications, cell-cycle stages, etc.).

## **UMLS**

The Unified Medical Language System (UMLS) – freely available for research through <http://www.nlm.nih.gov/research/umls/> – is under development at the U.S. National

Library of Medicine since 1986. It is a knowledge source that is created through merging various sources. The 2003 edition merges more than 60 sources, with more than 900.000 concepts and around 2.5 million concepts names. It contains an associated lexical program for developer (f.ex. for use within NLP applications). The UMLS consists of three parts, i.e the Metathesaurus, the Semantic Network, and the Specialist Lexicon.

- The *metathesaurus*, which contains semantic information about biomedical concepts, their various names, and the relationships among them integrates various thesauri and classifications developed and maintained by many different organisations. In the 2003 edition it contained more than 1,5 million terms.
- The *semantic network* is a top-level ontology of biomedical concepts. It is constructed in order to structure the concepts in the *metathesaurus*. All concepts in the Metathesaurus thus were assigned a concept from the *semantic network*. In the 2003 edition, 134 top-level concepts and 54 relationships were to be found in the *semantic network*.
- The *specialist lexicon* contains syntactic information about biomedical terms (like subcategorisation information) and covers a major part of the concept names in the Metathesaurus.

Users of the UMLS have developed different tools that can be used with the UMLS Knowledge Sources. The following shows a list of tools which are offered through UMLS with short descriptions:

- MetamorphoSys Information (FAQ, advanced user instructions, etc.)
- Lexical Tools
- Norm
- CUI History
- MeSH Browser
- Sample Load Scripts

#### MeSH

MeSh abbreviates Medical Subject Headings and is a controlled vocabulary and thesaurus maintained and provided by the U.S. National Library of Medicine. On March, 3rd 2003 there were 22,568 MeSH descriptors. Additionally there are over 139,000 so-called Supplementary Concept Records in a separate thesaurus. Thousand of cross-references that help finding the most appropriate MeSH heading are included like *Ascorbic Acid* for *Vitamin C*.

The main use for MeSH is the annotation of biomedical journals in the MEDLINE/PubMED database. This makes it possible to retrieve for example all MEDLINE/PubMED articles or abstracts for a special organism. It is used for several other

projects for any kind of annotation as well. Several classes of MeSH terms exist, the most relevant for literature mining being 'Chemicals and Drugs' (MeSH-D) and 'Diseases' (MeSH-C).

### **EcoCyc**

EcoCyc (see Karp *et al.* (2002)) uses an ontology for an organism-specific database that describes the metabolic and signal-transduction pathways of *Escheria coli*. In addition it covers for this organism its enzymes, its transport proteins, and its mechanisms of transcriptional control of gene expression. MetaCyc represents metabolism not specific to *E. coli*.

### **TAMBIS**

TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) should be mentioned as the first ontology based information integration systems in the area of bioinformatics. It uses an ontology based on description logic to enable biologists to ask questions over multiple external databases using a common query interface. It is available through <http://imgproj.cs.man.ac.uk/tambis/>.

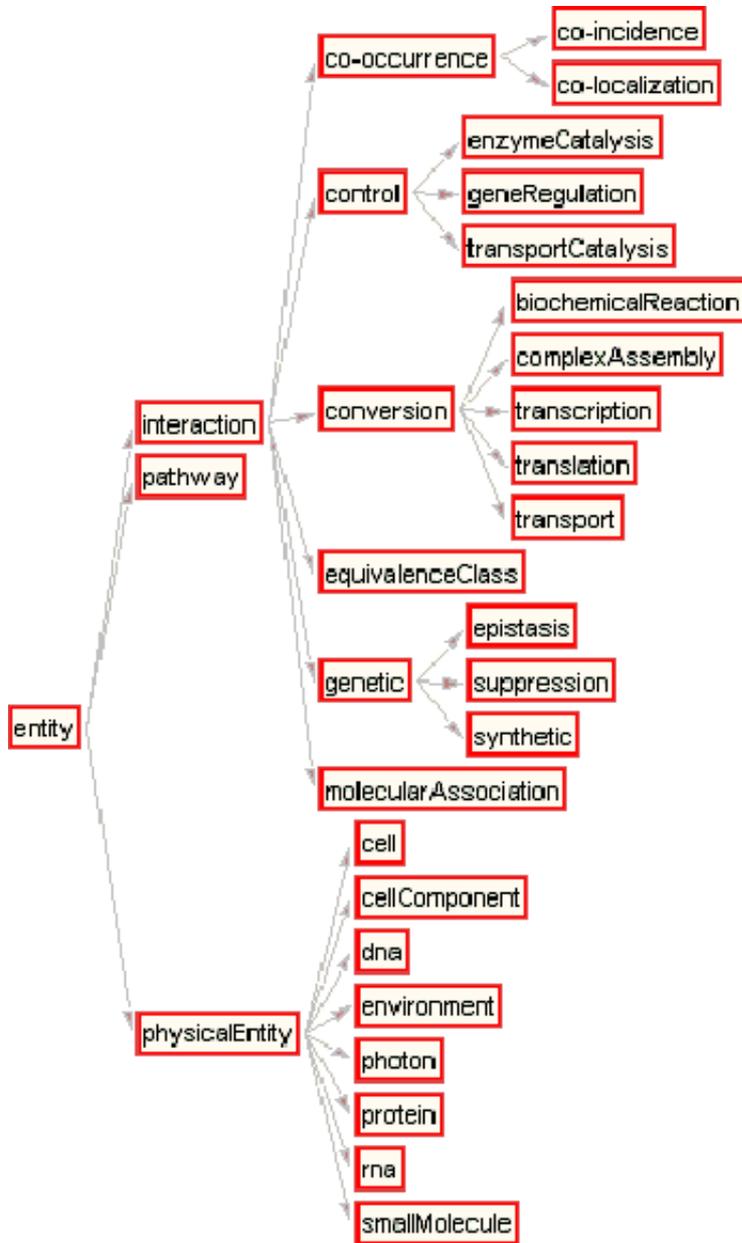


Figure 12.2: BioPAX ontology from Sep, 24th 2003



# Chapter 13

## Building a bio-ontology

As presented above, there exist several ontological approaches within the life sciences domain. In this chapter a conceptualisation is presented which introduces relevant concepts and relations between these concepts and axioms for biochemical entities and biochemical processes related to gene expression. The axioms that are presented and discussed should be considered as a fundamental fragment where key issues in modelling life science ontologies are introduced. The taxonomy, concepts and relations need later additions and changes to derive full deductive power and be applicable in the proposed contexts. However, it is the intention that re-organisation of larger blocks of the given fragment is not necessary. Implementational issues are not considered although it is my aim to describe minimal restrictions in order to avoid excessively complex explanations.

First, in section 13.1, a discussion of some linguistic examples on the use of ontological knowledge is presented to motivate the use of ontologies for Natural Language Processing systems. Other NLP scenarios/applications and applications like database model design have already been mentioned in earlier sections. Section 13.2 presents a rough formalisation of the biochemical sub-domain of gene expression. It introduces and discusses fundamental concepts, relations and properties related to gene expression. The introduced conceptualisation is a further development of the conceptualisation presented in Šarić *et al.* (2004c). The final sections inspect the introduced gene-expression model in detail. These sections focus on questions related to granularity. Mainly, to what extent is it possible to move from general biological aspects to biochemical aspects?

### 13.1 Motivation: Linguistics and Ontologies

The process of extracting gene expression relations by means of a rule-based approach has been presented in detail in the previous chapters. Taking a more ontological view of the system presented, the rules reflect a gene expression ontology in which the domain specific knowledge is hard-coded. We've also seen that such a rule-based system pro-

vides highly relevant results of good quality (a series of experiments has been carried out where all systems are based on the same core is described in Šarić *et al.* (2004a), Reyle and Šarić (2002), Šarić *et al.* (2004b), and Šarić *et al.* (2005b)). It should be noted that a major drawback of this rule-based IE approach is that the writing of rules is highly time-consuming and scaleable only to a limited extent<sup>1</sup>. To overcome this hurdle it would prove extremely useful to develop a system that allows (semi-)automatic interaction between ontological information and a rule-based IE system. The process of extracting information, mainly the process of rule-generation, should be ontology-driven.

Another advantage of an ontology-driven approach is related to what Schulze-Kremer (1998) calls the *communication problem in molecular biology*. To understand this argument it has to be noted that biology-related IE is usually carried out to provide database entries for further processing. In principle the same ontology can be used to design the database schema. Thus an ontology-driven approach – for both the IE system and the semantic design of the database – can ensure that consistency of data is much more likely than with other approaches.

As for the system architecture of an ontology-driven system it has to be mentioned that such an approach is based on compositional principles. More details of such a system are described in Cimiano *et al.* (2004). Following a compositional approach, one has to distinguish between syntactic and semantic processing. Note that this is in contrast to the IE system presented in this thesis. In a compositional system each syntactic processing rule has a corresponding semantic rule that assigns the correct interpretation of the syntactic structure. Given a syntactically processed and annotated sentence, the meanings of the individual words are looked up in the lexicon and associated with the corresponding ontological concept. In a subsequent step these are recursively recombined to form the meaning of the whole sentence. In what follows, I focus mainly on two components to explain what is meant by an ontology-driven IE system, *i.e.* a conceptual component and a relational component.

The main goal of a conceptual component within an IE system is to link and associate the relevant lexical entries with the corresponding concepts in the ontology. To give an example, a noun phrase that is recognised as referring to a protein, like *the GCN4 activator protein*, should not only be associated with the general concept *protein*, but it should be linked to the even more specific concept *transcription activator protein (R)*, which is a specific type of *protein*. Of course it inherits all properties from *protein*. In addition, it has some further properties like the potential to bind to a specific part of a promoter and enhance the expression of the corresponding gene. *R* should be instantiated with the specific protein *GCN4*.

The main objective of a relational component is to link and associate the relevant relational words with the corresponding relations in the ontology. A relation in the ontology links concepts for which this relation holds. Similarly, a verbal or nominal relation

---

<sup>1</sup>Although the follow up experiments mentioned are all based on the same named entity detection rules, the system can only be adapted to related questions, for example the step from protein-gene interactions to protein-protein interactions is manageable with little effort. Nonetheless the whole system would need a completely new dictionary and a new set of rules if it had to extract disease-drug relations.

within a sentence links nominal entities, *i.e.* the arguments of the relational word. An example like (13.1), shows the noun phrase *the ATR1 promoter region* that is associated with the ontological concept *promoter*, which is a part of the non-coding sequence of a specific gene *X*. This gene *X* should thus be instantiated with *ATR1*. On top of this conceptual component the binding relation should be associated with the *bind* relation in the ontology. An ontology modelling relevant biological relations encompasses several binding relations. As an example a binary relation  $bind(x,y)$  may link two concepts *chemical compound*, *i.e.* binding between two chemical compounds. This means that both arguments of the binary relation  $bind(x,y)$  select entities of type *chemical compounds*. Another type of binding relation selects *protein* for both arguments referring to protein-protein interaction. Analogously the given example shows that one binding relation selects one entity of type *protein* and one entity of type *DNA* or *gene*.

One can expect to benefit in two ways from such an approach. One benefit is with respect to consistency and completeness in that each rule in such an IE system has a semantically well-defined correspondence in the ontology classifying the type of information detected. A second benefit is that one can infer new extraction rules based on links or associations which have already been established. In the following I illustrate these principles with the help of some examples.

(13.1)The GCN4 activator protein binds to the ATR1 promoter region.

(13.2)The ATR1 promoter region contains a binding site for the GCN4 activator protein.

(13.3)The binding of GCN4 protein to the SER1 promoter in vivo...

Example (13.2) shows a syntactic variant expressing the same fact as (13.1). The advantage of a compositional and ontology-driven approach is that it allows the treatment of this different syntactic construction in the same way and thus the extraction of the same information. Of course, one cannot judge whether example (13.3) is about activation or repression of gene expression. However, a concept *bind* which relates proteins and genes is underspecified with respect to activation or repression. And the example does not mention whether GCN4 is an activator or a repressor of the gene expression. A system like the one proposed would thus come up with a neutral regulation relation.

(13.4)Endotoxin increased NF-kappaB p50/p65 heterodimer binding.

Some more examples that show the basic need for ontological knowledge have already been given in the section 3.1 *Naming in Biology*. Example (13.4) illustrates that in some cases combinations of different semantic issues are concerned. One issue concerns coordinations and appositions. A second issue concerns presuppositions. To somehow determine the meaning of *NF-kappaB p50/p65 heterodimer* one has to know what heterodimer means. A lexical entry bears information presupposing the existence of two entities *A* and *B* of type *protein*, and  $A \neq B$ . Concerning coordination, a common system would try to identify these two entities and associate *A* with *NF-kappaB* and *B* with

*p50/p65*. However this conclusion would be wrong. To determine the meaning correctly the system needs to know that the slash symbol is used in publications in the field of molecular biology to express conjunctions. However, more important is that *NF-kappaB* assigns the type of protein to both *p50* and *65*. This knowledge is the key for correct semantic processing. Otherwise the correct answer that  $A = p50$  and  $B = p65$  cannot be inferred.

## 13.2 Basic gene expression concepts and relations

As already explained above, gene expression is the process of obtaining a **peptide** or an **RNA** molecule from a **gene** or an **operon**. The first stage of this process is termed **transcription**, where the gene or operon is transcribed into an RNA molecule, which is also referred to as the **transcript**. If the RNA molecule is of type mRNA the transcript is then **translated** within the next step into a peptide. In addition to being one of the possible products of the translation step, proteins control and steer the processes of transcription and translation. This conceptualisation was already introduced in section 4.2 and is summed up in figure 4.2. Figure 13.1 illustrates the process of gene expression. The figure shows a simplified model that presents the basic concepts and relations which are treated formally within this section. The concepts presented in this section are coarse grained and avoid excessive detail, and some of them will be treated within subsequent sections.

### 13.2.1 Taxonomy and basic concepts

The main conceptual members which join the gene expression process are **genes** as the carrier of information, **gene transcripts** as the results of transcription, and **peptides** being the product of the translation step.

#### Composing and de-composing with the fusion operator

Before going into the details of the formalisation of gene expression I introduce the fusion operator  $\oplus$ , the part-of relation  $\sqsubset$  and the precedence operator  $\prec$ .

**Fusion:** For the composition of a larger whole from a set of entities the fusion operator  $\oplus$  is introduced. To express that given a set of things  $x_1 \cdots x_n$  the whole entity  $X$  is made up of these things and nothing else I write  $X = x_1 \oplus \cdots \oplus x_n$ . In the literature this is usually referred to as the mereological sum (or fusion) of these parts.

**Part-of:** In contrast to the  $\oplus$  operator, the  $\sqsubset$  relation is used to indicate part of relation that holds between a whole and its parts. It allows the decomposition of a whole and reference to its specific parts. From the above given example one can infer that  $\forall x_i \sqsubset (x_i, X)$ . Thus, each  $x_i$  that is part of  $X$  can be formally referred to as

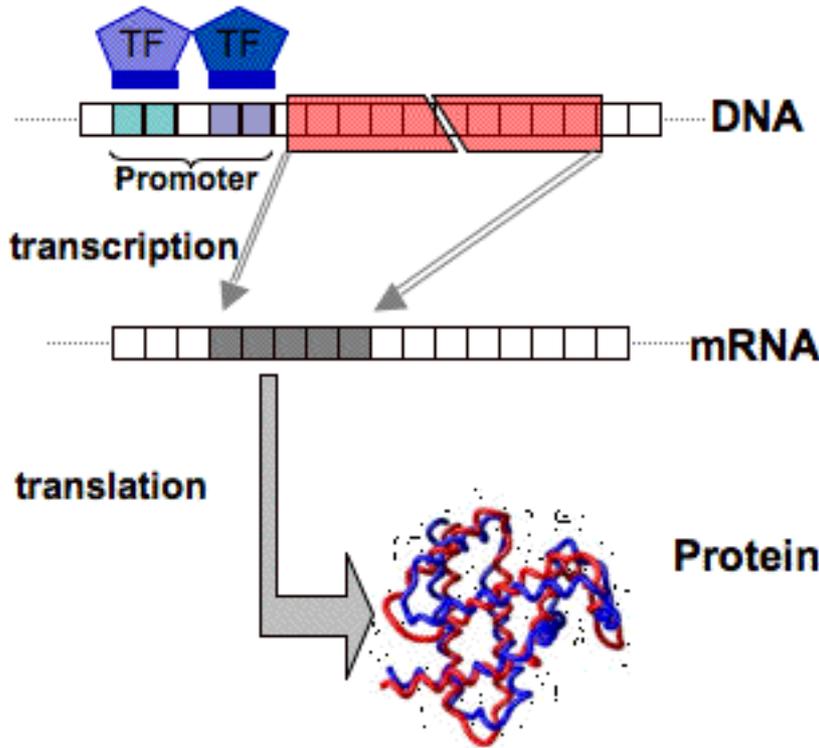


Figure 13.1: **Transcription and translation process.** The picture shows the transcription factors (TF) binding to the promoter of a gene. The gene (is shown as a part of the DNA) is transcribed into a mRNA (other types of transcripts are not shown). Then translation to a protein takes place. As a result a protein is shown.

such. It should be noted that the  $\sqsubset$  relation is transitive. This means whenever  $\sqsubset (a, b)$  holds and  $\sqsubset (b, c)$  holds that  $\sqsubset (a, c)$  can be inferred.

**Precedence:** In addition I use the binary precedence relation  $x \prec y$  to impose an order between any two entities  $x$  and  $y$ , meaning that  $x$  precedes  $y$ .

### Event variables and event structures

Verbs and some nouns denote events or states. Frequently they're also referred to as occurents in contrast to continuants, which is used as synonym for entities. Events typically bring about changes, which means that before some event  $e$  takes place there exists a state  $s_{pre}$  where some property  $p$  holds and after the completion of the event  $e$  we obtain a new state  $s_{post}$  in which the property does not hold any more. It should be noted that events have an internal structure. Detailed discussion can be found in Vendler (1957). To give an example, the *opening of a window* has a pre-state, *i.e.* the closed window, and a post-state, *i.e.* the open window. The event itself has an internal structure such that at the beginning of the event the window is closed and is getting

opened more and more such that the *opening of the window* is steadily changing.

When we talk or write, we usually locate events within some kind of range of linearly ordered time points, resulting in a time scale. In some respect this is true for biochemical events as well. A biochemical event, for example a chemical reaction, takes place within a certain time span. It has a pre-state where some substrates exist and a post-state where some products exist. Nonetheless within my approach I can neglect talking about temporal locations of events and entities. First, there is a biological reason for avoiding time points. When we talk about a biochemical event like *transcription* or *translation* we abstract from what *really* happens in a cell. In fact not only one single product, like one single protein, is expressed but thousands of proteins are expressed. Biologists refer to this as concentration of proteins in a specific compartment. I can thus neglect the specific time point or time span when some reaction takes place. As a consequence I will use an event variable as referential argument for the transcription, translation, and chemical reactions. This can be seen according to Kamp and Reyle (1993). I will not specify any temporal details of the events, not even relations like *before* and *about* between them. Similarly I will not use states to localise the entities temporally although each chemical compound has a *life-time* during which it is existent.

### The material that genes are made of

Before being able to define what a gene is I first discuss and introduce the conceptualisation of a DNA sequence. The concept of *DNA sequence* is fundamental for the definition of the concepts *gene* and *genome*. The components that a *DNA sequence* is made of are the deoxyribonucleotides. These are connected through covalent bonds. There exist four different types of deoxyribonucleotides. Each nucleotide has a specific base part, which makes it differ from the others.

**DNA sequence:** The DNA sequence is a chain of deoxyribonucleotides. A DNA sequence is made up of four types of nucleotides referred to as adenine, guanine, cytosine and thymine.<sup>2</sup> The continuous concatenation of these elementary building blocks – which are from a chemical point of view glued through covalent bonds – results in a sequence and is expressed formally through the binary operator  $\bullet$ . In contrast to the fusion operator  $\oplus$ , the  $\bullet$  operator ensures the neighbourhood of the two entities, while the  $\oplus$  does not reveal anything about the composition of the elements. Neighbourhood means that they are localised next to each other through binding. Note that the binding details, which affect chemical properties and functional groups, are added in later sections. The  $\oplus$  operator ensures that in (13.5) the DNA sequence consists of nucleotides only. The last negated conjunct stating that  $x_k$  is not bound to  $x_1$  (*i.e.*  $\neg(\bullet(x_k, x_1))$ ) ensures that the DNA sequence

---

<sup>2</sup>There are differences with respect to the shape and the components of a nucleotide when being bound to one other nucleotide, or to two other nucleotides or to a completely different chemical compound. In each of these cases a different part (*i.e.* functional group) participates in the binding. A similarity relation defines the equivalence class of nucleotides by specifying necessary constraints. More on related issues is introduced in the following sections.

is not arranged in a circular fashion but is arranged having a beginning and an end.

$$(13.5) \quad \forall X [dnaSequence(X) \rightarrow \\ \exists_{(i=1 \dots k)}^{x_i} (\bigwedge_{(i=1 \dots k)} Deoxyribonucleotide(x_i)) \wedge \\ (X = x_1 \oplus \dots \oplus x_k) \wedge (\bigwedge_{(j=1 \dots (k-1))} \bullet(x_j, x_{j+1})) \wedge \neg(\bullet(x_k, x_1)))]$$

It should be noted that the above axiom allows for loops in DNA sequences. Although this does not occur in nature, this is ignored for the moment. In addition, further axioms that describe the structure of a nucleotide in more detail avoid this problem. In section 13.4 I define a nucleotide with two possibilities for binding to other nucleotides.

**Deoxyribonucleotide:** A deoxyribonucleotide is also known as a nucleotide and it constitutes the main building block of DNA sequences. Each deoxyribonucleotide has one of the four bases adenosine, cytosine, thymidine or guanosine. Each of these four deoxyribonucleotides reflects the base in its name: deoxyadenylate (see 13.6, or 13.8), deoxyguanylate (see 13.6, or 13.9), deoxycytidylate (see 13.6, or 13.10), or thymidylate (see 13.6, or 13.11).

$$(13.6) \quad \forall x [Deoxyribonucleotide(x) \leftrightarrow \\ (Deoxyadenylate(x) \vee Deoxyguanylate(x) \vee \\ Deoxycytidylate(x) \vee Thymidylate(x))]$$

To ensure that a deoxyadenylate is different from all other deoxyribonucleotides one can express this by means of an axiom like (13.7).

$$(13.7) \quad \forall x [Deoxyadenylate(x) \rightarrow \\ (\neg Deoxyguanylate(x) \wedge \neg Deoxycytidylate(x) \wedge \neg Thymidylate(x))]$$

As for the other deoxyribonucleotides the same can be expressed along the lines of (13.7).

**Nucleotides:** The four nucleotides already introduced each have the respective base as proper part. These are adenine, guanine, cytidine and thymine.<sup>3</sup> This is expressed in axioms (13.8), (13.9), (13.10), and (13.11).

$$(13.8) \quad \forall x \exists y [Deoxyadenylate(x) \rightarrow (Adenine(y) \wedge \sqsubset (y, x))]$$

$$(13.9) \quad \forall x \exists y [Deoxyguanylate(x) \rightarrow (Guanine(y) \wedge \sqsubset (y, x))]$$

$$(13.10) \quad \forall x \exists y [Deoxycytidylate(x) \rightarrow (Cytidine(y) \wedge \sqsubset (y, x))]$$

$$(13.11) \quad \forall x \exists y [Thymidylate(x) \rightarrow (Thymine(y) \wedge \sqsubset (y, x))]$$

---

<sup>3</sup>Of course, nucleotides comprise more chemical compound parts as well. But these are too specific at this general level of formalisation. These components are introduced formally and discussed in later sections.

## The genome, the gene and its parts

**Genome:** The genome consists of the complete genetic information of an organism. This means the total set of all genes of an organism (see Lackie and Dow (2000)). This is formalised in axiom (13.12). The part-of relation between the genes  $z_i$  and the genome  $x$  is given through the fusion of the genes constituting the genome. Note that this allows for the unary predicate *Gene* with a referential argument  $z$  only although it is related through the genome to an organism, which accounts for the organism dependence of genes.

$$(13.12) \quad \forall x, o[(Genome(x, o) \wedge Organism(o) \wedge \sqsubset (x, o)) \rightarrow \\ \exists_{(i=1 \dots k)}^{z_i} (Gene(z_i) \wedge x = (z_1 \oplus \dots \oplus z_k))]$$

**Gene:** Genes are made up of DNA sequence, where the sequence need not be continuous. This means that a gene can be scattered over a DNA strand with irrelevant parts in between. Thus I define a gene as an ordered set of nucleotide sequences. It can be divided into two functional parts: a coding sequence, and a non-coding sequence; both being necessary parts. Genes are frequently referred to as the basic unit for hereditary information and thus defined through the product they code for.

I formalise the relation between *Gene*, *CodingSequence* and *NonCodingSequence* in axiom (13.13). The predicate *Gene*( $x$ ) means that  $x$  is a gene in an organism, as a gene is always related to an organism which is ensured by (13.12)<sup>4</sup>. The predicate *CodingSequence*( $c, x$ ) stands for the coding sequence  $c$  of gene  $x$ . The predicate *NonCodingSequence*( $n, x$ ) analogously stands for the non-coding sequence  $n$  of the gene  $x$ . In axiom (13.13) it is ensured that the gene  $x$  consists only of the coding sequence  $c$  and the non-coding sequence  $n$  through the  $\oplus$  operator.

$$(13.13) \quad \forall x, \exists c, n[Gene(x) \rightarrow \\ (CodingSequence(c, x) \wedge NonCodingSequence(n, x) \wedge \\ (x = c \oplus n))]$$

Axiom (13.14) ensures that coding sequences and non-coding sequences are disjoint, which means that they do not have any parts in common.

$$(13.14) \quad \forall x, y, g[(CodingSequence(x, g) \wedge \\ NonCodingSequence(y, g) \wedge Gene(g)) \rightarrow \\ \neg \exists z(\sqsubset (z, x) \wedge \sqsubset (z, y))]$$

**Coding sequence:** The coding sequence is a necessary part of a gene, which codes for the resulting peptide or the resulting stable RNA. It is formed by a set of subsequences of nucleotides, *i.e.* the coding sequence of a gene need not necessarily

---

<sup>4</sup>At this point I do not differentiate between genes in their *native* organism or in *non-native* organisms. From this formalisation here both are possible.

be a continuous stretch of a DNA sequence. In eucaryotes these sub-sequences are termed **exons**. The coding sequence can then be seen as being subdivided into several exons interspersed by non-coding sequences called introns. I do not formalise the concept of introns in this work. However, the formalisation given should allow for this extension. The concept of coding sequence is formalised in axiom (13.15). A coding sequence is an ordered set (or list) of DNA sequences referred to through the variables  $x_1, \dots, x_k$ . Each  $x_i$  is a component of a gene. The  $\oplus$  operator is used to formalise that the coding sequence is formed by the set of sequences  $x_1, \dots, x_k$ , and nothing else. A sequence  $x_i$  can only belong to either the coding sequence of a gene  $y$  or to the non-coding sequence of the same gene. I would like to emphasise the difference between the  $\bullet$  operator as shown in (13.5) and the  $\prec$  operator here in (13.15). In (13.5)  $\bullet$  denotes a type of chemical binding, whereas here in (13.15)  $\prec$  is an operator that does not denote chemical binding. It simply imposes an order on the DNA sequence parts. This is necessary for the transcription process.

$$(13.15) \quad \forall g, x [(CodingSequence(x, g) \wedge Gene(g)) \rightarrow \\ (\exists_{(i=1, \dots, k)}^{x_i} (\bigwedge_{(i=1, \dots, k)} dnaSequence(x_i) \wedge (x = x_1 \oplus \dots \oplus x_k) \wedge \\ \sqsubset (g, x_i) \wedge (\bigwedge_{(i=1, \dots, (k-1))} \prec (x_i, x_{i+1}))))]$$

**Non-coding sequence:** A non-coding sequence is a necessary part of each gene. It contains at least one regulatory element steering the transcription process as shown in (13.16). The non-coding sequence  $x$  of a gene  $y$  is thus referred to as *RegulatoryNCDS*( $x, y$ ).

$$(13.16) \quad \forall x \exists y [Gene(x) \rightarrow RegulatoryNCDS(y, x)]$$

The following list shows the various types of regulatory units.

- In eucaryotes the non-coding sequence consists of a binding site, a termination sequence and introns. For procaryotes it consists of a binding site and a termination sequence only. For both procaryotes and eucaryotes this can be expressed through (13.17) since no introns are introduced in the formalisation.

$$(13.17) \quad \forall x, y [(RegulatoryNCDS(x, y) \wedge Gene(y)) \rightarrow \\ \exists b, t (BindingSite(b, y) \wedge TerminationSequence(t, y) \wedge \\ \sqsubset (b, x) \wedge \sqsubset (t, x))]$$

Of course, a regulatory non-coding sequence is a special type of non-coding sequence. This is expressed by (13.18).

$$(13.18) \quad \forall x, g [RegulatoryNCDS(x, g) \rightarrow NonCodingSequence(x, g)]$$

- The binding site is a part of a gene. I specify this through the binary predicate  $BindingSite(b, x)$ . It is thus always the binding site of a gene.

$$(13.19) \quad \forall b \exists x [BindingSite(b, x) \rightarrow Gene(x)]$$

The binding site itself can have the following components:

- \* The promoter region is an obligatory site, where the transcription starts<sup>5</sup>. It is always the promoter of a gene. This is expressed through the binary predicate  $promoter(p, x)$ , where  $x$  is the gene.

$$(13.20) \quad \forall b \exists p, x [(BindingSite(b, x) \wedge Gene(x)) \rightarrow (Promoter(p, x))]$$

- \* The enhancer is an optional site to which transcriptional activators bind, it is also referred as *upstream activating sequence*. The enhancer – like the promoter – is a relational noun, which means it is always the enhancer site of a specific gene. This is expressed through the binary predicate  $enhancer(e, x)$  with  $x$  being the gene.

$$(13.21) \quad \forall e \exists x, b [Enhancer(e, x) \rightarrow (BindingSite(bs, x) \wedge Gene(x))]$$

- \* The silencer is an optional site to which transcriptional repressors bind, it is also called the *upstream repressing sequence*. It is represented as a relational predicate, too.

$$(13.22) \quad \forall s \exists x, b [Silencer(s, x) \rightarrow (BindingSite(b) \wedge Gene(x))]$$

- Termination sequence is an obligatory site. It delimits where the transcription ends.
- Introns are non-coding sequences which are found in between the (coding) exons of eucaryotic genes. They can contain binding sites. But, as already mentioned non-coding sequence is defined without introns.

The promoter and termination sequence are obligatory. The other sites are optional. I'm well aware of the fact that there are some restrictions concerning the ordering of these sites, *e.g.* a termination sequence is downstream at the end of a gene. However, they are not included in this formalisation of gene expression.

## Gene transcripts, peptides and proteins

**RNA sequence:** The RNA sequence is a chain of ribonucleotides connected through phospho-diester bonds. Just as for the DNA sequences, it is formalised here

---

<sup>5</sup>The RNA polymerase binds to the promoter and starts the transcription process from this position

through a concatenation operation without going into its chemical details. The axiom is analogous to that of *dnaSequence* in (13.5):

$$(13.23) \quad \forall X[rnaSequence(X) \rightarrow \\ \exists_{(i=1\dots k)}^{x_i} (\bigwedge_{(i=1\dots k)} Ribonucleotide(x_i) \wedge \\ (X = x_1 \oplus \dots \oplus x_k) \wedge (\bigwedge_{(j=1\dots(k-1))} \bullet(x_j, x_{j+1})) \wedge \neg(\bullet(x_k, x_1)))]$$

**Gene transcript:** A gene transcript is the result of the transcription process, which takes a gene as input and produces either a stable RNA or a mRNA<sup>6</sup>. In general different types of stable RNA are known, *i.e.* rRNA, tRNA and a list of other small RNAs, like siRNA, snRNA, miRNA. I do not specify them formally in (13.24).

$$(13.24) \quad \forall x[GeneTranscript(x) \rightarrow \\ (mRNA(x) \vee StableRNA(x))]$$

*mRNA* and *stable RNA* are two disjoint concepts which can be expressed along the lines of (13.7).

A gene transcript and thus both a mRNA and a stable RNA is always an RNA sequence.

$$\forall x[GeneTranscript(x) \rightarrow rnaSequence(x)]$$

**Peptide sequence:** A peptide sequence is a sequence or a chain of amino acids. The sequence is formalised along the lines of (13.5) and (13.23). Here, a peptide sequence is introduced independently of an organism.

$$(13.25) \quad \forall X[PeptideSequence(X) \rightarrow \\ \exists_{(i=1\dots k)}^{x_i} (\bigwedge_{(i=1\dots k)} AminoAcid(x_i) \wedge \\ (X = x_1 \oplus \dots \oplus x_k) \wedge (\bigwedge_{(j=1\dots(k-1))} \bullet(x_j, x_{j+1})) \wedge \neg(\bullet(x_k, x_1)))]$$

**Proteins:** Proteins are chains of amino acids. They are often also called peptides. Proteins are organism dependent, which means that a specific peptide sequence with the corresponding organism describes the protein. Proteins have a native three dimensional folding structure and usually one or more specific functions. In the following I neglect the three dimensional structure and focus on the sequence and the organism specificity. To define that a protein has a certain function – for example, that it acts as transcriptional regulator – can be introduced as a special type of protein.

$$(13.26) \quad \forall x, o[(Protein(x) \rightarrow \\ PeptideSequence(x) \wedge Organism(o) \wedge \sqsubset(x, o))]$$

---

<sup>6</sup>If the organism is a eucaryote, the mRNA is produced from pre-mRNA through RNA-processing. I subsume this processing step within the transcription process.

### 13.2.2 Relations between concepts

**Transcription:** The transcription process is formalised as an event  $e^7$ . A transcription is treated as a special type of reaction, where a reaction has substrates – here, the substrate is a gene – and products – here, the product is the corresponding gene transcript –<sup>8</sup>.

$$(13.27) \quad \forall e, x, y [transcription(e, x, y) \rightarrow \\ (reaction(e, x, y) \wedge Product(y) \wedge Substrate(x) \wedge \\ GeneTranscript(y) \wedge Gene(x))]$$

**Translation:** On a very general level the translation process can be seen as a reaction with products and substrates as well. This is expressed by (13.28).

$$(13.28) \quad \forall e, x, y [translation(e, x, y) \rightarrow \\ (reaction(e, x, y) \wedge Product(y) \wedge Substrate(x) \wedge \\ mRNA(x) \wedge Peptide(y))]$$

## 13.3 Chemical elements and chemical bonds

In the previous section I introduced a conceptualisation of gene expression where the most relevant concepts have been introduced. In addition, some part-of relations between the concepts have been specified as well. Nonetheless, this conceptualisation describes gene expression at a coarse level of granularity. For example, it does not specify whether a nucleotide is built of a sugar part, or has a phosphate group. Depending on the application of such an ontology, one might wish to have different levels of specificity, *i.e.* to be able to switch from this rough biological model to a more chemistry-related and thus more detailed model. To show how such a more chemistry-related model can be built and where the main challenges are, I introduce in this section a conceptualisation of chemical elements (especially as part of compounds) and chemical bonds to form chemical compounds.

### 13.3.1 Chemical elements in chemical compounds

Chemical elements (atoms) constitute the building blocks of molecules. I introduce a set of atoms which are necessary for describing bindings within organic molecules. I do not specify additional properties of the chemical elements like valence, molecular weight etc., since I do not specify the complete structure of chemical compounds. Chemical

---

<sup>7</sup>Events were previously introduced in the subsection *Event variables and event structures*

<sup>8</sup>To define the concept of chemical reactions in detail would go far beyond the scope of this work and is open to definition later. It is enough for the moment to specify it as having substrates and products and to note that these are roles that the entities play. And, the roles can change, as a product of one reaction can be the substrate of another reaction.

elements are only introduced when they are necessary for describing chemical bindings between molecules. Valence information can be added or specified later if needed.

The biologically relevant atoms (and the abbreviations I use) are: carbon (C), hydrogen (H), oxygen (O), nitrogen (N), sulphur (S), chlorine (Cl), bromine (Br), iodine (I), lithium (Li), magnesium (Mg), boron (B), and phosphorus (P).

$$(13.29) \quad \forall x[(BioChemicalElement(x) \leftrightarrow \\ C(x) \vee H(x) \vee O(x) \vee N(x) \vee S(x) \vee Cl(x) \vee \\ Br(x) \vee I(x) \vee Li(x) \vee Mg(x) \vee B(x) \vee P(x))]$$

Later on in the following formalisation I use in addition the ternary predicate  $\alpha(x, y, z)$  to refer to a certain chemical element  $x$  within a molecule  $z$ .  $\alpha$  is the name of the chemical element  $x$ . In the case of oxygen I thus write  $O(x, y, z)$ .  $y$  denotes the number of  $x$  within a molecule  $z$ , where the number is given by IUPAC recommendation. The connectedness of  $x$  to the rest of the molecule  $z$  is expressed through the binding relation  $bond(x, z)$ , which is discussed and introduced thoroughly in the following section 13.3.2.<sup>9</sup>

A first example for the use of  $\alpha(x, y, z)$  can be found in (13.44). In this example I explicitly introduce a nitrogen element, which is used later to denote the relevant binding partner within molecules.

### 13.3.2 Binding relations

A binding event brings about a change in the linkage between two chemical entities  $x$  and  $y$ . After the completion of a binding event,  $x$  and  $y$  are in the state of being bound to each other. This means that a complex formation has taken place. To explain the mechanisms of *bindings* and to specify the types of *bindings* it is necessary to go more into the detail of what *bind* means and to talk about the role of electrons in this process. In chemistry and biochemistry two complementary types of bindings are distinguished. These are *covalent* and *noncovalent binding*. Both refer to binding between single atoms as well as binding between molecules where the binding might take place between several atoms. In a first step I inspect binding on the level between atoms. Subsequently I introduce a more general concept of binding which allows binding between any kind of chemical compounds.

#### Chemical bonds between atoms

**Covalent binding** is the strongest type of binding between two atoms and it is formed when two atoms are able to share at least two electrons, where each participating atom

---

<sup>9</sup>Although there exists an extended nomenclature on the unique numbering for chemical compounds (see IUPAC (1993) or IUPAC (1979) for details) I only use a very reduced version. More on parsing of chemical compounds is described in Gerstenberger (2001).

donates at least one electron<sup>10</sup>. In the case of two atoms each donates one electron and the type of the resulting bond is called a *single bond*. In cases where each of two atoms donate two electrons, the bond is called *double bond*. In the case of triple bonds, three electron pairs are shared. *Ring conformations* behave slightly differently. Within ring structures it appears that a steady change between single and double bonds takes place. This means that more than two atoms share some electron such that each of the two cannot be assigned to one single atom. Or, to put it differently, the electrons “jump” between pairs of atoms. The figure 13.2 shows a schematic representation of a covalent bond between two atoms.



Figure 13.2: Schematic representation of a covalent bond. *A* and *B* represent two atoms. The  $\bullet$  represent the electrons. On the right hand side of the arrow the two electrons are shared.

The different types of covalent bonds have different characteristics like restrictions on 3-dimensional conformation, strength of binding (depending on which atoms participate in the binding) and the like. It should be noted that this formalisation that I present deals with the binding between participating atoms and not with types of bonds or even chemical structure like the 3-dimensional conformation of the compounds.

$$(13.30) \quad \forall x, y [covalent\_atom\_bond(x, y) \leftrightarrow \\ \exists e_1, e_2 (Atom(x) \wedge Atom(y) \wedge Electron(e_1) \wedge Electron(e_2) \wedge \\ \sqsubset (e_1, x) \wedge \sqsubset (e_1, y) \wedge \\ \sqsubset (e_2, x) \wedge \sqsubset (e_2, y) \wedge \\ Overlap(x, y))] ]$$

The formalisation given in (13.30) does not account for the fact that an electron first belongs to only one atom and then to two. By introducing events and states one might specify in addition the states that hold before the binding and the states that hold after the binding. I neglect this within the formalisation, since it is not needed any more in the following work.

It should be mentioned that in axiom (13.30) the overlap of *x* and *y* can be inferred from the sharing of electrons. Nonetheless I leave it to the predicate *Overlap(x, y)* to make this state explicit. In addition, the equivalence relation in (13.30) ensures that whenever two atoms share two electrons that they are covalently bound.

**Noncovalent bonds** between atoms can be described as the attraction resulting from the different charges of the participating atoms. Noncovalent bonds can either occur as ionic bonds, hydrogen bonds and van der Waals bonds. Only one of the three can hold for two noncovalently bound atoms. It should be noted that the three types of noncovalent bonds are complementary, which could easily be expressed as shown before.

<sup>10</sup>I model this sharing by stating that these electrons are proper parts  $\sqsubset$  of both atoms. This does not account for the fact that each belonged to one atom before. An alternative to the proper part would be to represent the sharing with a *share(x, y, {e<sub>i</sub>, ...})* predicate, where the first two arguments refer to the participants that share the set of entities given by the third argument.

However, this is ignored to keep the formalisation as short as possible.<sup>11</sup>

$$(13.31) \quad \forall x, y [noncovalent\_atom\_bond(x, y) \rightarrow \\ ((ion\_bond(x, y) \vee hydrogen\_bond(x, y) \vee v\_d\_W\_bond(x, y)) \wedge \\ Atom(x) \wedge Atom(y))]$$

Ionic bonds result from the attraction of atoms with opposite charges. The opposite charges result from a loss of one (or more) electron(s) from at least one of the participating atoms. An exchange of electrons between two noncovalently bound atoms usually happens because the electron is in a more stable state within the new atom after the exchange. Salt crystals are a good example. The NaCl molecules are arranged in a 3-dimensional lattice structure. Concerning the noncovalent binding, the molecule NaCl results from the oppositely charged  $Na^+$  and  $Cl^-$ . The binding responsible for the NaCl conformation is called ionic binding.

Hydrogen bonds result from noncovalent bonds in which hydrogen is one participant. Having two different types of binding partners – like in water – the two binding partners attract the shared electrons to a different degree. A polar structure is the consequence. For  $H_2O$  this means that the hydrogen ends are positively charged and the oxygen end is negatively charged. Whenever the positively charged hydrogen and the negatively charged oxygen come close to each other they are attracted. Hydrogen bonds are weaker than ionic bonds.

The last type of noncovalent bonds are van der Waals bonds. This is the weakest type of binding and usually strong enough only when many of them are formed simultaneously. The main reason for van der Waals attractions is that at very short distances any two atoms show weak bonding interactions<sup>12</sup>.

The order for the types of bindings introduced according to their strength is as follows:

Covalent bond > Ionic bond > Hydrogen bond > Van der Waals bond

Table 13.1: Chemical bindings hierarchy.

To be able to refer to atomic binding as a superconcept of covalent and noncovalent binding, I introduce the predicate *atom\_bond*.

$$(13.32) \quad \forall x, y [atom\_bond(x, y) \leftrightarrow \\ (noncovalent\_atom\_bond(x, y) \vee covalent\_atom\_bond(x, y))]$$

<sup>11</sup>Similarly as in axiom (??) it is possible to introduce an axiom for not noncovalently bound atoms.

<sup>12</sup>Although if the two atoms are too close together rejection is the consequence.

It should be noted that covalent binding and noncovalent binding are complementary. This means either two atoms are covalent or noncovalent bound. They cannot be both, which is shown in (13.33) and (13.34).

$$(13.33) \quad \forall x, y [noncovalent\_atom\_bond(x, y) \rightarrow \neg covalent\_atom\_bond(x, y)]$$

$$(13.34) \quad \forall x, y [covalent\_atom\_bond(x, y) \rightarrow \neg noncovalent\_atom\_bond(x, y)]$$

### Chemical bonds between molecules

When biologists talk about binding they do not always mean binding between atoms but also refer to binding between molecules. Although binding between molecules can in principle be decomposed into binding(s) between atoms, this not always possible<sup>13</sup> or desirable<sup>14</sup>. Often it is simply not necessary to know the exact positions within the proteins where bindings take place. But it is of interest which proteins interact. The binding relation remains underspecified.

With respect to binding relations I treat atoms as a special case of molecules. And, having *atom\_bond* and *bond* as part of my conceptualisation, this gives me the possibility to switch from *bond* to *atom\_bond* whenever necessary or desired. It is important that the formalisation offers the possibility to be as specific as necessary but not more than desired. Later extensions specifying the details about binding on an atomic level are possible too.

As we have seen, in the case of atomic bindings (bio-)chemists distinguish between covalent and noncovalent bindings. The same holds for the case of underspecified bindings. We can distinguish between covalent and noncovalent bindings on a less specific level, too. This means that I introduce the two predicates *covalent\_bond* and *noncovalent\_bond* (see (13.35) and (13.36)) and the superconcept *bond* (see (13.37)). Both refer to bindings between molecules and mean that we have *covalent\_atom\_bond* or *noncovalent\_atom\_bond* between atoms which are either the participating molecules or parts of these.

---

<sup>13</sup>For a protein complex with several proteins, it is mostly not possible to specify the bindings between atoms. Developments in biology are not yet far enough advanced such that complete information would be available.

<sup>14</sup>Regarding large-scale protein interaction experiments like mass spectrometry in combination with TAP (Tandem Affinity Purification) – as reported in Gavin *et al.* (2002) – data is produced which gives evidence about numerous proteins within an organism and evidence for possible protein interactions. The results do not give certainty with respect to every interaction taking place in an organism let alone some details about bindings between these proteins.

$$(13.35) \quad \forall x, y [covalent\_bond(x, y) \leftrightarrow (\exists x_1, y_1 (\sqsubset (x_1, x) \wedge \sqsubset (y_1, y) \wedge covalent\_atom\_bond(x_1, y_1)))]$$

$$(13.36) \quad \forall x, y [noncovalent\_bond(x, y) \leftrightarrow (\exists x_1, y_1 (\sqsubset (x_1, x) \wedge \sqsubset (y_1, y) \wedge noncovalent\_atom\_bond(x_1, y_1)))]$$

$$(13.37) \quad \forall x, y [bond(x, y) \leftrightarrow (covalent\_bond(x, y) \vee noncovalent\_bond(x, y))]$$

### Building Molecules

Given the binding relation we are now able to build molecules from atoms. Thus the simplest case for a molecule is when two atoms are bound. Then the molecule consists of these two atoms only.

$$(13.38) \quad \forall x, y, z [(Atom(x) \wedge Atom(y) \wedge bond(x, y)) \rightarrow (Molecule(z) \wedge (z = x \oplus y))]$$

Given an atom that is bound to a molecule the result is a molecule again. This is expressed by (13.39).

$$(13.39) \quad \forall x, y, z [(Molecule(x) \wedge Atom(y) \wedge bond(x, y)) \rightarrow (Molecule(z) \wedge (z = x \oplus y))]$$

### Chemical bonds summary

Table 13.2 gives an overview of the types of binding introduced. The upper part has the three underspecified types of binding *bond*, *covalent\_bond* and *noncovalent\_bond*. The lower part of the table shows the same bindings on atomic level. The arrows pointing to the middle can be read as *is\_a* relations. The two outer arrows represent the specific entailment relations between (non) covalent bonds and (non) covalent atomic bonds according to axioms (13.36) and (13.35).

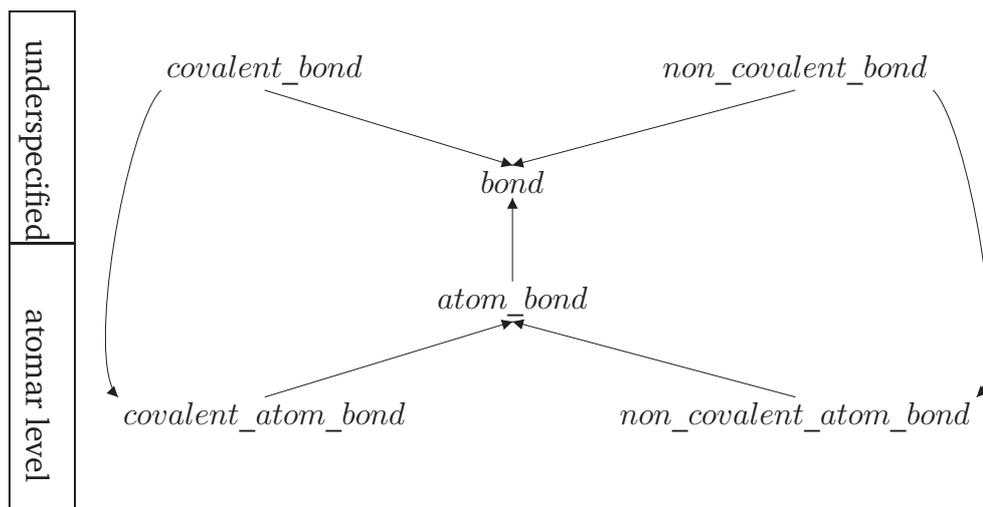


Table 13.2: **Summary of chemical bindings:** The table shows the different types of chemical bindings that have been introduced so far. The relations between them are presented as arrows representing *is\_a* relations with exception of the two outer arrows, which have to be read as entailment relations.

Given the definition of covalent and noncovalent bindings we are now able to use these relations to build complexes of atoms, like molecules, or complexes of special types of molecules, like sugars, amino acids or polynucleotides.

### 13.3.3 Functional groups

In biochemistry millions of organic compounds are known<sup>15</sup>. Organic compounds are defined in Frunder *et al.* (1995) as molecules containing carbons<sup>16</sup>. A carbon atom has four electrons and four vacancies in its outermost shell and so it can carry out four covalent bonds. It is able to bind to other carbon atoms and thus build carbon skeletons. A carbon skeleton is a kind of backbone where other chemical elements can be attached. For example hydrogen atoms are frequent binding partners of carbon atoms. But since they have very little reactive potential they are usually not referred to explicitly. It is thus a kind of default assumption that the not-specified binding partners are hydrogen atoms. A carbon backbone with the attached hydrogen atoms is called a carbohydrate and it usually forms the less reactive part of organic compounds. In addition to this carbohydrate part of organic compounds, more reactive parts are responsible for the reactive characteristics of the chemical compounds. Usually the *functional group(s)* bear this reactive potential. Functional groups are made up of a group of atoms and they are bound to the carbohydrate backbone. The chemical properties of a chemical compound or a molecule usually depend on the functional groups the compound has. The

<sup>15</sup>According to Frunder *et al.* (1995) there are more than  $7 \times 10^6$  known organic compounds.

<sup>16</sup>Frunder *et al.* (1995) explicitly mentions three types of exceptions, *i.e.* carbides, carbonates and carbon oxides.

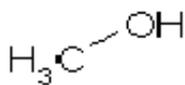


Figure 13.3: Methanol

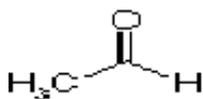


Figure 13.4: Ethanal

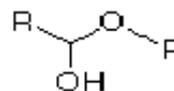


Figure 13.5: Hemiactal

functional groups thus determine the ability of the molecule to participate in certain reactions (see Peter *et al.* (1999) page 2).

One and the same *functional group* usually has – more or less - the same chemical property across different compounds. One possibility is thus to group chemical compounds according to the functional groups they carry. A prominent example are the *hydroxyl*-groups consisting of an oxygen and a hydrogen atom (-OH). The compounds carrying such a *hydroxyl*-group are usually referred as alcohols (see Fig. 13.3 for Methanol). A similar example is aldehydes with an oxygen and a hydrogen atom attached separately to the same carbon atom (see Fig. 13.4 for ethanal). An example for typical reactions of alcohols and aldehydes is that these react as substrates together to form a hemiacetal (see Fig. 13.5 for hemiacetal): alcohol + aldehyde  $\rightarrow$  hemiacetal<sup>17</sup>. It is important to note that once these functional groups have carried out this specific reaction they have lost their reactive potential. Another characteristic is that it is possible to reverse such reactions so that the substrates can be retrieved with their former reactive potential. One and the same compound can have various functional groups and thus belong to multiple classes<sup>18</sup>. Details about the chemical structure of the various functional groups can be found in IUPAC (1993) or IUPAC (1979). But, all in all there is no exhaustive list of functional groups. Some common ones are *alkane*, *alkene*, *alkyne*, *alcohol*, *aldehyde*, *ketone*, *carboxylic acid*, *ether*, *ester*, *amine*, *amide* etc.

Before going into the details of formalisation, I first give a short summary of the important facts so far and turn the discussion to a related question:

- Each functional group has a discrete chemical structure. It does not occur freely. It is part of a larger compound, *i.e.* it exists bound to some chemical compound.
- Functional groups have discrete states of reactive potential. This means they can carry out chemical reactions together with other functional groups and thus are bound to other chemical compounds. By undergoing such reactions they lose their reactive potential. But the states of reactive potential are not the same for all functional groups. Some functional groups can carry out several reactions, others only one.
- These chemical reactions are in principle reversible and thus the reactive potential is recovered.

<sup>17</sup>An example for such a reaction can be found when sugars change their structure from a carbon chain to a carbon ring in aqueous solutions.

<sup>18</sup>Another classification scheme is used for enzymes. They are not grouped according to the chemical reactions they may participate in, but they are grouped according to the chemical reaction they catalyse.

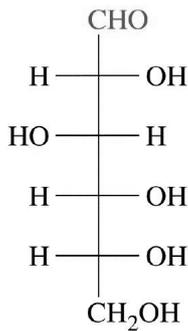


Figure 13.6: Aldose

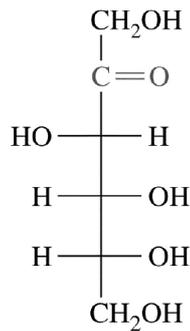


Figure 13.7: Ketose

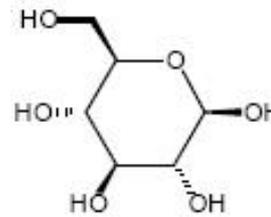


Figure 13.8: Monosaccharide

### 13.3.4 Sugar $\neq$ sugar

A related problem is that several chemical compounds do not exist as stable compounds, but change their conformation due to various reasons, for example due to contextual circumstances. One example is sugar. Both aldoses (see 13.6 for an example) and ketoses (see 13.7) can either exist as a chain or as a ring (see 13.8) within aqueous solutions. Aldoses for example have an aldehyde functional group and a hydroxyl group, each with its own reactive potential when they exist as a chain. In aqueous solutions these two groups react and the whole compound forms a cyclic hemiacetal.

Another example are ring conformations where the alternating single and double bonds are not permanently at the same place. Instead the electrons – responsible for the double bonds – move around the ring structure from carbon to carbon atom and thus between two carbon atoms the binding alternates from single to double bond.

### 13.3.5 Representing functional groups

In the following I use the binary predicate  $FunctionalGroup(x, y)$ , where  $x$  is the referential argument of the functional group and  $y$  is the chemical compound that it is part of. However, moving from a general level like the introduced nucleotides to this level of granularity brings about some inconsistencies. On the one hand I have pointed out two important things about the change in conformation in chemical compounds, *i.e.* chemical reactions (see section 13.3.3 and section 13.3.4). First, functional groups have a reactive potential that drives the process of building new chemical compounds. Second, these key players in chemical reactions are subject to change, which means they react with other functional groups being completely absorbed in new functional groups. On the other hand within section 13.2 I have treated a nucleotide – be it a deoxyribonucleotide or a ribonucleotide – as a stable chemical compound occurring freely, bound to one other nucleotide (at the end of a DNA or RNA chain), or bound to two other nucleotides within a DNA or RNA chain.

One consequence of this is that functional groups and thus chemical compounds are not stable over time. A change of conformation happens when being bound to other compounds. However, biologists identify them still as the same chemical compounds

within new compounds. This is shown within the following sections for nucleotides where, for example, a sugar is still identifiable within a nucleotide. Biologists even talk about the sugar-phosphate backbone (see Figure 13.9) of a nucleotide sequence. There exist several possibilities to find a remedy. I want to briefly introduce three of them, using a non-monotonic logical framework, modelling the chemical processes (*i.e.* chemical reactions) with events and states completely, or introducing discrete entities for the various types of the chemical compounds used in a specific application.

The term *non-monotonic logic* is used to cover a family of formal frameworks to represent defeasible inference. Usually this refers to inference of everyday life with tentative conclusions with the possibility of retracting them given further information. Non-monotonic refers thus to the fact that the set of conclusions does not increase steadily. In fact, it can shrink in contrast to classical first-order

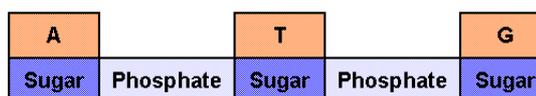


Figure 13.9: Schematic representation of a nucleotide sequence with a sugar-phosphate backbone (blue). The bases are shown as orange coloured boxes.

logic as I have used it so far. The standard and well-known example of non-monotonic reasoning from the field of artificial intelligence (AI) is the *Tweety* example. Generally, one would like to have a rule like *birds fly*. Given that *Tweety is a bird* is true, one would like to infer from that that *Tweety flies*. Unfortunately, if *Tweety is a penguin* is true, then we would like to ensure that *Tweety does not fly*. The *being-a-penguin* rule should be able to override the *birds-fly* rule. This can be dealt with in *non-monotonic logic* where default rules can be expressed and are applied provided no conflicting (or more specific) information is assumed to be the case. An application of default reasoning is shown in Lascarides and Asher (1993). Applying this type of logic to our formalism would allow us to treat the functional groups more correctly and thus to state properties for an unbound chemical compound (*i.e.* the default rule). If the compound becomes bound (*i.e.* further and more specific information on the context is provided) the system would allow the retraction of the information that a specific functional group is still part of chemical compound. A similar and less complex example is the default assumption of biochemists that carbon atoms have their free valences occupied with hydrogen atoms. To express this formally we use the  $>$  operator for default rules as shown in (13.40).

$$(13.40) \quad \forall x[Carbon(x) > \exists^4 z(Hydrogen(z) \wedge bound(z, x))]$$

This approach is suggested along the lines of Reyle (2005), which shows how to use non-monotonic default reasoning for deriving chemical structure from the names of chemical compounds. If at some later point we specify, for example a binding between two carbon atoms, each of them loses one free valence and thus only 3 hydrogen atoms can be bound by each of them. This is expressed with (13.41).

$$(13.41) \quad \forall x[(Carbon(x) \wedge \exists y bond(x, y) \supset \\ \exists^{\neq 3} z(Hydrogen(z) \wedge bound(z, x)))]$$

To catch the cases where a carbon atom has already lost two or three valences through binding to other atoms, similar axioms along the lines of (13.41) can be introduced.

The modelling of the chemical reactions would make it necessary to use event variables and state variables as described in Kamp and Reyle (1993)<sup>19</sup>. With the possibility to refer to discrete states of a chemical compound it allows us to specify properties of the compound that hold for a specific state and thus for a specific temporal interval. The chemical reactions that can be represented as events carry a compound from one state to another by specifying the changes that apply to the compounds. This approach meets the specified requirements very well. However, the main drawback of this approach is that the modelling effort is huge due to the amount of potential participants in a chemical compound. Each of the atoms within a chemical compound has to be specified and traced within a chemical reaction. These enormous efforts would lead to a shift of the priorities with respect to the core of the ontology. This is highly dependent on the application that is intended for the designed ontology.

>From a logical point of view, the most clumsy way of avoiding this issue is to introduce a concept for each conformation that a chemical compound can have. This means that for a nucleotide one might introduce an “unbound” version and a “bound” version, *i.e.* *nucleotide* and *nucleotide\_bond*. Of course, this multiplies the amount of chemical entities that have to be introduced. In addition it is not straightforward how to introduce the relation between these concepts, since one might want to be able to state some general properties about the nucleotides. Furthermore, since there are several possibilities of bindings for each chemical compounds, an enormous amount of entities would need to be invented. Nonetheless, if only few entities and very few types of interacting compounds are considered in the formalisation this might be the easiest way out of this problem. A related solution is presented in Reyle (2005). According to this approach one can introduce a concept like *Nucleotide* that represents the class of all nucleotides. This class comprises all nucleotides that fulfil a set of necessary conditions. At this level it is not possible to distinguish between a nucleotide that is unbound, or bound to one or two other nucleotides. By avoiding the introduction of these classes, one can abstract over the different types of conformation that a molecule can have.

## 13.4 Building chains of nucleotides

Within the following paragraphs the composition of nucleotide chains, *i.e.* DNA- and RNA-strands, is formally developed. The basic building blocks are nucleotides, which

---

<sup>19</sup>Events have already been introduced previously in the subsection *Event variables and event structures*.

themselves are built of the purine and pyrimidine bases, a sugar part (*i.e.* *ribose* and *2-deoxyribose*) and a phosphate group. I start by defining the bases' compounds, which are followed by the sugar compounds. These two types of compounds are then composed to build nucleosides, which get attached to a phosphate group to form nucleotides. In the last step the DNA- and RNA-strands are composed of the corresponding nucleotides.

### 13.4.1 The bases purine and pyrimidine

A parent compound is the basic compound from which derivatives are generated through substitution of single atoms through other atoms or functional groups. For methanol (or ethanol) the parent compound is methane (or ethane), *i.e.* one hydrogen atom is replaced by the functional alcohol group *-OH*.

For building nucleotides the parent compounds purine and pyrimidine are pivotal. The derivatives that are of interest in this context are the two purine derivatives:

**Guanine** is also known as *2-amino-6-oxy purine* and shows from its name that it is a purine derivative. The chemical structure can be seen in Figure 13.16).

**Adenine** is also referred to as *6-amino purine*. From the name one can see that it is a derivative of purine. Its chemical structure is shown in Figure 13.15.

The derivatives that derive from pyrimidine, which is also known as *1,3-Diazine*, are:

**Cytosine** which is also known as *2-oxy-4-amino pyrimidine*. Its chemical structure is shown in Figure 13.11.

**Thymine** which is also referred to as *2,4-dioxy-5-methyl pyrimidine*. The chemical structure is shown in Figure 13.12.

**Uracil** which is also named *2,4-dioxy pyrimidine* showing that it is a derivative of pyrimidine. Its chemical structure is shown in Figure 13.13.

The taxonomic relationships for these compounds are formalised in (13.42) and in (13.43). Since there exist no other pyrimidines than the mentioned ones we can use the equivalence relation  $\leftrightarrow$  in (13.42). In contrast to that there exist purines other than those mentioned (*e.g.* xanthine). But since they are not needed in the following sections and are not as important as guanine and adenine in the field of biochemistry I disregard them. And for this reason I put an equivalence relation  $\leftrightarrow$  in (13.43) as well. Whenever needed, it is possible to add further purines to the list. This gives me the ability to generalise over purines if needed.

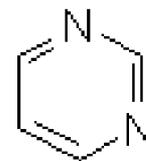


Figure 13.10: Pyrimidine

$$(13.42) \quad \forall x[(Cytosine(x) \vee Thymine(x) \vee Uracil(x)) \leftrightarrow PyrimidineDerivate(x)]$$

$$(13.43) \quad \forall x[(Guanine(x) \vee Adenine(x)) \leftrightarrow PurineDerivate(x)]$$

Purines and pyrimidines are both ring structures. While pyrimidines consist of one ring purines are fusions of two ring structures, where one is a pyrimidine ring and the second one is an imidazole ring attached to the first one. The pyrimidine ring has 6 carbon atoms with two replaced by nitrogen atoms. The first one is at position 1 (the N-1 atom) and the second is at position 3<sup>20</sup>. The first of the two nitrogen atoms is important for the later formalisation of nucleosides and can be found in (13.44). This N-1 position is the glycosydic binding site for sugar when building the nucleosides (see (13.68) and (13.69)).

$$(13.44) \quad \forall x[Pyrimidine(x) \rightarrow \exists yN(y, 1, x)]$$

As already mentioned, the purines are composed of a pyrimidine ring with an imidazole ring attached, consisting of 3 additional carbon atoms, two of which are replaced by nitrogen atoms at positions 7 and 9. Similar to the pyrimidines, the purines have a nitrogen atom which is important for carrying out the glycosidic binding, too. It is the nitrogen atom at position 9 (N-9 atom in (13.45)).

$$(13.45) \quad \forall x[Purine(x) \leftrightarrow \exists yN(y, 9, x)]$$

Pyrimidines and purines are also called nucleobases. The term nucleobase usually comprises all nitrogen-containing rings, which are part of nucleosides, nucleotides or nucleic acids. (13.46) shows this taxonomic relationship. I use the equivalence relation to be able to generalise over the concept. Other nucleobases could easily be added later.

$$(13.46) \quad \forall x[Purine(x) \vee Pyrimidine(x) \rightarrow Nucleobase(x)]$$

### 13.4.2 Glycosylation needs sugar

Sugars are the binding partner responsible for the glycosylation of the nucleobases. Sugar compounds contain a number of alcoholic OH-groups. These participate in several chemical reactions. The most important one is the formation of glycosidic linkages

---

<sup>20</sup>I do not go deeper into the structural details of chemical compounds than is necessary for defining the relevant binding relations. More on this can be found in Gerstenberger (2001).

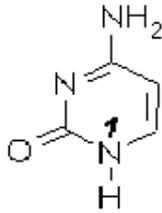


Figure 13.11: Cytosine

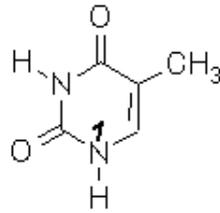


Figure 13.12: Thymine

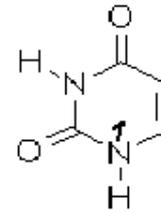


Figure 13.13: Uracil

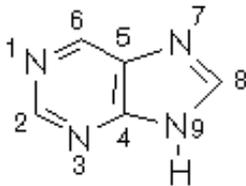


Figure 13.14: Purine

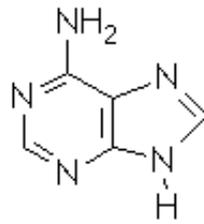


Figure 13.15: Adenine

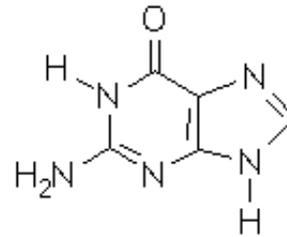


Figure 13.16: Guanine

(others are oxidation/reduction or for modification by substitution). In classifying sugars I follow the classification of Alberts *et al.* (1998) (pp. 53-57). These sugars are generally divided into four disjoint classes. These are (i) *monosaccharides*, (ii) *disaccharides*, (iii) *oligosaccharides* or *polysaccharides*<sup>21</sup>, and (iv) *complex saccharides*.

$$(13.47) \quad \forall x[Sugar(x) \leftrightarrow ((Monosaccharide(x) \vee Disaccharide(x) \vee OligoPolysaccharide(x) \vee CplxSaccharide(x)))]$$

In addition and to have the sugar classified one can state that sugar is a carbohydrate: *i.e.*  $\forall x[Sugar(x) \rightarrow Carbohydrate(x)]$ . More important are the -OH groups that are attached to these carbon atoms that don't have an oxygen atom bond. It should be noted that between this oxygen atom and the corresponding carbon atom there exists a double bond. I will refer to this double bond through the predicate  $=bond(x, y)$ <sup>22</sup>

The -OH groups can easily be introduced with the help of a default rule. The -OH group is a functional group and usually referred to as hydroxyl group. Thus I use *HydroxylGroup(r)* in (13.48).

$$(13.48) \quad \forall x, y[(Sugar(x) \wedge Carbon(y) \wedge \neg \exists z(Oxygen(z) \wedge =bond(z, y))) \supset \exists r(HydroxylGroup(r) \wedge =bond(x, r))]$$

<sup>21</sup>It is not clear whether both refer to the same group of sugars or not. One possibility is to define short chains of sugars as oligosaccharides and long chains as polysaccharides. This makes it necessary to define the distinction between the two, *i.e.* the length of the chains. Another possibility is to treat them as one group. The latter classification is preferred here.

<sup>22</sup>This predicate can be defined along the lines of (13.2).

It should be mentioned that with respect to this classification the four types of sugars are complementary. This can formally be expressed as shown in previous sections.

(i) Monosaccharides are discussed in detail below. (ii) Disaccharides can occur composed of two ring formations each of which is a monosaccharide<sup>23</sup>. More details on disaccharides are not necessary for the following sections, but in principle this formalisation is open to further extensions.

(iii) Oligo- and polysaccharides are either both linear or branched chains. The shorter ones are called oligosaccharides, the longer ones polysaccharides. (iv) All other conformations of saccharides that are not captured by the other classes with non-repetitive sequences of sugars are called complex saccharides.

>From all of these subgroups I only formalise some details of monosaccharides, *i.e.* to the degree that is necessary for correctly classifying ribose and 2-deoxyribose. These two sugars are necessary since they occur as building blocks for nucleosides. The other subgroups are not further formalised. Nonetheless, the given formalisation is open for including extensions at a later stage.

Along the lines of Alberts *et al.* (1998) within the class of monosaccharides, there are two different methods for orthogonal classification. The first follows a classification according to functional groups that are part of the monosaccharides. The second classification scheme depends on the length of the sugar. As regards the functional groups, two cases have to be mentioned. One case refers to monosaccharides bearing an aldehyde group. They are classified as aldoses. The second case refers to monosaccharide carrying a ketone group which can then be classified as a ketose. Since the aldose and ketose both might occur as ring structures, their reactive potential is not specified. This means that an aldose could be specified containing either an aldehyde functional group or a hemiacetal functional group and thus the sugar changes its conformation from chain structure to ring structure (see (13.49)). For the ketoses this means a change between a ketone and a hemiketal (see 13.50). It should be mentioned that each of the two functional groups for aldoses and ketoses are complementary.

$$(13.49) \quad \forall x[Aldose(x) \rightarrow \exists y(AldehydeGroup(y, x) \vee HemiAcetal(y, x))]$$

$$(13.50) \quad \forall x[Ketose(x) \rightarrow \exists y(KetoneGroup(y, x) \vee HemiKetal(y, x))]$$

Aldoses and ketoses are both monosaccharides as shown in (13.51). It should be mentioned that there exists a third class of monosaccharides, which is neglected in Alberts *et al.* (1998). This is the class of aldoketoses. This class refers to monosaccharides that can both be an aldose and a ketose (with the appropriate functional groups). However, I will disregard this class as well since it is not necessary for the following work.

---

<sup>23</sup>The three common disaccharides are maltose, lactose and sucrose, being composed of two glucose rings, a galactose and a glucose ring, or a glucose and a fructose ring respectively.

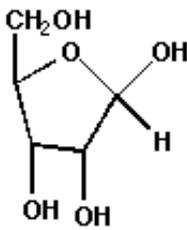
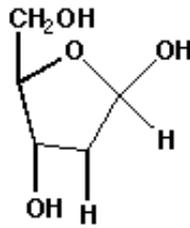
Figure 13.17:  $\beta$ -D-ribose

Figure 13.18: 2-deoxyribose

$$(13.51) \quad \forall x[\text{Monosaccharide}(x) \leftrightarrow ((\text{Aldose}(x) \vee \text{Ketose}(x)))]$$

The second (orthogonal) classification schema describes monosaccharides with the total formula  $(\text{CH}_2\text{O})_n$ <sup>24</sup>. The possible values for  $n$  are  $n = 3, 4 \dots 8$ . Depending on the length of  $n$ , the second classification schema is applied. Where  $n = 3$  they are called trioses (see (13.52)), where  $n = 4$  they are called tetroses.

$$(13.52) \quad \forall x[\text{Triose}(x) \leftrightarrow (\exists y_1, y_2, y_3(C(y_1, 1, x) \wedge C(y_3, 3, x) \wedge C(y_2, 2, x) \wedge \neg \exists y_i((y_i \neq y_1) \wedge (y_i \neq y_2) \wedge (y_i \neq y_3) \wedge C(y_i, i, x)) \wedge \text{Monosaccharide}(x)))]$$

Thus along the lines of (13.52) for  $n = 5$  they are called pentoses and can be defined as shown in (13.53). For  $n = 4$  and  $n > 5$  the axioms can be defined analogously.

$$(13.53) \quad \forall x[\text{Pentose}(x) \leftrightarrow (\exists y_1, y_2, y_3, y_4, y_5(C(y_1, 1, x) \wedge C(y_2, 2, x) \wedge C(y_3, 3, x) \wedge C(y_4, 4, x) \wedge C(y_5, 5, x) \wedge \neg \exists y_i((y_i \neq y_1) \wedge (y_i \neq y_2) \wedge (y_i \neq y_3) \wedge (y_i \neq y_4) \wedge (y_i \neq y_5)) \wedge C(y_i, i, x)) \wedge \text{Monosaccharide}(x)))]$$

There are two special types of monosaccharides that participate as building blocks for nucleosides. These are ribose (see Figure 13.17 for details) and 2-deoxyribose (see Figure 13.18 for details).

As already mentioned, these monosaccharidic carbon chains with their specific functional groups attached can also appear – in aqueous solution – as a ring structure (see 13.17 and 13.18). We have now the definitions of monosaccharides in terms of aldose

<sup>24</sup>The structural formula for aldose is  $\text{H}-[\text{CHOH}]_n-\text{CHO}$ , which is the same as the total formula given above. For a ketose the structural formula would thus be  $\text{H}-[\text{CHOH}]_n-\text{CO}-[\text{CHOH}]_m-\text{H}$ .

or ketoses and trioses, pentoses etc. at hand. This formalisation is for specifying further details of the chemical structure. With the given axioms it is now possible to define ribose as shown in (13.54).

$$(13.54) \quad \forall x[\text{Ribose}(x) \rightarrow (\text{Aldose}(x) \wedge \text{Pentose}(x))]$$

Finally we can now define two special types of ribose that serve as a scaffold for the nucleosides we have in mind. One is the  $\beta$ -D-ribose. The second one is the  $\beta$ -D-2-deoxyribose. The only difference is that one -OH group is replaced by a hydrogen atom (-H). The difference can be seen from the figures 13.17 and 13.18 . Note that these two types of ribose are complementary. This is formalised in (13.55) and (13.56).

$$(13.55) \quad \forall x[\beta\_D\_Ribose(x) \rightarrow (\neg\beta\_D\_2\_Deoxyribose(x))]$$

$$(13.56) \quad \forall x[\beta\_D\_2\_Deoxyribose(x) \rightarrow (\neg\beta\_D\_Ribose(x))]$$

In axioms (13.57) and (13.58)  $\beta$ -D-Ribose and  $\beta$ -D-2-Deoxyribose are now finally defined as special types of ribose.

$$(13.57) \quad \forall x[\beta\_D\_Ribose(x) \rightarrow (\text{Ribose}(x))]$$

$$(13.58) \quad \forall x[\beta\_D\_2\_Deoxyribose(x) \rightarrow (\text{Ribose}(x))]$$

>From the series of axioms that has been introduced up to now we can now infer that ribose has a carbon atom at position 1 (C-1 atom  $C(y_1, 1, x)$ ). This C-1 atom is the relevant binding position for the bases introduced previously to build nucleosides.

### 13.4.3 Linking sugars to molecules

The process of combining a sugar compound with a molecule is usually referred to as glycosylation<sup>25</sup>. The binding that is responsible for the connection of the two molecules

---

<sup>25</sup>From the definition given in Lackie and Dow (2000) it is not completely clear whether any molecule is a potential binding partner for the sugar. It is only mentioned that the adding of a sugar is treated analogously as in the case of adding glycan to chains of proteins. At least it can be assumed that proteins, lipids and bases can be glycosylated. As no restriction is mentioned in Lackie and Dow (2000) explicitly I use the predicate molecule in the following.

is thus called glycosidic binding. To be able to establish the link between the riboses defined in the previous section (see (13.54)) and the nucleobases (defined in (13.46)) I first introduce the binary predicate *glycosidic\_bond*. It has as one argument a sugar and as a second it accepts any kind of molecule.

$$(13.59) \quad \forall x, y \exists i, j, x_1, y_1 [glycosidic\_bond(x, y) \leftrightarrow (Sugar(x) \wedge Molecule(y) \wedge bond(x_1, y_1) \wedge C(x_1, i, x) \wedge \alpha(y_1, j, y))]$$

In molecular biology different types of glycosidic binding occur. The two most prominent ones are known as *N-glycosidic binding* and *O-glycosidic binding*.

$$(13.60) \quad \forall x, y [(N\_glycosidic\_bond(x, y) \vee O\_glycosidic\_bond(x, y)) \rightarrow (glycosidic\_bond(x, y))]$$

N-glycosidic binding and O-glycosidic binding are complementary. This is expressed through (13.61) and (13.62).

$$(13.61) \quad \forall x, y [N\_glycosidic\_bond(x, y) \rightarrow (\neg O\_glycosidic\_bond(x, y))]$$

$$(13.62) \quad \forall x, y [O\_glycosidic\_bond(x, y) \rightarrow (\neg N\_glycosidic\_bond(x, y))]$$

The two types of glycosidic binding are derived from the atom they bind to within the molecule. When the C-*i* atom of the sugar molecule binds to a nitrogen atom, it is called N-glycosidic binding, expressed by axiom (13.63).

$$(13.63) \quad \forall x, y \exists i, j, y_j, x_i [(N\_glycosidic\_bond(x, y) \wedge Sugar(x) \wedge Molecule(y)) \rightarrow (glycosidic\_bond(x, y) \wedge N(y_j, j, y) \wedge C(x_i, i, y) \wedge bond(x_i, y_j))]$$

When the C-*i* sugar atom binds to an oxygen atom in the molecule, it is called O-glycosidic binding, expressed through (13.64).

$$(13.64) \quad \forall x, y \exists i, j, y_j, x_i [(O\_glycosidic\_bond(x, y) \wedge Sugar(x) \wedge Molecule(y)) \rightarrow (glycosidic\_bond(x, y) \wedge O(y_j, j, y) \wedge C(x_i, i, y) \wedge bond(x_i, y_j))]$$

We can now use the types of glycosidic bindings introduced for building the nucleosides. In particular these bindings will be used to link the sugar atoms to the bases.

### 13.4.4 Linking sugars and bases

Before modelling the components of nucleotides we have to consider some details of nucleosides. A nucleoside is an assembly of a nucleobase – the most common ones are cytosine, uracil, thymine, guanine and adenine – and a pentose – the most common ones are ribose and deoxyribose –. The resulting nucleosides are named, corresponding to the sugar component, deoxyribosides or ribonucleosides. They are complementary, which can be axiomatised as already shown in previous sections.

$$(13.65) \quad \forall x [Nucleoside(x) \leftrightarrow (Deoxyriboside(x) \vee Ribonucleoside(x))]$$

It should be noted that these two nucleosides are complementary since the different type of attached sugar guides the different classification and thus the difference in naming. This disjointness is expressed in (13.66) and (13.67).

$$(13.66) \quad \forall x [Deoxyriboside(x) \rightarrow (\neg Ribonucleoside(x))]$$

$$(13.67) \quad \forall x [Ribonucleoside(x) \rightarrow (\neg Deoxyriboside(x))]$$

The ribonucleoside can now be defined as a composition of one ribose sugar part, *i.e.* the  $\beta\_D\_Ribose(x)$ , and a purine or a pyrimidine base part. The chemical binding between them can be specified as an N-glycosidic binding. This is expressed by (13.68).

$$(13.68) \quad \forall z \exists x, y [Ribonucleoside(z) \rightarrow (\beta\_D\_Ribose(x) \wedge (Purine(y) \vee Pyrimidine(y)) \wedge N\_glycosidic\_bond(x, y) \wedge z = x \oplus y)]$$

Along these lines deoxyribosides are composed of a deoxyribose sugar part, *i.e.* the  $\beta\_D\_2\_Deoxyribose(x)$ , and a purine or a pyrimidine base part. Again, the binding between them is carried out by an N-glycosidic binding. This is expressed by (13.69).

$$(13.69) \quad \forall z \exists x, y [(Deoxyriboside(z) \rightarrow ((\beta\_D\_Ribose(x) \vee \beta\_D\_2\_Deoxyribose(x)) \wedge (Purine(y) \vee Pyrimidine(y)) \wedge N\_glycosidic\_bond(x, y) \wedge z = x \oplus y)]$$

	Base	Ribonucleoside	Deoxyriboside
Purine {	Adenine	Adenosine (13.72)	Deoxyadenosine (13.77)
	Guanine	Guanosine (13.73)	Deoxyguanosine (13.78)
Pyrimidine {	Uracil	Uridine (13.75)	Deoxyuridine (13.79)
	Cytosine	Cytidine (13.74)	Deoxycytidine (13.80)
	Thymine	Thymine ribonucleoside (13.76)	Deoxythymidine (13.81)

Table 13.3: The table shows a classification of 10 different nucleosides. The classification depends on the type of ribose involved and the type of the base. The reference in the brackets indicate the corresponding axiom.

So far, two different ribose sugars and five different types of bases can occur as building blocks for the nucleosides. The table 13.4.4 shows the ten different possible pairings and their corresponding common names.

Of all these ten different nucleosides, only 4 occur as building building blocks for the nucleotides within DNA strands. Along these lines four nucleosides are used to build the RNA strands. With respect to DNA, these are deoxyadenosine, deoxyguanosine, deoxycytidine and deoxythymidine (often referred to as *thymidine*, since *thymidine* is not used within RNA). In addition, these five deoxyriboses are complementary. The four deoxyribosides are introduced in (13.70).

$$(13.70) \quad \forall x[\text{Deoxyriboside}(x) \leftrightarrow (\text{Deoxyadenosine}(x) \vee \text{Deoxyguanosine}(x) \vee \text{Deoxyuridine}(x) \vee \text{Deoxycytidine}(x) \vee \text{Deoxythymidine}(x))]$$

For RNA the relevant nucleosides are adenosine, guanosine, cytidine, and uridine. They are complementary as well. The taxonomic relationship that these are four ribonucleosides is expressed through (13.71).

$$(13.71) \quad \forall x[\text{Ribonucleoside}(x) \leftrightarrow (\text{Adenosine}(x) \vee \text{Guanosine}(x) \vee \text{Uridine}(x) \vee \text{Cytidine}(x) \vee \text{ThymineRibonucleoside}(x))]$$

These five different types of ribonucleosides differ with respect to the base part they contain. This is shown in the following five axioms:

**Adenosine** is thus composed of adenine and of the  $\beta\_D\_Ribose$  sugar part as can be seen from (13.72).

$$(13.72) \quad \forall x[\text{Adenosine}(x) \rightarrow$$

$$\exists y, z(Adenine(y) \wedge \beta\_D\_Ribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

**Guanosine** is composed of guanine and the  $\beta\_D\_Ribose$  sugar. Along the lines of adenosine (13.72) guanosine is formalised in (13.73).

$$(13.73) \quad \forall x[Guanosine(x) \rightarrow \exists y, z(Guanine(y) \wedge \beta\_D\_Ribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

**Cytidine** has cytosine as the necessary base and the  $\beta\_D\_Ribose$  sugar, formalised in (13.74).

$$(13.74) \quad \forall x[Cytidine(x) \rightarrow \exists y, z(Cytosine(y) \wedge \beta\_D\_Ribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

**Uridine** is thus composed of uracil and  $\beta\_D\_Ribose$  sugar. This is to be found expressed by axiom (13.75).

$$(13.75) \quad \forall x[Uridine(x) \rightarrow \exists y, z(Uracil(y) \wedge \beta\_D\_Ribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

**Thymine** has a thymine as base and the  $\beta\_D\_Ribose$  sugar. This is formalised in (13.76). To avoid confusion about thymine either being a base or a nucleoside, I distinguish between  $Thymine(x)$  and  $ThymineRibonucleoside(x)$ .

$$(13.76) \quad \forall x[ThymineRibonucleoside(x) \rightarrow \exists y, z(Thymine(y) \wedge \beta\_D\_Ribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

Along the lines of ribonucleosides we can now define the five different types of deoxyribosides. They differ with respect to the participating base part as well.

**Deoxyadenosine** is thus composed of adenine and  $\beta\_D\_2\_Deoxyribose$ , see (13.77).

$$(13.77) \quad \forall x[Deoxyadenosine(x) \rightarrow \exists y, z(Adenine(y) \wedge \beta\_D\_2\_Deoxyribose(z) \wedge x = y \oplus z \wedge N\_glycosidic\_bond(z, y))]$$

**Deoxyguanosine** is composed of the sugar  $\beta\_D\_2\_Deoxyribose$  and the base guanine. This is axiomatised by (13.78).

$$(13.78) \quad \forall x[Deoxyguanosine(x) \rightarrow \\ \exists y, z(Guanine(y) \wedge \beta\_D\_2\_Deoxyribose(z) \wedge x = y \oplus z \\ N\_glycosidic\_bond(z, y))]$$

**Deoxyuridine** has uridine as base and the sugar  $\beta\_D\_2\_Deoxyribose$  as parts formalised by (13.79).

$$(13.79) \quad \forall x, y, z[Deoxyuridine(x) \rightarrow \\ \exists y, z(Uridine(y) \wedge \beta\_D\_2\_Deoxyribose(z) \wedge x = y \oplus z \\ N\_glycosidic\_bond(z, y))]$$

**Deoxycytidine** is composed of the base cytosine and the sugar  $\beta\_D\_2\_Deoxyribose$ , which is to be found expressed by axiom (13.80).

$$(13.80) \quad \forall x[Deoxycytidine(x) \rightarrow \\ \exists y, z(Cytosine(y) \wedge \beta\_D\_2\_Deoxyribose(z) \wedge x = y \oplus z \\ N\_glycosidic\_bond(z, y))]$$

**Deoxythymidine** is the last of a series of deoxyribosides to be formalised. Similarly as above, there is one base part, *i.e.* thymine, and an obligatory sugar part, *i.e.*  $\beta\_D\_2\_Deoxyribose$ . This is expressed by (13.81).

$$(13.81) \quad \forall x[Deoxythymidine(x) \rightarrow \\ \exists y, z(Thymine(y) \wedge \beta\_D\_2\_Deoxyribose(z) \wedge x = y \oplus z \\ N\_glycosidic\_bond(z, y))]$$

### 13.4.5 Phosphate groups

The base parts are the coding parts within the DNA-strands and within the RNA-strands. In contrast to that, the sugar and the phosphate group together have the function of stabilising the strands acting as backbones for the coding parts. This is depicted in Figure 13.9 above. In the previous sections I described how sugar residues and base parts are combined in a model to build nucleosides. To be able to make the nucleotides complete we need the phosphate groups. A phosphate group consists of a phosphorus atom (with four valences) and four oxygen atoms attached to it ( $PO_4$ ). It is also called a mono-phosphate. Binding between two mono-phosphates results in a di-phosphate. In

this case one oxygen atom is shared by the two mono-phosphates ( $PO_7$ ). Analogously tri-phosphates can be built out of three mono-phosphates<sup>26</sup>. The binding between the phosphorus atom of one phosphate group to an oxygen atom of the following phosphate group is referred to as phosphoanhydride bond.

Since these phosphate groups occur frequently in biochemistry as part of other chemical compounds they are treated as functional groups. Similar to, for example, hydroxyl groups or amino groups the phosphate groups belong to the hydrophilic functional groups.

$$(13.82) \quad \forall x, y \exists z [(PhosphateGroup(x, y) \rightarrow (FunctionalGroup(x, y) \wedge Phosphorus(z) \wedge \sqsubset (z, x)))]$$

To ensure that each phosphorus atom is supplied with four oxygen atom a default rule is introduced.

$$(13.83) \quad \forall x, y, z [(PhosphateGroup(x, y) \wedge Phosphorus(z) \wedge \sqsubset (z, x)) \supset \exists^{=4} r (Oxygen(r) \wedge bound(r, z) \wedge \sqsubset (r, x))]$$

As in the case of carbon atoms being supplied with hydrogen atoms in axiom (13.40) one can introduce axioms for the case that one or more valences are already occupied. These axioms should be along the lines of (13.41) for the carbon atoms.

The specific types of phosphate groups can now be formalised as containing exactly the corresponding amount of phosphorus atoms:

**Monophosphates** have only one phosphorus atom and all properties of a phosphate group, *i.e.* being a functional group and having the free valences supplied by oxygen atoms.

$$(13.84) \quad \forall x, y \exists^{=1} z [Monophosphate(x, y) \rightarrow (PhosphateGroup(x, y) \wedge Phosphorus(z) \wedge \sqsubset (z, x))]$$

**Diphosphates** have two phosphorus atoms and the properties of phosphate groups as well.

$$(13.85) \quad \forall x, y \exists^{=2} z [Diphosphate(x, y) \rightarrow (PhosphateGroup(x, y) \wedge Phosphorus(z) \wedge \sqsubset (z, x))]$$

**Triphosphates** have three phosphorus atoms and the properties of phosphate groups.

$$(13.86) \quad \forall x, y \exists^{=3} z [Triphosphate(x, y) \rightarrow (PhosphateGroup(x, y) \wedge Phosphorus(z) \wedge \sqsubset (z, x))]$$

---

<sup>26</sup>Whenever a chemical reaction forces a tri-phosphate to split off a mono-phosphate, the result is a free ionic mono-phosphate, *i.e.*  $PO_3^-$ . It lacks one oxygen atom.

### 13.4.6 Ester bonds and nucleotides

This section describes the details of binding between mono-phosphate group to the nucleosides in order to form nucleotides. This binding is also referred to as a *phosphate ester of a nucleoside* as can be seen from Lackie and Dow (2000). The binding between the 5' carbon atom of the nucleoside and the oxygen atom of the phosphate group is called ester linkage, phosphoester bond or phosphate ester bond. These kind of bonds are derived from ester bonds. Ester bonds are bonds where one oxygen atom links to carbon chains, *i.e.* R-C-O-C-R, where R can possibly consist of one hydrogen at least. A phosphoester bond differs from the given ester bond in that one of the carbon atoms is replaced by a phosphorus atom. To build the nucleotides the phosphorus atom and the oxygen atom come from a mono-phosphate and are bound to a carbon atom of the nucleosides' sugar parts. The resulting nucleotide can also be described as a phosphate ester of a nucleoside. Since different kinds of phosphate groups can be linked – for example mono-, di-, tri-phosphate – the corresponding complexes are also called mono-, di-, tri-phosphonucleotides.

Axiom (13.87) shows an ester bond between two entities  $x$  and  $y$ . Each has a carbon atom participating, *i.e.*  $x_1$  and  $y_1$ . Both are bound to a connecting oxygen atom. I do not specify to which of the two entities it belonged before the binding took place.

$$(13.87) \quad \forall x, y [ester\_bond(x, y) \rightarrow \\ \exists x_1, y_1, z (covalent\_bond(x_1, z) \wedge covalent\_bond(z, y_1) \wedge \\ Carbon(x_1) \wedge Carbon(y_1) \wedge Oxygen(z) \wedge \\ \sqsubset (x_1, x) \wedge \sqsubset (y_1, y))] ]$$

An ester binding is a special type of binding. This taxonomic relationship is expressed by the axiom (13.88).

$$(13.88) \quad \forall x, y [ester\_bond(x, y) \rightarrow \\ bond(x, y)]$$

A phosphoester bond is a special type of ester bond. The major difference is that one of the participating atoms is a phosphorus instead of a carbon atom. Since I'm not tracking substitution operations in chemical compounds and thus do not have a substitution axiom that allows replacement of specific atoms I cannot express that *phosphoester\_bond* is a type of *ester\_bond* with this phosphorus substitution<sup>27</sup>. (13.89) shows for *phosphoester\_bond* similar properties as (13.87) for *ester\_bond*, except than one of the participating carbon atoms is replaced by a phosphorus atom.

<sup>27</sup>Different alternatives could be found. One is to use a default operator  $>$  instead of  $\rightarrow$  in axiom 13.88, which could then be over-ruled by a more specific type of (13.89). Still an axiom like  $\forall x, y phosphoester\_bond(x, y) \rightarrow ester\_bond(x, y)$  would not be possible.

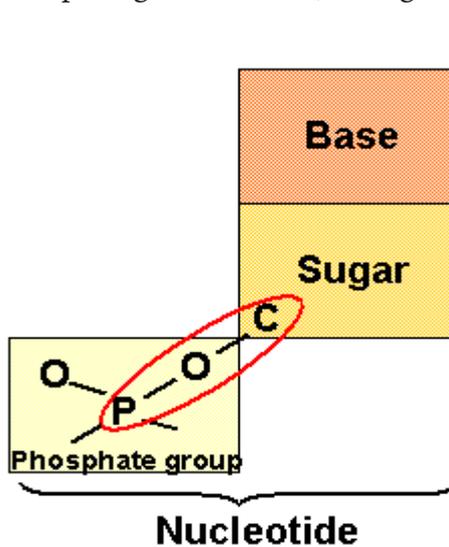
$$(13.89) \quad \forall x, y [phosphoester\_bond(x, y) \rightarrow \\ \exists x_1, y_1, z (bond(x_1, z) \wedge bond(z, y_1) \wedge \\ Phosphorus(x_1) \wedge Carbon(y_1) \wedge Oxygen(z) \wedge \\ \sqsubset (x_1, x) \wedge \sqsubset (z, x) \wedge \sqsubset (y_1, y))] ]$$

The taxonomic relationship that *phosphoester\_bond* is a type of binding (*i.e.* *bond(x, y)*) is given through axiom (13.90).

$$(13.90) \quad \forall x, y [phosphoester\_bond(x, y) \rightarrow \\ bond(x, y)]$$

### 13.4.7 Putting together the nucleotides

After a series of axioms we are now able to axiomatise nucleotides according to the participating molecules (see Figure 13.19).



Deoxyribonucleotides are the nucleotides that occur as monomeric building blocks in DNA sequences. Nucleotides are defined as the phosphate esters of nucleosides (see for example Lackie and Dow (2000)). The nucleosides have been introduced so far with the help of a series of axioms. In addition phosphate groups have been axiomatised and ester bonds have been introduced as well. A deoxyribonucleotide can now easily be defined as a nucleoside (*i.e.* a deoxyriboside with a deoxyribose sugar part) that has a mono-phosphate group attached through a phosphoester bond. This is shown schematically in figure (13.19) where the red oval indicates the phosphoester bond. The corresponding axiom that fuses the mentioned concepts with the phosphoester bond to a deoxyribonucleotide is expressed by axiom (13.91).

Figure 13.19: Schematic representation of a nucleotide with a mono-phosphate, a sugar and a base part and the phosphoester bond (red oval).

$$(13.91) \quad \forall z [Deoxyribonucleotide(z) \rightarrow \\ \exists x, y (x \oplus y = z \wedge Deoxyriboside(x) \wedge \\ Monophosphate(y, z) \wedge \\ phosphoester\_bond(x, y))] ]$$

Along the lines of the deoxyribonucleotides the ribonucleotides can now be defined formally as the nucleotides that occur as monomers within RNA sequences.

The main difference is with respect to the sugar part. Here we have connected the ribonucleoside (with the ribose sugar rather than deoxyribose sugar) with a phosphate group. Along the lines of DNA nucleotides, the deoxyribonucleotides, we can now define the RNA nucleotides, the ribonucleotides, as ribonucleosides bound through phosphoester bonds to monophosphates in axiom (13.92).

$$(13.92) \quad \forall z [Ribonucleotide(z) \rightarrow \exists x, y (x \oplus y = z \wedge Ribonucleoside(x) \wedge Monophosphate(x, y) \wedge phosphoester\_bond(x, y))]$$

As mentioned above, a nucleotide is either a ribonucleotide or a deoxyribonucleotide. This is shown in (13.93). That ribonucleotides and deoxyribonucleotides are disjoint can also be formalised as previously seen.

$$(13.93) \quad \forall x [Nucleotide(x) \leftrightarrow (Ribonucleotide(x) \vee Deoxyribonucleotide(x))]$$

The single nucleotides differ with respect to the base part and with respect to the different sugar parts. In table (13.4) the possible combinations are presented. As can be seen from the table, there exist four deoxyribonucleotides that occur within DNA sequence. These are deoxyadenylate, deoxyguanylate, deoxycytidylate, and thymidylate. The part-whole relations between the bases and the nucleotides have already been introduced in the rough formalisation at the very beginning of this chapter by axioms (13.8), (13.9), (13.10), and (13.11). The corresponding ribonucleotides have not been introduced but can easily be defined along these lines.

Base	Ribonucleoside-5'-triphosphate	Deoxyriboside-5'-triphosphate
Adenine	Adenosine-5'-triphosphate (ATP)	Deoxyadenosine-5'-triphosphate (dATP) <sup>+</sup>
Guanine	Guanosine-5'-triphosphate (GTP)	Deoxyguanosine-5'-triphosphate (dGTP)
Uracil	Uridine-5'-triphosphate (UTP)	Deoxyuridine-5'-triphosphate (dUTP)
Cytosine	Cytidine-5'-triphosphate (CTP)	Deoxycytidine-5'-triphosphate (dCTP)
Thymine	Thymine riboside-5'-triphosphate*	Deoxythymidine-5'-triphosphate (dTTP)
	<b>RNA-nucleotides except for (*)</b>	<b>DNA-nucleotides except for (+)</b>

Table 13.4: Table of Nucleotides

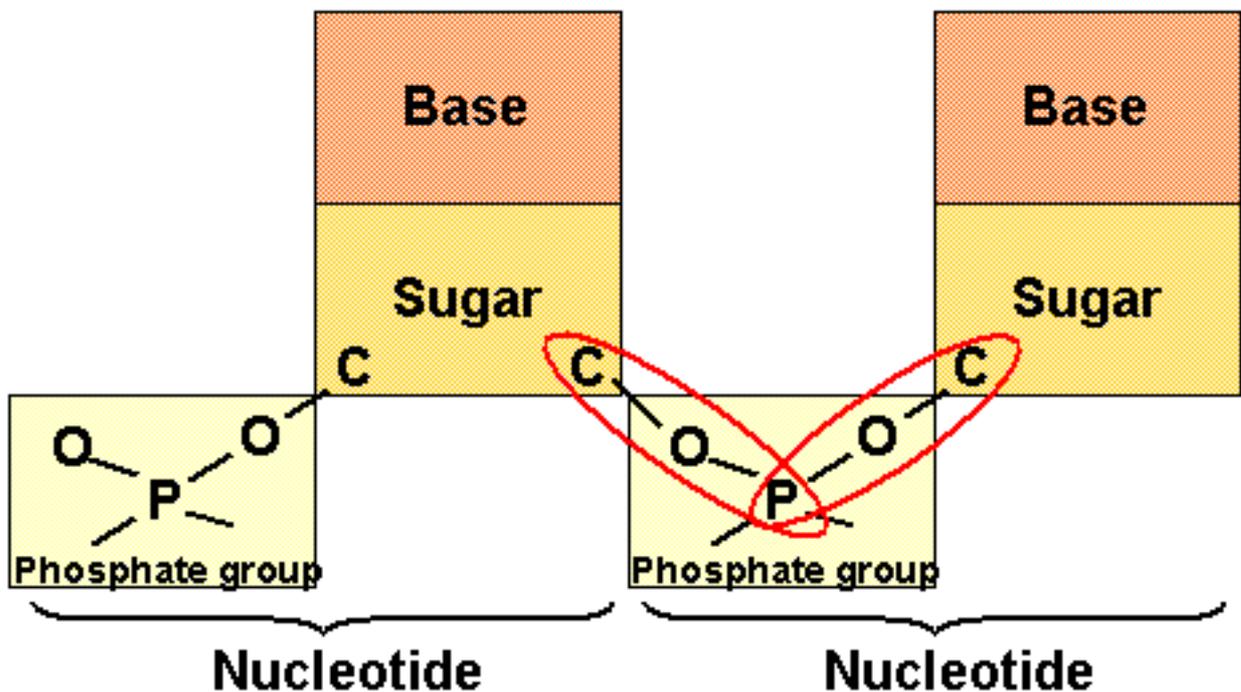


Figure 13.20: Schematic representation of two nucleotides each with a mono-phosphate, a sugar and a base part. The two phosphoester bonds that constitute the phospho-diester bond are both indicated through red ovals.

### 13.4.8 Building chains of nucleotides

We saw at the very beginning of this chapter that DNA sequences are chains of deoxyribonucleotides (see axiom (13.5)) and that RNA sequences are chains of ribonucleotides (see axiom (13.23)). However the binding relation that concatenates the nucleotides is not very specific, at least from a biochemical point of view. In the following I define phospho-diester bond as the main concatenating operation for building DNA and RNA strands.

A phospho-diester bond can be described as a special type of phosphoester bond where one phosphate group binds to two different oxygen atoms. In contrast to the definition of phosphate bond given above – where one phosphorus atom is bound to an oxygen which again is bound to a carbon atom – the phospho-diester bond consists of two of these phosphorus-oxygen-carbon bindings, where one phosphorus atom is shared. Sometimes a phospho-diester bond is also interpreted as a binding between two sugar compounds. Thus a phosphate group can act as a type of glue for building chains of nucleotides. It simply carries out phospho-diester bindings between pairs of nucleotides. This can be seen from the schematic figure 13.20. The formalisation is given by axiom (13.94).

It should be noted that the phospho-diester bonds links the two nucleotides leaving an unbound side at each end of the nucleotide chain. The unbound parts are different,

*i.e.* the unbound 5' and 3' end. The phosphate end is referred to as 5' end or upstream end, where the opposite end is the 3' end or downstream end. This allows not only the establishment of a linear order but also a direction. Usually this is 5' → 3' end.

$$(13.94) \quad \forall N_1, N_2 [(phospho\_diester\_bond(N_1, N_2) \wedge Nucleotide(N_2) \wedge Nucleotide(N_1)) \rightarrow \exists y_1, S_1, S_2, P_1 (phosphoester\_bond(S_1, P_1) \wedge Sugar(S_1) \wedge Phosphorus(P_1) \wedge \sqsubset (S_1, N_1) \wedge \sqsubset (P_1, N_1) \wedge phosphoester\_bond(S_2, P_1) \wedge Sugar(S_2) \wedge Phosphorus(P_1) \wedge \sqsubset (S_2, N_2))] ]$$

A DNA strand or DNA sequence can now easily be defined as a chain of nucleotides ( $i \dots k$ ) with a fixed length. The sequence of nucleotides is connected through phosphodiester bonds. The abbreviated notation ACG denotes a strand of three nucleotides consisting of deoxyadenylate mono-phosphate, deoxycitydylate mono-phosphate, and deoxyguanylate mono-phosphate linked through phosphodiester bonds. A nucleotide (or a set of nucleotides) appearing before another nucleotide (or set of nucleotides) is said to be *upstream*. Looking back to the previously introduced concept of  $dnaSequence(X)$  by axiom (13.5) we find that most of the work for defining a DNA strand or sequence has already been done. We can now easily specify the type of relation between the nucleotides. Instead of using the concatenation operator  $\bullet(x, y)$  we can now use  $phospho\_diester\_bond(N_1, N_2)$  to concatenate nucleotides. Thus a more fine grained version can be introduced by (13.95), which has in addition the two predicates  $5\_end(x_1)$  and  $3\_end(x_k)$  to indicate the direction of the sequence.

$$(13.95) \quad \forall X \exists [dnaSequence(X) \rightarrow \exists_{(i=1 \dots k)}^{x_i} (\bigwedge_{(i=1 \dots k)} Deoxyribonucleotide(x_i) \wedge (X = x_1 \oplus \dots \oplus x_k) \wedge (\bigwedge_{(j=1 \dots (k-1))} phospho\_diester\_bond(x_j, x_{j+1}) \wedge \neg(phospho\_diester\_bond(x_k, x_1)) \wedge 5\_end(x_1) \wedge 3\_end(x_k))] ]$$

In addition the RNA sequence can now be defined along the lines of (13.95). The introduced definitions allow us now to formalise DNA as well. A DNA is composed of two dna-strands. Both strands have the same length but run in opposite directions. They are held together by hydrogen bonds between the bases. Only adenine and thymine, as well as guanine and cytidine can be linked via these hydrogen bonds. However the concept of DNA is not needed for the gene expression relations and thus can be disregarded for the moment.

## 13.5 Amino acids and peptides

Another type of related entities, *i.e.* the potential result of gene expression, are proteins. Proteins are polymeric structures. They are built from monomers called amino acids. Ap-

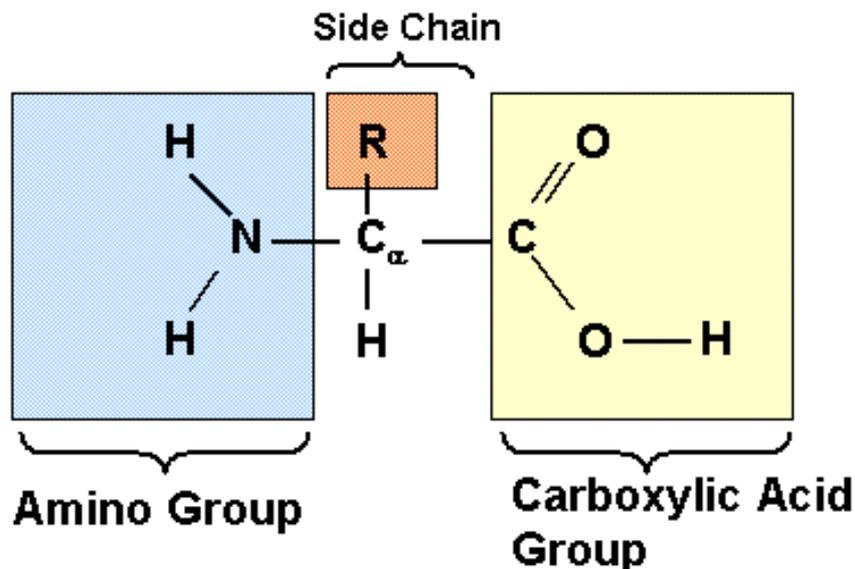


Figure 13.21: Schematic representation of an amino acid with an amino group, the carboxylic acid group and the side chain.

peering within chains the amino acids are called peptides. The structural and functional properties of peptides or proteins derive from the chemical properties of their parts and their linear arrangement. Within biology four levels of structural organisation are usually distinguished: primary, secondary, tertiary, and quaternary structure. The first level (*i.e.* the primary structure) refers to the linear arrangement of the amino acids, the polypeptide chain. The secondary structure is defined as the regular geometric figures that further shape the chain. The two most prominent are  $\alpha$ -helices and  $\beta$ -strands. The tertiary structure refers to the next level of folding that is carried out by the whole peptide chain. It refers to the three-dimensional grouping of helices or sheets of the entire peptide chain to complete proteins. At this point long-distance dependencies have to be mentioned, *i.e.* parts that are not immediately adjacent can be joined to form certain domains and motifs of a protein. The quaternary level describes the arrangement of multiple subunits, either from the same or from different polypeptide chains.

In the following I describe some structural and chemical properties of amino acids that are relevant for building the peptide chains. There exist 20 amino acids that occur as building blocks for proteins. Each amino acid has two parts that are crucial for carrying out the bonds between the amino acids (*i.e.* peptide bonds). The parts are the  $\alpha$ -carboxyl-group ( $-COOH$ ) and the  $\alpha$ -amino-group ( $-NH$ ). These are attached to a central carbon atom, usually referred to as  $\alpha$ -carbon atom. In addition each amino acid comes with a side chain, which is attached to this  $\alpha$ -carbon atom as well. On the basis of this side chain the 20 amino acids can be distinguished. The details of the binding between the  $\alpha$ -carboxyl-group and the  $\alpha$ -amino-group can be seen from the schematic figure in 13.21. The corresponding formalisation is given by axiom (13.96).

$$(13.96) \quad \forall x[AminoAcid(x) \rightarrow \\ \exists^1 y, z, c_\alpha, c_y, n_z(\alpha\_CarboxylGroup(y) \wedge \sqsubset (y, x) \wedge Carbon(c_y) \wedge \sqsubset (c_y, y) \\ \alpha\_AminoGroup(z) \wedge \sqsubset (z, x) \wedge Nitrogen(n_z) \wedge \sqsubset (n_z, z) \\ Carbon(c_\alpha) \wedge \sqsubset (c_\alpha, x) \wedge SideChain(s) \wedge \sqsubset (s, x) \wedge \\ bond(c_y, c_\alpha) \wedge bond(s, c_\alpha) \wedge bond(n_z, c))] ]$$

20 amino acids can be distinguished that only differ in their respective side chain, *i.e.* their chemical structure. In fact there are 22 names of amino acids to be found in the literature. This goes back to the fact that there are two cases where related amino acids are sometimes indistinguishable. In one case these are Asparagine and Aspartic acid (Asx, B), in the other these are Glutamine and Glutamic acid (Glx, Z). The following list shows the 22 names and gives the 3 letter as well as the one letter abbreviations (which are commonly in use) in brackets:

Alanine(Ala, A), Asparagine or aspartic acid (Asx, B), with Asparagine (Asn, N) and Aspartic acid (Asp, D), Arginine (Arg, R), Cysteine (Cys, C), Glutamine (Gln, Q), Glutamic acid (Glu, E), Glutamine or Glutamic acid (Glx, Z), Glycine (Gly, G), Histidine (His, H), Isoleucine (Ile, I), Leucine (Leu, L), Lysine (Lys, K), Methionine (Met, M), Phenylalanine (Phe, F), Proline (Pro, P), Serine (Ser, S), Threonine (Thr, T), Tryptophan (Trp, W), Tyrosine (Tyr, Y), Valine (Val, V). The different amino acids are formally introduced through (13.97). It should be noted that the amino acids are disjoint, which can easily be axiomatised as shown above.

$$(13.97) \quad \forall x[AminoAcid(x) \leftrightarrow \\ (Ala(x) \vee Asx(x) \vee Asn(x) \vee Asp(x) \vee \\ Arg(x) \vee Cys(x) \vee Gln(x) \vee Glu(x) \vee Glx(x) \vee \\ Gly(x) \vee His(x) \vee Ile(x) \vee Leu(x) \vee Lys(x) \vee \\ Met(x) \vee Phe(x) \vee Pro(x) \vee Ser(x) \vee \\ Thr(x) \vee Trp(x) \vee Tyr(x) \vee Val(x))] ]$$

As already mentioned, amino acids are grouped according to the characteristics of their side chain. Biologists distinguish according to the functional groups of the side chains seven groups of amino acids. These are aliphatic, aromatic, acidic, basic, hydroxylic, sulphur-containing, and amidic amino acids.

Given the parts of an amino acid a peptide bond can now be defined by an amide-binding between the  $\alpha$ -carboxyl-group from one amino acid and  $\alpha$ -amino-group from another amino acid. This is formally expressed by (13.98) on a general level.

$$(13.98) \quad \forall X, Y[(peptide\_bond(X, Y) \wedge \\ AminoAcid(X) \wedge AminoAcid(Y)) \rightarrow$$

$$\begin{aligned} & \exists x, y(x \sqsubset X \wedge y \sqsubset Y \wedge \\ & \text{amide\_bond}(x, y) \wedge \alpha\_AminoGroup(y) \wedge \\ & \alpha\_CarboxylGroup(x) \wedge X \neq Y) \end{aligned}$$

When inspecting the chemical details of polymeric chain formation through peptide bonds, the amino acids become bound to each other through a process called dehydration synthesis. Within this reaction water is lost between the carboxyl functional group (-COOH) and the amide functional group (-NH<sub>2</sub>). This reaction takes place within the ribosomes, which are complexes of proteins and RNA that translate gene sequences from mRNA into proteins. When building polypeptide chains peptide bonds are established between the nitrogen atom of the amide group and the carbon atom of the carboxyl group. This binding is called amide binding. The resulting bond is carried out between the carbon atom (with an oxygen double bound to it) of the carboxyl group and the nitrogen atom (with one hydrogen attached<sup>28</sup>) of the amino group. This amide bond between the nitrogen and the carbon atom with the oxygen atom attached is expressed through (13.99).

$$(13.99) \quad \forall x, y[\text{amide\_bond}(x, y) \rightarrow \\ (\text{Carbon}(x) \wedge \text{Nitrogen}(y) \wedge \text{bond}(x, y) \\ \text{Oxygen}(z) \wedge = \text{bond}(z, x))]$$

It should be noted that the *peptide\_bond* that has been introduced is a relation between amino acids. In contrast to that and according to IUPAC (IUPAC (1993)) an *amide\_bond* is the bond in -CONH<sub>2</sub> between the carbon atom and the nitrogen atom. However, the amide binding specifies the details of the peptide binding.

The amide binding as shown in (13.99) is used within (13.98) to define the peptide bonds between amino acids. The unbound ends of the chain are usually referred to as c-terminal, if the *α\_CarboxylGroup* is the unbound part, or n-terminal, if the *α\_AminoGroup* is the unbound part. The definition in (13.100) has now specified the details of the • relation in (13.25).

$$(13.100) \quad \forall X[\text{PeptideSequence}(X) \wedge \\ \exists_{(i=1 \dots k)}^{x_i} (\bigwedge_{(i=1 \dots k)} \text{AminoAcid}(x_i) \wedge (X = x_1 \oplus \dots \oplus x_k) \wedge \\ (\bigwedge_{(j=1 \dots (k-1))} \text{peptide\_bond}(x_j, x_{j+1})) \wedge \neg(\text{peptide\_bond}(x_k, x_1)) \wedge \\ nTerminal(x_1) \wedge cTerminal(x_k))]$$

At this point clearly many extensions of the presented work could be undertaken. To give an example the secondary, tertiary, and quaternary structures of proteins could be modelled. This work would imply a detailed formalisation of bio-spatial knowledge. Cohn (2001) shows how to formalise spatial knowledge. As a basis for his formalisation

---

<sup>28</sup>This could easily be guaranteed with default rules as already shown above.

he uses the part of a school biology textbook (Alberts *et al.* (1998)), that explains DNA structure. Although the formalisation presented specifies several partonomic relations between the chemical compounds, the spatial conformation is neglected. A possible extension of this work could be to attempt to integrate this work into the formalisation of Alberts *et al.* (1998). Although there is an overlap between the two approaches, the one that is presented here is more broadly arranged in that the basic concepts and relations of gene expression where DNA strands are a part is formalised.



# Chapter 14

## Summary

The last part of the present work is titled *Ontologies and Bio-Ontologies*. It consists of two related chapters. The first chapter explains the use of ontologies and their applications from a general point of view. The questions and items that are dealt with can be summed up as follows:

- What is the prime motivation for building ontologies?
- Why deal with ontologies? - moving towards applications.
- Why use ontologies for molecular biology?
- A scenario for using bio-ontologies.

In addition, I define what an ontology is and describe scenarios for the application of ontologies. In the following I introduce and discuss a set of criteria that are important when building ontologies. These comprise the following eight items: *expressivity of a language, complexity and efficiency, inference and interpretation, readability, clarity and visualisation, merging and integration of data, extensibility, granularity, and learning ontologies*. The first part concludes with a rough overview of different ontologies that are in use. I briefly present some general ontologies (*i.e.* Cyc and Wordnet) and in addition I present some biology-related ontologies.

The second chapter of the last part is titled *Building a bio-ontology*. In the beginning I present a linguistic justification for the use of ontologies. The guiding idea is that the previously described information extraction experiments are dependent on extensive manual work for writing rules both for the detection of named entities and for the detection of relations on top of the named entities. It should be noted that the more a system like the one presented is extended, the more rules have to be added, and the less it is possible to control the interactions between the rules. To reduce the amount of manual work, to better control this process of adding new rules as well as achieving a higher consistency, an ontology based approach is suggested for further research. I discuss and present potential applications for the use of ontologies, *i.e.* both to make implicit knowledge in biological databases more explicit as well as in order to achieve more consistency with biological data.

## Summary

The main part of this second chapter is titled *The basic gene expression concepts and relations* and it introduces briefly the biological process of gene expression on a general level. This general biology-driven picture is then axiomatised in a formal approach. The main concepts that are formally expressed are *DNA sequence* and its parts, the *Nucleotides*. In addition a rough conceptualisation of *Genome* and *Gene* is introduced as well as the corresponding part *Coding sequence* and *Non-coding sequence*. The subunits of these are introduced as well: *regulatory non-coding sequence*, *promoter*, *enhancer* and *binding site*. The products of the gene transcription and translation process are formalised in the following, being *RNA sequence*, *gene transcript*, *peptide sequence*, *protein*. Finally the formalisation on this level finishes with the relations *transcription* and *translation*.

This biology-driven formalisation is then followed by a chemistry-driven formalisation of gene expression consisting of three sections. The section *Chemical elements and chemical bonds*, section 13.3, inspects the details of chemical binding between atoms as well as binding between molecules. Within the section *Building chains of nucleotides*, section 13.4 the composition of nucleotide chains, *i.e.* DNA-strands and RNA-strands, is formally developed. The basic building blocks are the nucleotides, which themselves are built of the purine and pyrimidine bases, a sugar part (*i.e.* *ribose* and *2-deoxyribose*) and a phosphate group. The compositional structure with specific chemical properties is formalised and presented. In the last step the DNA and RNA-strands are composed of the corresponding and previously introduced nucleotides. The last section *Amino acids and peptides*, section 13.5, concludes with the potential result of gene expression, the proteins. Proteins are polymeric structures as well as the DNA strands and are formalised along the same lines.

Although roughly 100 axioms are presented, the main contribution of this modelling effort is not that all possible concepts and relations with respect to gene expression are formalised. The crucial contributions are two insights that are gained when setting up such a model. The first refers to questions of granularity. This is presented in the context moving from biological to chemical properties and showing the different perspectives within one and the same conceptualisation. The second insight is that depending on what is to be expressed (*i.e.* is the focus on the event structure of biomedical processes or is the focus on partonomic relations between biomedical entities) the logical framework has to meet the different needs and thus has to be chosen well.

**Part V**  
**Conclusion**



This chapter summarises the major contributions of this thesis. In addition an outline for further research activities is given.

## 14.1 Contributions

### The main contributions of this work

From what is reported in the introductory sections it has become clear that there is a great need for NLP methods in the field of bioinformatics. A series of applicable NLP approaches is presented for the processing of the huge amount of heterogeneous and complex data in the field of molecular biology or biomedicine.

The present work shows that the field of NLP can provide methods for dealing with biological data be it from databases or be it textual data for achieving high quality results. High quality results are a prerequisite in order to obviate or at least to reduce labour intensive manual curation of these results. These results usually are fed into databases to make them accessible to other researchers. It should be noted that the high quality and relevance of the results can only be assured if biological knowledge is made available for computational linguists' methods. This basically means that the extraction of biologically meaningful data can only be guaranteed in an interdisciplinary setting where computational linguists, bioinformaticians and biologists work jointly. This collaborative effort is key for a successful outcome. It should be noted that the communicative effort between the disciplines is not trivial and has to guarantee at least that:

- i The domain experts from the biomedical domain understand the NLP techniques employed are able to specify their needs according to what the limitations of NLP methods are.
- ii The NLP experts should be able to understand what the biology behind the biologists' needs is in order to be able to propose appropriate solutions.
- iii A collective effort has to be undertaken for the implementation of such a system. When building a rule-based information extraction system this ensures that each rule and each lexical entry that is specified makes biological sense and is processable by NLP software.

From an NLP perspective a major contribution is that it has been shown that rule-based IE systems are not necessary systems serving a single purpose but can be adapted to related questions with reasonable efforts. Given that some guidelines are followed - such as modularisation of entity recognition and relation recognition - these modules can be re-used and/or adapted later on for related questions.

## 14.2 Further Research

### 14.2.1 Extending the biological domain

A first issue for further research is already being addressed, that is the extension of the existing information extraction system to deal with related questions. As mentioned in the summary, initial results of these extensions were presented at the SMBM Symposium 2005. This is still ongoing work and new entities such as chemical compounds should be included in the next version of the named entity extraction component. In a second step we then intend to extract relations between the chemical compounds and the proteins which have already been recognised.

### 14.2.2 Improving the methods for information extraction

Some first efforts have already been made for the use of more sophisticated methods for the information extraction system. A limitation of the information extraction system described in the thesis is the expressivity of the grammar formalism. This means that the finite-state approach that has been used is limited with respect to the linguistic constructions it can deal with. In order to be able to detect more complex linguistic constructions, such as relative clauses, a context-free grammar will be used. However, very few groups (if at all) have managed to fully parse a relevant part of PubMed. As a first step we have set up an experiment that strongly encourages us to use a combination of finite-state tools to pre-process the PubMed abstracts and use this as input for the full-parser. The finite-state grammar recognises so-called core noun phrases, which are fed into the full-parser.

### 14.2.3 Ontologising TigerSearch

Another issue for further research which should be undertaken is the use of an (explicit) ontology to extend TigerSearch options for the querying of biomedical databases. A further continuation of this work includes the development of a parser and an ontology that allows the structuring of this information. This makes the data accessible to queries that use concepts from this ontology and hence abstract over the arbitrariness of the free text form. Descriptions associated with regions, like ASP/GLU-RICH, ACIDIC, CYTOPLASMIC TAIL, GOGO-A0101 ALPHA CHAIN could then be translated to feature value pairs like the following:

SEQ_CHAR	ASP/GLU-RICH
???	ACIDIC
LOC	CYTOPLASMIC
NAME	GOGO-A0101 ALPHA CHAIN

For a translation like the one proposed, a lexicon associating the single lexical items with ontological categories is required. Here it is necessary to know that *cytoplasmic* in

the above mentioned example refers to the cytoplasm of a cell, and that the cytoplasm of a cell is a location. Analogously such an approach will allow us to annotate *extracellular* with the same ontological concept, *i.e.* location.

#### 14.2.4 Further development of the gene expression model

It is planned to further develop the proposed model of gene expression in mainly two directions. First, the refinement and extension of the model. This implies that a series of default axioms for defining further knowledge on the chemical structure of the entities have to be added. Furthermore, work has to be invested in elaborating the description of chemical reactions comprising transcription and translation. The overall benefit would be to provide more information about the relations between the introduced concepts. A second issue to address is the implementation of this proposed model. I have already mentioned a series of possible applications. In addition this model could further be used to work on the results described in Reyle (2005). This work mainly describes the translation of chemical compound names to structural representations of these chemical entities. Given an ontological model like the one described in this work, it would connect structural information about the chemical compounds with functional information, based on the reactions that they can undergo.



**Part VI**  
**Appendix**



# Appendix A

## Swiss-Prot entries

### A.1 A Swiss-Prot example entry

ID GRAA\_HUMAN STANDARD; PRT; 262 AA.  
AC P12544;  
DT 01-OCT-1989 (Rel. 12, Created)  
DT 01-OCT-1989 (Rel. 12, Last sequence update)  
DT 01-OCT-2004 (Rel. 45, Last annotation update)  
DE Granzyme A precursor (EC 3.4.21.78) (Cytotoxic T-lymphocyte proteinase  
DE 1) (Hanukkah factor) (H factor) (HF) (Granzyme 1) (CTL tryptase)  
DE (Fragmentin 1).  
GN Name=GZMA; Synonyms=CTLA3, HFSP;  
OS Homo sapiens (Human).  
OC Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;  
OC Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.  
OX NCBI\_TaxID=9606;  
RN [1]  
RP SEQUENCE FROM N.A.  
RC TISSUE=T-cell;  
RX MEDLINE=88125000; PubMed=3257574;  
RA Gershenfeld H.K., Hershberger R.J., Shows T.B., Weissman I.L.;  
RT "Cloning and chromosomal assignment of a human cDNA encoding a T cell-  
RT and natural killer cell-specific trypsin-like serine protease."  
RL Proc. Natl. Acad. Sci. U.S.A. 85:1184-1188(1988).  
RN [2]  
RP SEQUENCE FROM N.A.  
RC TISSUE=Blood;  
RX MEDLINE=22388257; PubMed=12477932; DOI=10.1073/pnas.242603899;  
RA Strausberg R.L., Feingold E.A., Grouse L.H., Derge J.G.,  
RA Klausner R.D., Collins F.S., Wagner L., Shenmen C.M., Schuler G.D.,  
RA Altschul S.F., Zeeberg B., Buetow K.H., Schaefer C.F., Bhat N.K.,  
RA Hopkins R.F., Jordan H., Moore T., Max S.I., Wang J., Hsieh F.,  
RA Diatchenko L., Marusina K., Farmer A.A., Rubin G.M., Hong L.,  
RA Stapleton M., Soares M.B., Bonaldo M.F., Casavant T.L., Scheetz T.E.,

*Swiss-Prot entries*

RA Brownstein M.J., Usdin T.B., Toshiyuki S., Carninci P., Prange C.,  
RA Raha S.S., Loquellano N.A., Peters G.J., Abramson R.D., Mullahy S.J.,  
RA Bosak S.A., McEwan P.J., McKernan K.J., Malek J.A., Gunaratne P.H.,  
RA Richards S., Worley K.C., Hale S., Garcia A.M., Gay L.J., Hulyk S.W.,  
RA Villalon D.K., Muzny D.M., Sodergren E.J., Lu X., Gibbs R.A.,  
RA Fahey J., Helton E., Ketteman M., Madan A., Rodrigues S., Sanchez A.,  
RA Whiting M., Madan A., Young A.C., Shevchenko Y., Bouffard G.G.,  
RA Blakesley R.W., Touchman J.W., Green E.D., Dickson M.C.,  
RA Rodriguez A.C., Grimwood J., Schmutz J., Myers R.M.,  
RA Butterfield Y.S.N., Krzywinski M.I., Skalska U., Smailus D.E.,  
RA Schnerch A., Schein J.E., Jones S.J.M., Marra M.A.;  
RT "Generation and initial analysis of more than 15,000 full-length human  
RT and mouse cDNA sequences.";  
RL Proc. Natl. Acad. Sci. U.S.A. 99:16899-16903(2002).  
RN [3]  
RP SEQUENCE OF 1-23 FROM N.A.  
RA Goralski T.J., Krensky A.M.;  
RT "The upstream region of the human granzyme A locus contains both  
RT positive and negative transcriptional regulatory elements.";  
RL Submitted (NOV-1995) to the EMBL/GenBank/DDBJ databases.  
RN [4]  
RP SEQUENCE OF 29-53.  
RX MEDLINE=88330824; PubMed=3047119;  
RA Poe M., Bennett C.D., Biddison W.E., Blake J.T., Norton G.P.,  
RA Rodkey J.A., Sigal N.H., Turner R.V., Wu J.K., Zweerink H.J.;  
RT "Human cytotoxic lymphocyte tryptase. Its purification from granules  
RT and the characterization of inhibitor and substrate specificity.";  
RL J. Biol. Chem. 263:13215-13222(1988).  
RN [5]  
RP SEQUENCE OF 29-40, AND CHARACTERIZATION.  
RX MEDLINE=89009866; PubMed=3262682;  
RA Hameed A., Lowrey D.M., Lichtenheld M., Podack E.R.;  
RT "Characterization of three serine esterases isolated from human IL-2  
RT activated killer cells.";  
RL J. Immunol. 141:3142-3147(1988).  
RN [6]  
RP SEQUENCE OF 29-39, AND CHARACTERIZATION.  
RX MEDLINE=89035468; PubMed=3263427;  
RA Kraehenbuhl O., Rey C., Jenne D.E., Lanzavecchia A., Groscurth P.,  
RA Carrel S., Tschopp J.;  
RT "Characterization of granzymes A and B isolated from granules of  
RT cloned human cytotoxic T lymphocytes.";  
RL J. Immunol. 141:3471-3477(1988).  
RN [7]  
RP 3D-STRUCTURE MODELING.  
RX MEDLINE=89184501; PubMed=3237717;  
RA Murphy M.E.P., Moulton J., Bleackley R.C., Gershenfeld H.,

## A.1 A Swiss-Prot example entry

RA Weissman I.L., James M.N.G.;  
RT "Comparative molecular model building of two serine proteinases from  
RT cytotoxic T lymphocytes";  
RL Proteins 4:190-204(1988).  
CC -!- FUNCTION: This enzyme is necessary for target cell lysis in cell-  
CC mediated immune responses. It cleaves after Lys or Arg. May be  
CC involved in apoptosis.  
CC -!- CATALYTIC ACTIVITY: Hydrolysis of proteins, including fibronectin,  
CC type IV collagen and nucleolin. Preferential cleavage: Arg-|-Xaa,  
CC Lys-|-Xaa >> Phe-|-Xaa in small molecule substrates.  
CC -!- SUBUNIT: Homodimer; disulfide-linked.  
CC -!- SUBCELLULAR LOCATION: Secreted; cytoplasmic granules.  
CC -!- SIMILARITY: Belongs to peptidase family S1. Granzyme subfamily.  
CC -----  
CC This SWISS-PROT entry is copyright. It is produced through a collaboration  
CC between the Swiss Institute of Bioinformatics and the EMBL outstation -  
CC the European Bioinformatics Institute. There are no restrictions on its  
CC use by non-profit institutions as long as its content is in no way  
CC modified and this statement is not removed. Usage by and for commercial  
CC entities requires a license agreement (See <http://www.isb-sib.ch/announce/>  
CC or send an email to [license@isb-sib.ch](mailto:license@isb-sib.ch)).  
CC -----  
DR EMBL; M18737; AAA52647.1; -.  
DR EMBL; BC015739; AAH15739.1; -.  
DR EMBL; U40006; AAD00009.1; -.  
DR PIR; A31372; A31372.  
DR PDB; 1HF1; Model; @=29-262.  
DR PDB; 1OP8; X-ray; A/B/C/D/E/F=29-262.  
DR PDB; 1ORF; X-ray; A=29-262.  
DR MEROPS; S01.135; -.  
DR Genew; HGNC:4708; GZMA.  
DR MIM; 140050; -.  
DR GO; GO:0004277; F:granzyme A activity; TAS.  
DR InterPro; IPR001254; Peptidase\_S1.  
DR InterPro; IPR001314; Peptidase\_S1A.  
DR InterPro; IPR009003; Pept\_Ser\_Cys.  
DR Pfam; PF00089; Trypsin; 1.  
DR PRINTS; PR00722; CHYMOTRYPSIN.  
DR PROSITE; PS50240; TRYPSIN\_DOM; 1.  
DR PROSITE; PS00134; TRYPSIN\_HIS; 1.  
DR PROSITE; PS00135; TRYPSIN\_SER; 1.  
KW 3D-structure; Apoptosis; Cytolysis; Direct protein sequencing;  
KW Hydrolase; Serine protease; Signal; T-cell; Zymogen.  
FT SIGNAL 1 26  
FT PROPEP 27 28 Activation peptide.  
FT CHAIN 29 262 Granzyme A.  
FT ACT\_SITE 69 69 Charge relay system (By similarity).

*Swiss-Prot entries*

FT	ACT_SITE	114	114	Charge relay system (By similarity).
FT	ACT_SITE	212	212	Charge relay system (By similarity).
FT	DISULFID	54	70	By similarity.
FT	DISULFID	148	218	By similarity.
FT	DISULFID	179	197	By similarity.
FT	DISULFID	208	234	By similarity.
FT	CARBOHYD	170	170	N-linked (GlcNAc...) (Potential).
FT	STRAND	30	30	
FT	STRAND	33	34	
FT	TURN	37	38	
FT	TURN	41	42	
FT	STRAND	43	48	
FT	TURN	49	51	
FT	STRAND	52	60	
FT	TURN	61	62	
FT	STRAND	63	66	
FT	TURN	68	69	
FT	STRAND	76	80	
FT	STRAND	84	84	
FT	TURN	85	86	
FT	TURN	90	91	
FT	STRAND	93	102	
FT	TURN	104	105	
FT	HELIX	108	110	
FT	TURN	112	113	
FT	STRAND	116	120	
FT	STRAND	127	127	
FT	TURN	128	129	
FT	STRAND	130	130	
FT	STRAND	134	134	
FT	TURN	138	139	
FT	TURN	144	145	
FT	STRAND	147	152	
FT	STRAND	155	157	
FT	TURN	158	159	
FT	STRAND	160	162	
FT	STRAND	165	165	
FT	STRAND	167	173	
FT	HELIX	176	180	
FT	TURN	181	182	
FT	TURN	184	185	
FT	TURN	193	194	
FT	STRAND	195	199	
FT	TURN	201	202	
FT	STRAND	206	206	
FT	TURN	209	210	
FT	TURN	212	213	

## A.2 HTML version of a Swiss-Prot entry

```
FT STRAND      215      218
FT TURN        219      220
FT STRAND      221      228
FT TURN        231      232
FT TURN        234      235
FT TURN        237      238
FT STRAND      241      245
FT TURN        246      249
FT HELIX       252      260
**
** ##### INTERNAL SECTION #####
**CL 5q11-q12;
SQ SEQUENCE    262 AA;  28968 MW;  DA87363A0D92BAF4 CRC64;
MRNSYRFLAS SLSVVVSLLL IPEDVCEKII GGNEVTPHSR PYMVLLSLDR KTICAGALIA
KDWVLTAAHC NLNKRQVIL GAHSITREEP TKQIMLVKKE FPYPCYDPAT REGDLKLLQL
TEKAKINKYV TILHLPKKGD DVKPGTMCQV AGWGRTHNSA SWSDTLREVN ITIIDRKVCN
DRNHYNFNPV IGMNMVCAGS LRGGRDSCNG DSGSPLLCEG VFRGVTSFGL ENKCGDPRGP
GVYILLSKKH LNWIIMTIKG AV
//
```

## A.2 HTML version of a Swiss-Prot entry

The screenshot shows a web browser window titled "NiceProt View of TrEMBL: Q9BR43 - Microsoft Internet Explorer". The address bar shows the URL "http://www.expasy.org/cgi-bin/niceprot.pl?Q9BR43". The main content area displays the entry for Q9BR43 with various sections:

- Entry information:**

Entry name	Q9BR43
Primary accession number	Q9BR43
Secondary accession numbers	None
Entered in TrEMBL in	Release 17, June 2001
Sequence was last modified in	Release 17, June 2001
Annotations were last modified in	Release 22, October 2002
- Name and origin of the protein:**

Protein name	DJ1093G12.6 [Fragment]
Synonym	A novel protein
Gene name	DJ1093G12.6
From	Homo sapiens (Human) [TaxID: 9606]
Taxonomy	Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
- References:**

[1] SEQUENCE FROM NUCLEIC ACID.  
Lloyd D.  
Submitted (FEB-2001) to the EMBL/GenBank/DDBJ databases.
- Comments:** None
- Cross-references:**

EMBL	AL121751; CAC28878.1; - [EMBL / GenBank / DDBJ] [CoDingSequence]
Genew	HGNC:16224; C20orf93.
CleanEx	HGNC:16224; C20orf93.
GeneCards	14099240

Figure A.1: A screenshot of a database entry in Swissprot. Suprisingly “a novel protein” occurs as a synonym.

# Appendix B

## The regular expression grammar

### B.1 Level 1

```
#####  
#  
:Level1  
#  
#####  
  
# Prenumeral Chunk  
cdqlx -> as much as | more than | about | only | well? over;  
  
# Numeral Chunk  
cdx -> (cdql|cdqlx) cd  
      | (cdql|cdqlx)? cd+ cd;  
  
# Measure Phrase Chunk  
mx -> (cd|cdx) h=(units|tunits)  
      | (cdql|cdqlx)? dt-a h=(unit|tunit);  
  
# Protein or gene name with numeral and symbol modifiers  
nnpqx -> ((cd | sym)* h=nnpq (cd | sym)*+);  
  
# Capture stuff like UAS 2  
uasx -> uas cd?;  
ursx -> urs cd?;  
  
# Protect "be vvn" phrases  
bevvn -> (be|bez|ber|bedz|bedr) vvn;  
  
# Protect "vvn by" phrases  
vvnby -> vvn by;  
  
# recognising sequences of amino acids
```

```
aasequ -> aa aa+;
```

## B.2 Level 2

```
#####  
#  
:Level2  
#  
# Level identifies chunks of one or more protein/gene names  
#  
#####  
#  
# Variable declaration  
#  
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);  
ADVHD = rb | cdql | then | well;  
ADV = ADVHD | rbr | more | rbs | ql;  
NUM = cd | cdx;  
JX = ADV? (jj | jjr | jjs);  
JXC = JX (cma JX)* (cc | cma) JX;  
ADJ = JX | JXC | mx;  
PTC = ADV* (vbn | vbg);  
MOD = (org | ADJ | PTC | vvn | cd | sym);  
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);  
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;  
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;  
  
#  
# Groups and lists of protein/gene names  
#  
nxpg -> PRE (nnp gx cc)? nnp gx POST  
      | PRE nnp gx (cma (nnp gx | nn))+ cma? cc nnp gx POST;  
  
#  
# nxpg dependent/independent/responsive  
#  
nxpg_dep -> PRE nxpg (dependent | independent | responsive) POST;
```

## B.3 Level 3

```
#####
```

```

#
:Level3
#
# Level identifies biological entities such as genes and proteins
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;

#
# Genes
#
nxgene -> PRE nxpg (POST (bez | ber) PRE)? ((prot | enzyme |
    prokin) \ coding)? h=gene (nxpg | cma nxpg cma? |
    \ ( PRE nxpg POST \ ))? POST | PRE ((prot | enzyme |
    prokin) coding)? h=gene ((from | in | of)? MOD)*
    (nxpg | cma nxpg cma? | \ ( PRE nxpg POST \ )) POST;

```

## B.4 Level 4

```

#####
#
:Level4
#
# Level identifies biological entities such as genes and proteins
#
#####
#
# Variable declaration
#

```

## The regular expression grammar

```
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;

#
# RNAs and transcripts
#
nxrna -> PRE (nxpg | nxgene) (POST (bez | ber | codv) PRE)?
        h=(rna | trans) (nxpg | nxgene | cma (nxpg | nxgene) cma?
        | \( PRE (nxpg | nxgene) POST \))? POST | PRE h=(rna |
        trans) ((from | in | of)? MOD)* (nxpg | nxgene | cma
        (nxpg | nxgene) cma? | \( PRE (nxpg | nxgene) POST \)) POST;

#
# Proteins
#
nxprot -> PRE nxpg (POST (bez | ber | codv) PRE)?
        h=(prot | enzyme | prokin) (nxpg | cma nxpg cma? | \(
        PRE nxpg POST \))? POST | PRE h=(prot | enzyme | prokin)
        ((from | in | of)? MOD)* (nxpg | cma nxpg
        cma? | \( PRE nxpg POST \)) POST;

#
# Transcription factors
#
nxtf -> PRE nxpg (POST (bez | ber | codv) PRE)? h=tf prot?
        cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST \))? POST |
        PRE h=tf prot? cmpx? ((from | in | of)? MOD)* (nxpg | cma
        nxpg cma? | \( PRE nxpg POST \)) POST;
```

## B.5 Level 5

```
#####
#
```

```

:Level5
#
# Level identifies parts of biological entities
#
#####
#
# Variables declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | np | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;

#
# Alleles
#
nxalle -> PRE (nxpg | nxgene) MOD* h=alle POST
        | PRE h=alle of? (nxpg | nxgene) POST;

#
# Diploids
#
nxdiploid -> PRE nxpg h=diploid POST;

#
# Domain
#
nxdom -> PRE nxpg h=(dom | dnabddom | bddom | kindom) POST
        | PRE h=(dom | dnabddom | bddom | kindom) nxpg POST;

#
# Open reading frames
#
nxorf -> PRE (nxpg | nxgene) MOD* h=(orf | coding sequ) POST
        | PRE h=(orf | coding sequ) of? (nxpg | nxgene) POST;

#
# Promoters
#

```

## The regular expression grammar

```
nxprom -> PRE (nxpg | nxgene) MOD* h=prom fragment? POST
        | PRE h=prom fragment? of? (nxpg | nxgene) POST;
```

## B.6 Level 6

```
#####
#
:Level6
#
# Level identifies products of biological entities
#
#####
#
# Variables declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
ALT = (aa | aasequ | actdom | actr | bddom | bs | dnabddom | enzyme |
expr | fragment | gene | kindom | org | orf | prod | prod | prokin |
prot | rna | rnaspec | regdom | repdom | transc | trans | tract |
trrep | ursx | uasx);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;

#
# Gene products.
#
nxgeneprod -> PRE ALT* (nxpg | nxgene | nxorf) ALT* NUM? transl?
             h=(prod | prods) POST
             | PRE gene? transl? h=(prod | prods) of? (nxpg |
             nxgene | nxorf) transl? (cma nxpg cma? | \( nxpg \))? POST;
```

## B.7 Level 7

```
#####
#
:Level7
#
# Level identifies protein complexes
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym );
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPROT = (nxprot | nextf | nxgeneprod);

#
# Complexes
#
nxcmpx -> PRE (prot | enzyme | prokin | tf)? h=(cmpx | cmpxs) of?
          (nxpg | nxrna | NXPROT) POST | PRE (nxpg | nxrna | NXPROT)
          h=(cmpx | cmpxs) POST;
```

## B.8 Level 8

```
#####
#
:Level7
#
# Level identifies biological entities such as genes and proteins
#
#####
#
```

## The regular expression grammar

```
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPG = (nxpg | nxgene | nxrna | nxprot | nxtf | nxgeneprod | nxcmpx);

#
# Event: "X transcriptional activator/repressor/regulator R"
#
nxev_tractr -> PRE nxpg h=tractr prot? cma? nxpg POST;
nxev_trrepr -> PRE nxpg h=trrepr prot? cma? nxpg POST;
nxev_trregr -> PRE nxpg h=trregr prot? cma? nxpg POST;

#
# Transcription activator/repressor/regulator agents
#
ntractr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=tractr
    prot? cmpx? (nxpg | cma nxpg cma? |
    \( PRE nxpg POST \))? POST
    | PRE h=tractr ((from | in | of)? MOD)* (nxpg |
    cma nxpg cma? | \( PRE nxpg POST \)) POST;
ntrrepr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=trrepr prot?
    cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST \))? POST
    | PRE h=trrepr ((from | in | of)? MOD)* (nxpg | cma nxpg
    cma? | \( PRE nxpg POST \)) POST;
ntrregr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=trregr prot?
    cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST \))?
    POST
    | PRE h=trregr ((from | in | of)? MOD)* (nxpg |
    cma nxpg cma? | \( PRE nxpg POST \)) POST;

#
# Activator/repressor/regulator agents
#
nactr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=actr prot?
    cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST \))? POST
    | PRE h=actr ((from | in | of)? MOD)* (nxpg | cma nxpg
```

```

        cma? | \( PRE nxpg POST\) POST;
nxrepr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=repr prot?
        cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST\))? POST
        | PRE h=repr ((from | in | of)? MOD)* (nxpg | cma nxpg
        cma? | \( PRE nxpg POST\) POST;
nxregr_ag -> PRE NXPG (POST (bez | ber | codv) PRE)? h=regr prot?
        cmpx? (nxpg | cma nxpg cma? | \( PRE nxpg POST\))? POST
        | PRE h=regr ((from | in | of)? MOD)* (nxpg | cma nxpg
        cma? | \( PRE nxpg POST\) POST;

```

## B.9 Level 9

```

#####
#
:Level9
#
# Level identifies parts of biological entities
#
#####
#
# Variables declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | np | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPROT = (nxprot | nxtf | nxgeneprod | nxactr_ag | nxrepr_ag |
        nxregr_ag | nxtractr_ag | nxtrrepr_ag | nxtrregr_ag);

#
# Upstream activating sequences
#
nxuas_ag -> PRE h=uasx (\( uasx \))? in? nxpg POST
        | PRE h=uasx (\( uasx \))? (from | in | of)?
        (nxgene | nxprom) POST;

```

## The regular expression grammar

```
nxuas_pt -> PRE (nxpg | NXPROT) bing? h=uasx (\( uasx \))? POST
          | PRE h=uasx (\( uasx \))? (for | of) (nxpg | NXPROT) POST
          | PRE (nxactr_ag | ntractr_ag) bing? h=(bs | bingsequ) POST
          | PRE h=(bs | bing sequ) (for | of) (nxactr_ag |
            ntractr_ag) POST;

#
# Upstream repressing sequences
#
nxurs_ag -> PRE h=ursx (\( urx \))? in? nxpg POST
          | PRE h=ursx (\( ursx \))? (from | in | of)?
            (nxgene | nxprom) POST;

nxurs_pt -> PRE (nxpg | NXPROT) bing? h=ursx (\( ursx \))? POST
          | PRE h=ursx (\( ursx \))? (for | of) (nxpg | NXPROT) POST
          | PRE (nxrepr_ag | nxtrrepr_ag | nxtrregr_ag) bing?
            h=(bs | bingsequ) POST
          | PRE h=(bs | bing sequ) (for | of) (nxrepr_ag |
            nxtrrepr_ag | nxtrregr_ag) POST;

#
# Binding sites
#
nxbs -> PRE h=(bs | bing sequ) of nxpg POST;

nxbs_ag -> PRE h=(bs | bing sequ) in? (nxpg | nxgene) POST
          | PRE h=(bs | bing sequ) of nxgene POST;

nxbs_pt -> PRE nxpg h=(bs | bing sequ) POST
          | PRE h=(bs | bing sequ) for (nxpg | nxgeneprod | NXPROT) POST
          | PRE h=(bs | bing sequ) of (nxgeneprod | NXPROT) POST;
```

## B.10 Level 10

```
#####
#
:Level10
#
# Level identifies binding proteins and complexes
#
#####
#
```

```

# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym );
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPROT = (nxpg | nxprot | nxgeneprod | nxactr_ag | nxrepr_ag |
          nxregr_ag | nxtractr_ag | nxtrrepr_ag | nxtrregr_ag);
NXPG = (nxpg | nxgene | nxrna | NXPROT | nxcmpx);

#
# Event: "A binding protein/complex B"
#
nxev_bing_prot -> PRE NXPG bing NXPROT POST;
nxev_bing_cmpx -> PRE NXPG bing prot? nxcmpx POST;

#
# Binding protein/complex
#
nxbingprot -> PRE nxpg bing prot POST;
nxbingcmpx -> PRE nxpg bing (prot | tf)? cmpx POST;

#
# Complexes are now allowed to be "binding complexes"
#
nxcmpxx -> PRE (dna | prot | rna | uasx | ursx)* bing nxcmpx POST;
nxcmpx -> nxcmpxx;

#
# Proteins are now allowed to be "binding proteins"
#
nxprotx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
          bing nxprot POST;
nxprot -> nxprotx;

#
# Activators are now allowed to be "binding activators"
#
nxactr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*

```

## *The regular expression grammar*

```
        bing nxactr_ag POST;
nxactr_ag -> nxactr_agx;

#
# Repressors are now allowed to be "binding repressors"
#
nxrepr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
        bing nxrepr_ag POST;
nxrepr_ag -> nxrepr_agx;

#
# Repressors are now allowed to be "binding repressors"
#
nxregr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
        bing nxregr_ag POST;
nxregr_ag -> nxregr_agx;

#
# Transcription activators are now allowed to be "binding
# transcription activators"
#
nxtractr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
        bing nxtractr_ag POST;
nxtractr_ag -> nxtractr_agx;

#
# Transcription repressors are now allowed to be "binding
# transcription repressors"
#
nxtrrepr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
        bing nxtrrepr_ag POST;
nxtrrepr_ag -> nxtrrepr_agx;

#
# Transcription regulators are now allowed to be "binding
# transcription regulators"
#
nxtrregr_agx -> PRE (dna | prot | enzyme | rna | uasx | ursx)*
        bing nxtrregr_ag POST;
nxtrregr_ag -> nxtrregr_agx;
```

## B.11 Level 11

```
#####
#
# :Level11
#
# Level joins adjacent entities
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such?
        DET? MODX)*;

nxgenex -> PRE nxgene cc nxgene POST
          | PRE nxgene (cma nxgene)+ cma? cc nxgene POST;
nxgene -> nxgenex;

nxrnax -> PRE nxrna cc nxrna POST
          | PRE nxrna (cma nxrna)+ cma? cc nxrna POST;
nxrna -> nxrnax;

nxprotx -> PRE nxprot cc nxprot POST
          | PRE nxprot (cma nxprot)+ cma? cc nxprot POST;
nxprot -> nxprotx;

nxgeneprodx -> PRE nxgeneprod cc nxgeneprod POST
              | PRE nxgeneprod (cma nxgeneprod)+ cma? cc
                nxgeneprod POST;
nxgeneprod -> nxgeneprodx;

nxcmpxx -> PRE nxcmpx cc nxcmpx POST
          | PRE nxcmpx (cma nxcmpx)+ cma? cc nxcmpx POST;
nxcmpx -> nxcmpxx;
```

## The regular expression grammar

```
nxpromx -> PRE nxprom cc nxprom POST
          | PRE nxprom (cma nxprom)+ cma? cc nxprom POST;
nxprom  -> nxpromx;

nxuas_agx -> PRE nxuas_ag cc nxuas_ag POST
            | PRE nxuas_ag (cma nxuas_ag)+ cma? cc nxuas_ag POST;
nxuas_ag -> nxuas_agx;

nxuas_ptx -> PRE nxuas_pt cc nxuas_pt POST
            | PRE nxuas_pt (cma nxuas_pt)+ cma? cc nxuas_pt POST;
nxuas_pt  -> nxuas_ptx;

nxurs_agx -> PRE nxurs_ag cc nxurs_ag POST
            | PRE nxurs_ag (cma nxurs_ag)+ cma? cc nxurs_ag POST;
nxurs_ag  -> nxurs_agx;

nxurs_ptx -> PRE nxurs_pt cc nxurs_pt POST
            | PRE nxurs_pt (cma nxurs_pt)+ cma? cc nxurs_pt POST;
nxurs_pt  -> nxurs_ptx;

nxbs_agx -> PRE nxbs_ag cc nxbs_ag POST
            | PRE nxbs_ag (cma nxbs_ag)+ cma? cc nxbs_ag POST;
nxbs_ag  -> nxbs_agx;

nxbs_ptx -> PRE nxbs_pt cc nxbs_pt POST
            | PRE nxbs_pt (cma nxbs_ag)+ cma? cc nxbs_pt POST;
nxbs_pt  -> nxbs_ptx;
```

## B.12 Level 12

```
#####
#
:Level12
#
# Level joins adjacent entities
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
```

```

NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPG = (nxpg | nxgene | nxrna | nxprot | nxtf | nxactr_ag |
        nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
        nxtrregr_ag | nxgeneprod | nxcmpx);
nxpgx -> PRE NXPG cc ADV NXPG POST
      | PRE NXPG (cma NXPG)+ cma? cc NXPG POST;
nxpg -> nxpgx;

```

## B.13 Level 13

```

#####
#
:Level13
#
# Level identifies activities of biological entities
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
ALT = aa | aasequ | actdom | actr | bddom | bs | dnabddom | enzyme |
      expr | fragment | gene | kindom | org | orf | pre | prokin |
      prod | prods | prot | rna | rnaspec | regdom | repdom |
      transc | trans | tract | trrep | urs | uas;
MOD = (ALT | org | ADJ | PTC | vvg | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;

```

## The regular expression grammar

```
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPG = (nxpg | nxgene | nxrna | nxprot | nxf | nxgeneprod |
        nxcpx | nxactr_ag | nxrepr_ag | nxregr_ag | nxtractr_ag |
        nxrepr_ag | nxtrregr_ag);

#
# nxfunct
#         func = function
#
nxfunct -> PRE NXPG func POST
          | PRE func of NXPG POST;

#
# nxpromact singular nxpg promoter activities
#         promact = promoter activity
#
nxpromact -> PRE nxprom h=(acty|actys) POST
             | PRE prom h=(acty|actys) of? (nxpg | nxgene) POST;
```

## B.14 Level 14

```
#####
#
:Level14
#
# The level nxNE2 includes rules detecting a second set named entities
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
MODprot = aa;
MODrna = pre;
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
ALT = aa | aasequ | actdom | actr | bddom | bs | dnabddom |
      enzyme | expr | fragment | gene | kindom | org | orf |
      pre | prom | prokin | prod | prods | prot | rna | rnaspec |
      regdom | repdom | transc | trans | tract | trrep | urs | uas;
```

```

MOD = (ALT | org | MODprot | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPROT = nxprot | nxf | nxactr_ag | nxrepr_ag | nxregr_ag |
        nxtractr_ag | nxtrrepr_ag | nxtrregr_ag | nxgeneprod;

#
# Transcription activator/repressor/regulator patients
#
nxtractr_pt -> PRE h=tractr of nxpg POST
              | PRE h=(actr | tractr) of (nxgene | nxprom |
              nxpromact) POST;
nxtrrepr_pt -> PRE h=trrepr of nxpg POST
              | PRE h=(repr | trrepr) of (nxgene | nxprom |
              nxpromact) POST;
nxtrregr_pt -> PRE h=trregr of nxpg POST
              | PRE h=(regr | trregr) of (nxgene | nxprom |
              nxpromact) POST;

#
# Activator/repressor/regulator patients
#
nxactr_pt -> PRE h=actr of (nxpg | NXPROT) POST;
nxrepr_pt -> PRE h=repr of (nxpg | NXPROT) POST;
nxregr_pt -> PRE h=regr of (nxpg | NXPROT) POST;

```

## B.15 Level 15

```

#####
#
# :Level15
#
# Level identifies expression of entities
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
MODprot = aa;
MODrna = pre;
ADVHD = rb | cdql | then | well;

```

## The regular expression grammar

```
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
ALT = aa | aasequ | actdom | actr | bddom | bs | dnabddom |
      enzyme | expr | fragment | gene | kindom | org | orf |
      pre | prom | prokin | prod | prods | prot | rna |
      rnaspec | regdom | repdom | transc | trans | tract |
      trrep | urs | uas;
MOD = (ALT | org | MODprot | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPG = (nxpg | nxgene | nxrna | nxprot | nxtf | nxgeneprod |
        nxcmpx | nxactr_ag | nxrepr_ag | nxregr_ag |
        nxtractr_ag | nxtrrepr_ag | nxtrregr_ag);

#
# Event: "R expression X"
#
nxev_expr -> PRE nxpg h=(expr | transc | transl) of (NXPG | nxprom) POST;

#
# Event: "R dependent expression of X"
#
nxev_dep_expr -> PRE nxpg_dep h=(expr | transc | transl) of
                (NXPG | nxprom) POST;

#
# Expression/transcription/translation of nxpg.
#
nxexpr -> PRE (NXPG | nxprom) h=(expr | transc | transl) POST
        | PRE h=(expr | transc | transl) (from | of) (NXPG | nxprom) POST;
```

## B.16 Level 16

```
#####
#
:Level16
#
# Level identifies "entity in entity" and "entity of entity"
```

```

#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
ALT = aa | aasequ | actdom | actr | bddom | bs | dnabddom |
      enzyme | expr | fragment | gene | kindom | org | orf |
      pre | prokin | prod | prods | prot | rna | rnaspec |
      regdom | repdom | transc | trans | tract | trrep | urs | uas;
MOD = (ALT | org | ADJ | PTC | vvg | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPG = (nxpg | nxgene | nxrna | nxprot | nxtf | nxgeneprod |
        nxcmpx | nxactr_ag | nxrepr_ag | nxregr_ag | nxtractr_ag |
        nxtrrepr_ag | nxtrregr_ag);

#
# UAS/URS/binding site in gene or promoter
#
nxev_uas_in -> PRE nxuas_pt in (nxgene | nxprom) POST;
nxev_urs_in -> PRE nxurs_pt in (nxgene | nxprom) POST;
nxev_bs_in -> PRE nxbs_pt in (nxgene | nxprom) POST;

#
# Activator/repressor/regulator of
#
nxev_actr_of_expr -> PRE nxactr_ag of (nxgene | nxprom |
        nxpromact | nxexpr) POST
        | PRE nxtractr_ag of (nxpg | nxgene | nxprom |
        nxexpr) POST;
nxev_repr_of_expr -> PRE nxrepr_ag of (nxgene | nxprom |
        nxpromact | nxexpr) POST
        | PRE nxtrrepr_ag of (nxpg | nxgene |
        nxprom | nxexpr) POST;
nxev_regr_of_expr -> PRE nxregr_ag of (nxgene | nxprom |
        nxpromact | nxexpr) POST
        | PRE (nxtf | nxtrregr_ag) of (nxpg | nxgene |
        nxprom | nxexpr) POST;

```

## B.17 Level 17

```
#####  
#  
:Level17  
#  
# The level nxdomains includes rules detecting mutations  
#  
#####  
#  
# Variable declaration  
#  
#  
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);  
ADVHD = rb | cdql | then | well;  
ADV = ADVHD | rbr | more | rbs | ql;  
NUM = cd | cdx;  
JX = ADV? (jj | jjr | jjs);  
JXC = JX (cma JX)* (cc | cma) JX;  
ADJ = JX | JXC | mx;  
PTC = ADV* (vbn | vbg);  
MOD = (org | JXC | ADJ | PTC | vvn | cd | sym);  
MODX = (MOD | (nn | nns) (dependent | independent | responsive?));  
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;  
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;  
NXPG = (nxpg | nxgene | nxprot | nxf | nxactr_ag | nxrepr_ag |  
        nxregr_ag | nxtractr_ag | nxtrrepr_ag | nxtrregr_ag | nxgeneprod);  
  
#  
# Deletion  
#  
nxdel -> PRE (NXPG | nxuas_pt | nxurs_pt | nxbs_pt) h=(abse |  
            degr | deletion | disr)  
        | PRE (abse | degr | deletion | disr) of (NXPG |  
            nxuas_pt | nxurs_pt | nxbs_pt)  
        | PRE deleting (NXPG | nxuas_pt | nxurs_pt | nxbs_pt);  
  
#  
# Mutation  
#     mut = (defect | mutant)  
#  
nxmut -> PRE (NXPG | nxuas_pt | nxurs_pt | nxbs_pt) repr?
```

```

        h=(mut | mution) POST
    | PRE repr? h=(mut | mutd | mution) forms? (for | in | of | cma)?
      (NXPG | nxuas_pt | nxurs_pt | nxbs_pt) POST;

#
# Overproduction
#
nxoprđ -> PRE oprđ of NXPG
        | PRE NXPG oprđ;

```

## B.18 Level 18

```

#####
#
# :Level18
#
# The level nxgarbage collects garbage and is not processed afterwards.
#
#####
#
# Variable declaration
#

ENTITIES = nxbingprot | nxbs_ag | nxbs_pt | nxdom | nxexpr |
           nxfunct | nxgarbage | nxgene | nxgeneprod | nxmut |
           nxoprđ | nxorf | nxpg | nxprom | nxpromact | nxrna |
           nxprot | nxf | nxactr_ag | nxrepr_ag | nxregr_ag |
           nxtractr_ag | nxtrrepr_ag | nxtrregr_ag | nxrepr_pt |
           nxuas_ag | nxuas_pt | nxurs_ag | nxurs_pt;
PROPER = person | name | ci-st | doll | np;
COMMON = nn | nns | month | unit | units | tunit | tunits ;
N = COMMON | PROPER;
NUM = cd | cdx;

#
# nxbrack brackets containing noun chunks with no special noun
# from semlex
#
nxbrack -> \( vvn? ENTITIES \)
          | \( vvn? such? DET? NUM? (ADJ | PTC)* (ADJ | N)*
            h=(COMMON) cd? (ADJ | N)* \)
          | \( vvn? DET? NUM? (ADJ | PTC)* h=PROPER \)
          | \( vvn? DET h=(jjr | jjs | such) \)

```

```
| \( vvn? cdq1? h=ntp-q \  
| \( vvn? NUM to NUM nn \  
| \( vvn? h=( prp | cd | dtp | cd | dtp | qq |  
ex | name | person | doll | ci-st | rbr | rbs ) \);
```

## B.19 Level 19

```
#####  
#  
:Level19  
#  
# The level nxgarbage collects garbage and is not processed afterwards.  
#  
#####  
  
nxactr_agx -> nxactr_ag cma? nxbrack cma?;  
nxactr_ptx -> nxactr_pt cma? nxbrack cma?;  
nxbingprotx -> nxbingprot cma? nxbrack cma?;  
nxbs_agx -> nxbs_ag cma? nxbrack cma?;  
nxbs_ptx -> nxbs_pt cma? nxbrack cma?;  
nxdomx -> nxdom cma? nxbrack cma?;  
nxexprx -> nxexpr cma? nxbrack cma?;  
nxfunctx -> nxfunct cma? nxbrack cma?;  
nxgarbagex -> nxgarbage cma? nxbrack cma?;  
nxgenex -> nxgene cma? nxbrack cma?;  
nxgeneprodx -> nxgeneprod cma? nxbrack cma?;  
nxdelx -> nxdel cma? nxbrack cma?;  
nxmutx -> nxmut cma? nxbrack cma?;  
nxoprxdx -> nxoprd cma? nxbrack cma?;  
nxorfx -> nxorf cma? nxbrack cma?;  
nxpgx -> nxpg cma? nxbrack cma?;  
nxpromx -> nxprom cma? nxbrack cma?;  
nxpromactx -> nxpromact cma? nxbrack cma?;  
nxprotx -> nxprot cma? nxbrack cma?;  
nxrnax -> nxrna cma? nxbrack cma?;  
nxtractr_agx -> nxtractr_ag cma? nxbrack cma?;  
nxtrrepr_agx -> nxtrrepr_ag cma? nxbrack cma?;  
nxtrregr_agx -> nxtrregr_ag cma? nxbrack cma?;  
nxreprx_ag -> nxrepr_ag cma? nxbrack cma?;  
nxreprx_pt -> nxrepr_pt cma? nxbrack cma?;  
nxregrx_ag -> nxregr_ag cma? nxbrack cma?;  
nxregrx_pt -> nxregr_pt cma? nxbrack cma?;
```

```

nxuas_agx -> nxuas_ag cma? nxbrack cma?;
nxuas_ptx -> nxuas_pt cma? nxbrack cma?;
nxurs_agx -> nxurs_ag cma? nxbrack cma?;
nxurs_ptx -> nxurs_pt cma? nxbrack cma?;

nxactr_ag -> nxactr_agx;
nxactr_pt -> nxactr_ptx;
nxbingprot -> nxbingprotx;
nxbs_ag -> nxbs_agx;
nxbs_pt -> nxbs_ptx;
nxdom -> nxdomx;
nxexpr -> nxexprx;
nxfunct -> nxfunctx;
nxgarbage -> nxgarbagex;
nxgene -> nxgenex;
nxgeneprod -> nxgeneprodx;
nxdel -> nxdelx;
nxmut -> nxmutx;
nxopr -> nxoprxd;
nxorf -> nxorfx;
nxpg -> nxpgx;
nxprom -> nxpromx;
nxpromact -> nxpromactx;
nxprot -> nxprotx;
nxrna -> nxrnax;
nxtractr_ag -> nxtractr_agx;
nxtrrepr_ag -> nxtrrepr_agx;
nxtrregr_ag -> nxtrregr_agx;
nxrepr_ag -> nxrepr_agx;
nxrepr_pt -> nxrepr_ptx;
nxregr_ag -> nxregr_agx;
nxregr_pt -> nxregr_ptx;
nxuas_ag -> nxuas_agx;
nxuas_pt -> nxuas_ptx;
nxurs_ag -> nxurs_agx;
nxurs_pt -> nxurs_ptx;

```

## B.20 Level 20

```

#####
#
:Level20
#

```

## The regular expression grammar

```
# Thex level nxgarbage collects garbage and is not processed afterwards.
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | JXC | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
ENTITIES = nxbingprot | nxbs_ag | nxbs_pt | nxdom | nxexpr |
           nxfunct | nxgarbage | nxgene | nxgeneprod | nxmut |
           nxopr | nxorf | nxpg | nxprom | nxpromact | nxrna |
           nxprot | nxf | nxactr_ag | nxrepr_ag | nxregr_ag |
           nxtractr_ag | nxtrrepr_ag | nxtrregr_ag | nxrepr_pt |
           nxuas_ag | nxuas_pt | nxurs_ag | nxurs_pt;

#
#
#
nxgarbage -> PRE ENTITIES (plasm | vect | transp)
           | PRE (plasm | vect) ENTITIES
           | PRE transp of ENTITIES;
```

## B.21 Level 21

```
#####
#
:Level21
#
# The level nxevent includes all nominalisations describing regulatory events
#
#####
#
# Variable declaration
```

```

#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | JXC | NUM | ADJ | PTC | vvn | cd | sym);
MODX = (MOD | (nn | nns) (dependent | independent | responsive)?);
PRE = (NUM ((cc | of | to) NUM)?)? such? DET? (MODX cma?)*;
POST = ((from | in | of)? (NUM ((cc | of | to) NUM)?)? such? DET? MODX)*;
NXPROT = nxprot | nxf | nxactr_ag | nxrepr_ag | nxregr_ag |
        nxtractr_ag | nxtrrepr_ag | nxtrregr_ag | nxgeneprod;
ACTIVATION = activ | activation | ampn | oexpr | oexpring;
REPRESSION = repression | repring | inh;
REGULATION = regul | expr | opdr | trans | transl;
ENTITIES = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
           nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
           nxtrregr_ag | nxgeneprod | nxorf | nxcmpx | nxdom |
           nxexpr | nxfunct | nxpromact;

#
# Binding to promoter.
#
nxev_bind_uas -> PRE (bing of (nxpg | NXPROT | nxcmpx) |
                    (nxpg | NXPROT | nxcmpx) bing) to nxuas_ag;
nxev_bind_urs -> PRE (bing of (nxpg | NXPROT | nxcmpx) |
                    (nxpg | NXPROT | nxcmpx) bing) to nxurs_ag;
nxev_bind_prom -> PRE (bing of (nxpg | NXPROT | nxcmpx) |
                     (nxpg | NXPROT | nxcmpx) bing) to (nxbs |
                     nxbs_ag | nxprom);

#
# Activation/repression/regulation of entities by entities
#
nxev_act_expr -> PRE ACTIVATION of (nxgene | nxprom |
                                   nxpromact | nxexpr) by ENTITIES
                | PRE (nxgene | nxprom | nxpromact |
                       nxexpr) ACTIVATION by ENTITIES
                | PRE (nxpg | nxgene | nxprom | nxpromact |
                       nxexpr) (ACTIVATION | REGULATION) by nxtract_ag;
nxev_rep_expr -> PRE REPRESSION of (nxgene | nxprom |
                                   nxpromact | nxexpr) by ENTITIES
                | PRE (nxgene | nxprom | nxpromact | nxexpr)
                   REPRESSION by ENTITIES

```

## The regular expression grammar

```
        | PRE (nxpg | nxgene | nxprom | nxpromact |
          nxexpr) (REPRESSION | REGULATION) by nxtrrep_ag;
nxev_reg_expr -> PRE REGULATION of (nxgene | nxprom |
          nxpromact | nxexpr) by ENTITIES
        | PRE (nxgene | nxprom | nxpromact | nxexpr)
          REGULATION by ENTITIES
        | PRE (nxpg | nxgene | nxprom | nxpromact | nxexpr)
          REGULATION by (nxtf | nxtrreg_ag);

#
# Activity/inactivity
#
nxactiv -> PRE ACTIVATION of ENTITIES
        | PRE ENTITIES ACTIVATION
        | ENTITIES vx oexprv
        | actv forms of ENTITIES;
nxinactiv -> PRE REPRESSION of ENTITIES
        | PRE ENTITIES REPRESSION;

#
# Ability/inability
#
nxabil -> PRE ENTITIES abil
        | PRE abil of ENTITIES;
nxinabil -> PRE ENTITIES inabil
        | PRE inabil of ENTITIES;
```

## B.22 Level 22

```
#####
#
:Level22
#
# The level nxRest does some non-semantic chunking and builds verbal complexes
#
#####
#
# Variable declaration
#
DET = dt | dtp | prps | (cdql | cdqlx)? (dt-a | dt-q | dtp-q);
ADVHD = rb | cdql | then | well;
ADV = ADVHD | rbr | more | rbs | ql;
NUM = cd | cdx;
```

```

JX = ADV? (jj | jjr | jjs);
JXC = JX (cma JX)* (cc | cma) JX;
ADJ = JX | JXC | mx;
PTC = ADV* (vbn | vbg);
MOD = (org | JXC | NUM | ADJ | PTC | vvn | cd | sym);
ALT = (aa | aasequ | actdom | actr | bddom | bs | dnabddom |
       enzyme | expr | fragment | gene | kindom | org | orf | prod
       | prods | prom | prokin | prot | repr | rna | rnaspec |
       regdom | repdom | transc | trans | tract | trrep | urs |
       uas);
PROPER = person | name | ci-st | doll | np;
COMMON = nn | nns | month | unit | units | tunit | tunits ;
N = ALT | COMMON | PROPER;

VADVP = ADV* (ADVHD | only);
MX = mx | units | tunits;

#
# Noun Chunk
#
nxrest -> such? DET? NUM? (ADJ | PTC)* (ADJ | N)* h=(COMMON | ALT)
        cd? (ADJ | N)*
        | DET? NUM? (ADJ | PTC)* h=PROPER
        | DET h=(jjr | jjs | such)
        | cdql? h=ntp-q
        | h=( prp | cd | dtp | cd | dtp | qq | ex
        # prp=Personal Pronoun
        | name | person | doll | ci-st | rbr | rbs);

#
#
#
perx -> ( (cd | cdx | dt-a) (COMMON | doll) | MX ) (dt-a | per) COMMON;

such-as -> such as;

rx -> ADV+ ADV
    | by then
    | MX ago;

ax -> MX? (ADV | rx)* h=(jj | jjr | jjs);

vgx -> VADVP? h=vvg;
vnx -> VADVP? h=vbn;

VB-TNS = vb | vbp | vbz | vbd;

```

## The regular expression grammar

```
DO-TNS = do | doz | dod;
HV-TNS = hv | hvz | hvd;
BE-TNS = be | bem | bez | bedz | bed | ber | bedr;
MODAL = md | doz | do | dod;

VP-PASS = VADVP? h=(vbn | vbd | hvn | ben);
VP-PROG = VADVP? ( h=(vbg | hvg | beg)
                  | f=begin VP-PASS
                  | beg h=ax );
VP-PERF = VADVP? ( h=(vbn | hvn | ben)
                  | f=begin (VP-PROG | VP-PASS)
                  | ben h=ax );
VP-INF = VADVP? ( h=(vb | do | hv | be)
                  | f=hv VP-PERF
                  | f=be (VP-PROG | VP-PASS)
                  | be h=ax );
vx -> VADVP? ( h=(md | DO-TNS | HV-TNS | BE-TNS | VB-TNS)
              | md VP-INF
              | DO-TNS VP-INF
              | f=HV-TNS VP-PERF
              | f=BE-TNS (VP-PROG | VP-PASS)
              | BE-TNS h=ax );
inf -> VADVP? to VADVP? h=VP-INF;
infneg -> VADVP? to VADVP? neg VADVP? h=VP-INF
        | VADVP? neg to VADVP? VADVP? h=VP-INF
        | failed to VADVP? h=VP-INF;
```

## B.23 Level 23

```
#####
#
#Level23
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
AUX    = vx | to;
MODV   = ax | rb;
RESTV  = vvz | vvn | vvd | vv;
```

```

VCCAUS = AUX* MODV* cauv;
VCCAUSTO = AUX* (MODV | RESTV)* (inf | to) VCCAUS;

VCACT = AUX* MODV* actv;
VCACTTO = AUX* (MODV | RESTV)* (inf | to) VCACT;

VCREP = AUX* MODV* repv;
VCREPTO = AUX* (MODV | RESTV)* (inf | to) VCREP;

VCREG = AUX* MODV* (regv | dirv);
VCREGTO = AUX* (MODV | RESTV)* (inf | to) VCREG;

#
# Ability to activate/repress/regulate expression.
#
nxev_abil_act_expr -> nxabil (cma? wdt)? (VCCAUSTO | VCACTTO)
                    nxbrack? (nxgene | nxprom | nxpromact | nxexpr);
nxev_abil_rep_expr -> nxabil (cma? wdt)? VCREPTO nxbrack?
                    (nxgene | nxprom | nxpromact | nxexpr);
nxev_abil_reg_expr -> nxabil (cma? wdt)? VCREGTO nxbrack?
                    (nxgene | nxprom | nxpromact | nxexpr);

```

## B.24 Level 24

```

#####
#
:Level24
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
AUX      = vx | to;
MODV     = ax | rb;
RESTV    = vvz | vvn | vvd | vv;

VCCAUS = AUX* MODV* cauv;
VCCAUSNEG = AUX* MODV* neg MODV* cauv;
VCCAUSNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCCAUS;
VCCAUSTO = AUX* (MODV | RESTV)* (inf | to) VCCAUS;

```

## The regular expression grammar

```
VCMED = AUX* MODV* mediv;
VCMEDNEG = AUX* MODV* neg MODV* mediv;

VCACT = AUX* MODV* actv;
VCACTNEG = AUX* MODV* neg MODV* actv;
VCACTNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCACT;
VCACTTO = AUX* (MODV | RESTV)* (inf | to) VCACT;

VCREP = AUX* MODV* repv;
VCREPNEG = AUX* MODV* neg MODV* repv;
VCREPNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCREP;
VCREPTO = AUX* (MODV | RESTV)* (inf | to) VCREP;

VCREG = AUX* MODV* (regv | dirv);
VCREGNEG = AUX* MODV* neg MODV* (regv | dirv);
VCREGNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCREG;
VCREGTO = AUX* (MODV | RESTV)* (inf | to) VCREG;

NXnnpgx = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
          nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
          nxtrregr_ag | nxgeneprod | nxorf | nxprom | nxuas_ag |
          nxurs_ag | nxbs | nxbs_ag | nxcmpx | nxpromact |
          nxexpr;

#
# Transcriptional activator activates or regulates something.
#
ev_tractr_reg_va -> nxtractr_ag (cma? wdt)? (VCACT | VCACTTO |
          VCREG | VCREGTO) nxbrack? (NXnnpgx | nxactiv)
          | nxtractr_ag (cma? wdt)? VCCAUS nxbrack?
          (NXnnpgx | nxactiv) (VCACTTO | VCREGTO);
ev_tractr_reg_vp -> (NXnnpgx) (cma? wdt)? (VCCAUS (VCACTTO |
          VCREGTO)? | VCACT | VCREG) by nxbrack? nxtractr_ag
          | (NXnnpgx | nxactiv) (VCACTTO | VCREGTO)
          nxbrack? nxtractr_ag;
ev_tractr_reg_va_n -> nxtractr_ag (cma? wdt)? (VCACTNEG |
          VCACTNEGTO | VCREGNEG | VCREGNEGTO) nxbrack?
          (NXnnpgx | nxactiv)
          | nxtractr_ag (cma? wdt)? VCCAUSNEG nxbrack?
          (NXnnpgx | nxactiv) (VCACTTO | VCREGTO)
          | nxtractr_ag (cma? wdt)? VCCAUS nxbrack?
          (NXnnpgx | nxactiv) (VCACTNEGTO | VCREGNEGTO);
ev_tractr_reg_vp_n -> (NXnnpgx | nxactiv) (cma? wdt)? (VCCAUSNEG
          (VCACTTO | VCREGTO)? | VCCAUS (VCACTNEGTO |
          VCREGNEGTO) | VCACTNEG | VCREGNEG) by
          nxbrack? nxtractr_ag | (NXnnpgx | nxactiv)
          (VCACTNEGTO | VCREGNEGTO) nxbrack? nxtractr_ag;
```

```

#
# Transcriptional repressor represses or regulates something.
#
ev_trrepr_reg_va -> nxtrrepr_ag (cma? wdt)? (VCREP | VCREPTO |
    VCREG | VCREGTO) nxbrack? NXnnpngx
    | nxtrrepr_ag (cma? wdt)? VCCAUS nxbrack?
    NXnnpngx (VCREPTO | VCREGTO);
ev_trrepr_reg_vp -> NXnnpngx (cma? wdt)? (VCCAUS (VCREPTO |
    VCREGTO)? | VCREP | VCREG) by nxbrack? nxtrrepr_ag
    | NXnnpngx (VCREPTO | VCREGTO) nxbrack? nxtrrepr_ag;
ev_trrepr_reg_va_n -> nxtrrepr_ag (cma? wdt)? (VCREPNEG |
    VCREPNEGTO | VCREGNEG | VCREGNEGTO) nxbrack? NXnnpngx
    | nxtrrepr_ag (cma? wdt)? VCCAUSNEG nxbrack?
    NXnnpngx (VCREPTO | VCREGTO)
    | nxtrrepr_ag (cma? wdt)? VCCAUS nxbrack?
    NXnnpngx (VCREPNEGTO | VCREGNEGTO);
ev_trrepr_reg_vp_n -> NXnnpngx (cma? wdt)? (VCCAUSNEG (VCREPTO |
    VCREGTO)? | VCCAUS (VCREPNEGTO | VCREGNEGTO) |
    VCREPNEG | VCREGNEG) by nxbrack? nxtrrepr_ag
    | NXnnpngx (VCREPNEGTO | VCREGNEGTO) nxbrack?
    nxtrrepr_ag;

#
# Transcriptional regulator activates something.
#
ev_trrepr_act_va -> (nxtrrepr_ag | nxtf) (cma? wdt)? (VCCAUS |
    VCCAUSTO | VCACT | VCACTTO) nxbrack?
    (NXnnpngx | nxactiv)
    | (nxtrrepr_ag | nxtf) (cma? wdt)? VCCAUS nxbrack?
    (NXnnpngx | nxactiv) VCACTTO;
ev_trrepr_act_vp -> (NXnnpngx | nxactiv) (cma? wdt)? (VCMED | VCCAUS
    VCACTTO? | VCACT) by nxbrack? (nxtrrepr_ag | nxtf)
    | (NXnnpngx | nxactiv) VCACTTO nxbrack?
    (nxtrrepr_ag | nxtf);
ev_trrepr_act_va_n -> (nxtrrepr_ag | nxtf) (cma? wdt)? (VCCAUSNEG |
    VCCAUSNEGTO | VCACTNEG | VCACTNEGTO) nxbrack?
    (NXnnpngx | nxactiv)
    | (nxtrrepr_ag | nxtf) (cma? wdt)? VCCAUSNEG
    nxbrack? (NXnnpngx | nxactiv) VCACTTO
    | (nxtrrepr_ag | nxtf) (cma? wdt)? VCCAUS
    nxbrack? (NXnnpngx | nxactiv) VCACTNEGTO;
ev_trrepr_act_vp_n -> (NXnnpngx | nxactiv) (cma? wdt)? (VCMEDNEG |
    VCCAUSNEG VCACTTO? | VCCAUS VCACTNEGTO |
    VCACTNEG) by nxbrack? (nxtrrepr_ag | nxtf)
    | (NXnnpngx | nxactiv) VCACTNEGTO nxbrack?
    (nxtrrepr_ag | nxtf);

```

## The regular expression grammar

```
#
# Transcriptional regulator represses something.
#
ev_trregr_rep_va -> (nxtregr_ag | nxf) (cma? wdt)?
                    (VCREP | VCREPTO) nxbrack? NXnnpqx
                    | (nxtregr_ag | nxf) (cma? wdt)?
                    VCCAUS nxbrack? NXnnpqx VCREPTO;
ev_trregr_rep_vp -> NXnnpqx (cma? wdt)? (VCCAUS VCREPTO? |
                    VCREP) by nxbrack? (nxtregr_ag | nxf)
                    | NXnnpqx VCREPTO nxbrack? (nxtregr_ag | nxf);
ev_trregr_rep_va_n -> (nxtregr_ag | nxf) (cma? wdt)?
                    (VCREPNEG | VCREPNEGTO) nxbrack? NXnnpqx
                    | (nxtregr_ag | nxf) (cma? wdt)?
                    VCCAUSNEG nxbrack? NXnnpqx VCREPTO
                    | (nxtregr_ag | nxf) (cma? wdt)?
                    VCCAUS nxbrack? NXnnpqx VCREPNEGTO;
ev_trregr_rep_vp_n -> NXnnpqx (cma? wdt)? (VCCAUSNEG VCREPTO? |
                    VCCAUS VCREPNEGTO | VCREPNEG) by nxbrack?
                    (nxtregr_ag | nxf)
                    | NXnnpqx VCREPNEGTO nxbrack? (nxtregr_ag | nxf);

#
# Transcriptional regulator regulates something.
#
ev_trregr_reg_va -> (nxtregr_ag | nxf) (cma? wdt)? (VCREG | VCREGTO)
                    nxbrack? (NXnnpqx | nxactiv | nxinactiv)
                    | (nxtregr_ag | nxf) (cma? wdt)? VCCAUS nxbrack?
                    (NXnnpqx | nxactiv | nxinactiv) VCREGTO;
ev_trregr_reg_vp -> (NXnnpqx | nxactiv | nxinactiv) (cma? wdt)?
                    (VCCAUS VCREGTO? | VCREG) by nxbrack?
                    (nxtregr_ag | nxf)
                    | (NXnnpqx | nxactiv | nxinactiv) VCREGTO
                    nxbrack? (nxtregr_ag | nxf);
ev_trregr_reg_va_n -> (nxtregr_ag | nxf) (cma? wdt)?
                    (VCREGNEG | VCREGNEGTO) nxbrack?
                    (NXnnpqx | nxactiv | nxinactiv)
                    | (nxtregr_ag | nxf) (cma? wdt)?
                    VCCAUSNEG nxbrack? (NXnnpqx | nxactiv |
                    nxinactiv) VCREGTO
                    | (nxtregr_ag | nxf) (cma? wdt)? VCCAUS
                    nxbrack? (NXnnpqx | nxactiv | nxinactiv)
                    VCREGNEGTO;
ev_trregr_reg_vp_n -> (NXnnpqx | nxactiv | nxinactiv) (cma? wdt)?
                    (VCCAUSNEG VCREGTO? | VCCAUS VCREGNEGTO |
                    VCREGNEG) by nxbrack? (nxtregr_ag | nxf)
                    | (NXnnpqx | nxactiv | nxinactiv)
```

```
VCREGNEGTO nxbrack? (nxtregr_ag | nxf);
```

## B.25 Level 25

```
#####
#
:Level25
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
    AUX    = vx | to;
    MODV   = ax | rb;
    RESTV  = vvz | vvn | vvd | vv;

    VCCAUS = AUX* MODV* cauv;
    VCCAUSNEG = AUX* MODV* neg MODV* cauv;
    VCCAUSNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCCAUS;
    VCCAUSTO = AUX* (MODV | RESTV)* (inf | to) VCCAUS;

    VCMED  = AUX* MODV* mediv;
    VCMEDNEG = AUX* MODV* neg MODV* mediv;

    VCACT  = AUX* MODV* actv;
    VCACTNEG = AUX* MODV* neg MODV* actv;
    VCACTNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCACT;
    VCACTTO = AUX* (MODV | RESTV)* (inf | to) VCACT;

    VCREP  = AUX* MODV* repv;
    VCREPNEG = AUX* MODV* neg MODV* repv;
    VCREPNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCREP;
    VCREPTO = AUX* (MODV | RESTV)* (inf | to) VCREP;

    VCREG  = AUX* MODV* (regv | dirv);
    VCREGNEG = AUX* MODV* neg MODV* (regv | dirv);
    VCREGNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCREG;
    VCREGTO = AUX* (MODV | RESTV)* (inf | to) VCREG;

    NXnnpgx = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
              nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
```

## The regular expression grammar

```
nxtrregr_ag | nxgeneprod | nxorf | nxuas_pt | nxurs_pt
| nxbs | nxbs_pt | nxcmpx | nxdom | nxexpr | nxfunc |
nxopr | nxactiv;

# ..._l lose the direction
# ..._r reverse the direction

#
# Activation of expression.
#
ev_act_expr_va -> NXnnpngx (cma? wdt)? (VCCAUS | VCCAUSTO | VCACT |
VCACTTO) nxbrack? (nxgene | nxprom | nxexpr)
| NXnnpngx (cma? wdt)? VCCAUS nxbrack? (nxgene |
nxprom | nxexpr) VCACTTO;
ev_act_expr_vp -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCMED | VCCAUS VCACTTO? | VCACT) by nxbrack? NXnnpngx
| (nxgene | nxprom | nxexpr) VCACTTO nxbrack? NXnnpngx;
ev_act_expr_va_l -> nxmut (cma? wdt)? (VCCAUS | VCCAUSTO | VCACT |
VCACTTO) nxbrack? (nxgene | nxprom | nxexpr)
| nxmut (cma? wdt)? VCCAUS nxbrack? (nxgene |
nxprom | nxexpr) VCACTTO;
ev_act_expr_vp_l -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCMED | VCCAUS VCACTTO? | VCACT)
(by | from | in) nxbrack? nxmut
| (nxgene | nxprom | nxexpr) VCACTTO nxbrack? nxmut;
ev_act_expr_va_r -> (nxdel | nxinactiv) (cma? wdt)?
(VCCAUS | VCCAUSTO | VCACT | VCACTTO) nxbrack?
(nxgene | nxprom | nxexpr)
| (nxdel | nxinactiv) (cma? wdt)? VCCAUS
nxbrack? (nxgene | nxprom | nxexpr) VCACTTO;
ev_act_expr_vp_r -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCMED | VCCAUS VCACTTO? | VCACT)
(by | from | in) nxbrack? (nxdel | nxinactiv)
| (nxgene | nxprom | nxexpr) VCACTTO nxbrack?
(nxdel | nxinactiv);
ev_act_expr_va_n -> (NXnnpngx | nxdel | nxmut | nxopr |
nxactiv | nxinactiv) (cma? wdt)? (VCCAUSNEG |
VCCAUSNEGTO | VCACTNEG | VCACTNEGTO) nxbrack?
(nxgene | nxprom | nxexpr)
| (NXnnpngx | nxdel | nxmut | nxopr | nxactiv |
nxinactiv) (cma? wdt)? VCCAUSNEG nxbrack? (nxgene |
nxprom | nxexpr) VCACTTO
| (NXnnpngx | nxdel | nxmut | nxopr | nxactiv |
nxinactiv) (cma? wdt)? VCCAUS nxbrack?
(nxgene | nxprom | nxexpr) VCACTNEGTO;
ev_act_expr_vp_n -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCMEDNEG | VCCAUSNEG VCACTTO? |
```

```

VCCAUS VCACTNEGTO | VCACTNEG) by nxbrack?
(NXnnpngx | nxdel | nxmut | nxinactiv)
| (nxgene | nxprom | nxexpr) VCACTNEGTO
nxbrack? (NXnnpngx | nxdel | nxmut | nxinactiv);

#
# Repression of expression.
#
ev_rep_expr_va -> NXnnpngx (cma? wdt)? (VCREP | VCREPTO) nxbrack?
(nxgene | nxprom | nxexpr)
| NXnnpngx (cma? wdt)? VCCAUS nxbrack?
(nxgene | nxprom | nxexpr) VCREPTO;
ev_rep_expr_vp -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCCAUS VCREPTO? | VCREP) by nxbrack? NXnnpngx
| (nxgene | nxprom | nxexpr) VCREPTO nxbrack? NXnnpngx;
ev_rep_expr_va_l -> nxmut (cma? wdt)? (VCREP | VCREPTO) nxbrack?
(nxgene | nxprom | nxexpr)
| nxmut (cma? wdt)? VCCAUS nxbrack?
(nxgene | nxprom | nxexpr) VCREPTO;
ev_rep_expr_vp_l -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCCAUS VCREPTO? | VCREP) (by | from | in)
nxbrack? nxmut
| (nxgene | nxprom | nxexpr) VCREPTO nxbrack? nxmut;
ev_rep_expr_va_r -> (nxdel | nxinactiv) (cma? wdt)? (VCREP | VCREPTO)
nxbrack? (nxgene | nxprom | nxexpr)
| (nxdel | nxinactiv) (cma? wdt)? VCCAUS nxbrack?
(nxgene | nxprom | nxexpr) VCREPTO;
ev_rep_expr_vp_r -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCCAUS VCREPTO? | VCREP) (by | from | in)
nxbrack? (nxdel | nxinactiv)
| (nxgene | nxprom | nxexpr) VCREPTO nxbrack?
(nxdel | nxinactiv);
ev_rep_expr_va_n -> (NXnnpngx | nxdel | nxmut | nxoprnd |
nxactiv | nxinactiv) (cma? wdt)? (VCREPNEG |
VCREPNEGTO) nxbrack? (nxgene | nxprom | nxexpr)
| (NXnnpngx | nxdel | nxmut | nxoprnd | nxactiv |
nxinactiv) (cma? wdt)? VCCAUSNEG nxbrack?
(nxgene | nxprom | nxexpr) VCREPTO
| (NXnnpngx | nxdel | nxmut | nxoprnd | nxactiv |
nxinactiv) (cma? wdt)? VCCAUS nxbrack?
(nxgene | nxprom | nxexpr) VCREPNEGTO;
ev_rep_expr_vp_n -> (nxgene | nxprom | nxexpr) (cma? wdt)?
(VCCAUSNEG VCREPTO? | VCCAUS VCREPNEGTO |
VCREPNEG) by nxbrack? (NXnnpngx | nxdel | nxmut |
nxinactiv)
| (nxgene | nxprom | nxexpr) VCREPNEGTO
nxbrack? (NXnnpngx | nxdel | nxmut | nxinactiv);

```

## The regular expression grammar

```
#
# Regulation of expression.
#
ev_reg_expr_va -> (NXnnpgx | nxdel | nxmut | nxinactiv)
                  (cma? wdt)? (VCREG | VCREGTO) nxbrack?
                  (nxgene | nxprom | nxexpr)
                  | (NXnnpgx | nxdel | nxmut | nxinactiv)
                  (cma? wdt)? VCCAUS nxbrack?
                  (nxgene | nxprom | nxexpr) VCREGTO;
ev_reg_expr_vp -> (nxgene | nxprom | nxexpr) (cma? wdt)?
                  (VCCAUS VCREGTO? | VCREG) by nxbrack?
                  (NXnnpgx | nxdel | nxmut | nxinactiv)
                  | (nxgene | nxprom | nxexpr) VCREGTO nxbrack?
                  (NXnnpgx | nxdel | nxmut | nxinactiv);
ev_reg_expr_va_n -> (NXnnpgx | nxdel | nxmut | nxinactiv)
                   (cma? wdt)? (VCREGNEG | VCREGNEGTO) nxbrack?
                   (nxgene | nxprom | nxexpr)
                   | (NXnnpgx | nxdel | nxmut | nxinactiv)
                   (cma? wdt)? VCCAUSNEG nxbrack?
                   (nxgene | nxprom | nxexpr) VCREGTO
                   | (NXnnpgx | nxdel | nxmut | nxinactiv)
                   (cma? wdt)? VCCAUS nxbrack?
                   (nxgene | nxprom | nxexpr) VCREGNEGTO;
ev_reg_expr_vp_n -> (nxgene | nxprom | nxexpr) (cma? wdt)?
                   (VCCAUSNEG VCREGTO? | VCCAUS VCREGNEGTO | VCREGNEG)
                   by nxbrack? (NXnnpgx | nxdel | nxmut | nxinactiv)
                   | (nxgene | nxprom | nxexpr) VCREGNEGTO nxbrack?
                   (NXnnpgx | nxdel | nxmut | nxinactiv);
```

## B.26 Level 26

```
#####
#
:Level26
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
    AUX    = vx | to;
```

```

MODV  = ax | rb;
RESTV = vvz | vvn | vvd | vv;
BINDV = bindv | bindvs | bindvd;

VCCAUS = AUX* MODV* cauv;
VCCAUSNEG = AUX* MODV* neg MODV* cauv;
VCCAUSTO = AUX* (MODV | RESTV)* (inf | to) VCCAUS;

VCBIND = AUX* MODV* BINDV;
VCBINDNEG = AUX* MODV* neg MODV* BINDV;
VCBINDNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCBIND;
VCBINDTO = AUX* (MODV | RESTV)* (inf | to) VCBIND;

NXnnp gx = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
           nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
           nxtrregr_ag | nxgeneprod | nxcmpx | nxmut | nxopr;

#
# Binding to UAS.
#
ev_bind_uas_va -> NXnnp gx (cma? wdt)? (VCBIND | VCBINDTO)
                 nxbrack? to? nxuas_pt
                 | NXnnp gx (cma? wdt)? VCCAUS nxbrack? nxuas_pt VCBINDTO;
ev_bind_uas_vp -> nxuas_pt (cma? wdt)? (VCCAUS VCBINDTO? | VCBIND)
                 by nxbrack? NXnnp gx
                 | nxuas_pt VCBINDTO nxbrack? NXnnp gx;
ev_bind_uas_va_n -> NXnnp gx (cma? wdt)? (VCBINDNEG | VCBINDNEGTO)
                  nxbrack? to? nxuas_pt
                  | NXnnp gx (cma? wdt)? VCCAUSNEG nxbrack?
                  nxuas_pt VCBINDTO
                  | NXnnp gx (cma? wdt)? VCCAUS nxbrack?
                  nxuas_pt VCBINDNEGTO;
ev_bind_uas_vp_n -> nxuas_pt (cma? wdt)? (VCCAUSNEG VCBINDTO? |
                  VCCAUS VCBINDNEGTO | VCBINDNEG) by nxbrack? NXnnp gx
                  | nxuas_pt VCBINDNEGTO nxbrack? NXnnp gx;

#
# Binding to URS.
#
ev_bind_urs_va -> NXnnp gx (cma? wdt)? (VCBIND | VCBINDTO)
                 nxbrack? to? nxurs
                 | NXnnp gx (cma? wdt)? VCCAUS nxbrack? nxurs_pt VCBINDTO;
ev_bind_urs_vp -> nxurs_pt (cma? wdt)? (VCCAUS VCBINDTO? |
                  VCBIND) by nxbrack? NXnnp gx
                  | nxurs_pt VCBINDTO nxbrack? NXnnp gx;
ev_bind_urs_va_n -> NXnnp gx (cma? wdt)? (VCBINDNEG |
                  VCBINDNEGTO) nxbrack? to? nxurs_pt

```

```

        | NXnnpqx (cma? wdt)? VCCAUSNEG nxbrack?
          nxurs_pt VCBINDTO
        | NXnnpqx (cma? wdt)? VCCAUS nxbrack?
          nxurs_pt VCBINDNEGTO;
ev_bind_urs_vp_n -> nxurs_pt (cma? wdt)? (VCCAUSNEG VCBINDTO? |
          VCCAUS VCBINDNEGTO | VCBINDNEG) by nxbrack? NXnnpqx
        | nxurs_pt VCBINDNEGTO nxbrack? NXnnpqx;

#
# Binding to promoter or binding sites.
#
ev_bind_prom_va -> NXnnpqx (cma? wdt)? (VCBIND | VCBINDTO)
          nxbrack? to? (nxbs | nxbs_ag | nxprom)
        | NXnnpqx (cma? wdt)? VCCAUS nxbrack?
          (nxbs | nxbs_ag | nxprom) VCBINDTO;
ev_bind_prom_vp -> (nxbs | nxbs_ag | nxprom) (cma? wdt)?
          (VCCAUS VCBINDTO? | VCBIND) by nxbrack? NXnnpqx
        | (nxbs | nxbs_ag | nxprom) VCBINDTO nxbrack? NXnnpqx;
ev_bind_prom_va_n -> NXnnpqx (cma? wdt)? (VCBINDNEG | VCBINDNEGTO)
          nxbrack? to? (nxbs | nxbs_ag | nxprom)
        | NXnnpqx (cma? wdt)? VCCAUSNEG nxbrack?
          (nxbs | nxbs_ag | nxprom) VCBINDTO
        | NXnnpqx (cma? wdt)? VCCAUS nxbrack?
          (nxbs | nxbs_ag | nxprom) VCBINDNEGTO;
ev_bind_prom_vp_n -> (nxbs | nxbs_ag | nxprom) (cma? wdt)?
          (VCCAUSNEG VCBINDTO? | VCCAUS VCBINDNEGTO |
          VCBINDNEG) by nxbrack? NXnnpqx
        | (nxbs | nxbs_ag | nxprom) VCBINDNEGTO
          nxbrack? NXnnpqx;

```

## B.27 Level 27

```

#####
#
# :Level27
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
    AUX    = vx | to;

```

```

MODV   = ax | rb;
RESTV  = vvz | vvn | vvd | vv;

VCCONT = AUX* MODV* contv;
VCCONTNEG = AUX* MODV* neg MODV* contv;
VCCONTNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCCONT;
VCCONTTTO = AUX* (MODV | RESTV)* (inf | to) VCCONT;

VCHAVE = (hvz | hv | hvg)
VCHAVENEG = (hvz | hv | hvg) neg | neg (hvz | hv | hvg);
VCHAVENEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCHAVE;
VCHAVETO = AUX* (MODV | RESTV)* (inf | to) VCHAVE;

#
# Containing UAS.
#
ev_cont_uas_va -> (nxpg | nxgene | nxorf | nxprom) (cma? wdt)?
                  (VCCONT | VCCONTTTO | VCHAVE | VCHAVETO)
                  nxbrack? nxuas_pt;
ev_cont_uas_vp -> nxuas_pt (cma? wdt)? (VCCONT | VCHAVE)
                  in nxbrack? (nxpg | nxgene | nxorf | nxprom)
                  | nxuas_pt (VCCONTTTO | VCHAVETO) nxbrack?
                  (nxpg | nxgene | nxorf | nxprom);
ev_cont_uas_va_n -> (nxpg | nxgene | nxorf | nxprom)
                  (cma? wdt)? (VCCONTNEG | VCCONTNEGTO |
                  VCHAVENEG | VCHAVENEGTO) nxbrack? nxuas_pt;
ev_cont_uas_vp_n -> nxuas_pt (cma? wdt)? (VCCONTNEG | VCHAVENEG)
                  in nxbrack? (nxpg | nxgene | nxorf | nxprom)
                  | nxuas_pt (VCCONTNEGTO | VCHAVENEGTO) nxbrack?
                  (nxpg | nxgene | nxorf | nxprom);

#
# Containing URS.
#
ev_cont_urs_va -> (nxpg | nxgene | nxorf | nxprom) (cma? wdt)?
                  (VCCONT | VCCONTTTO | VCHAVE | VCHAVETO) nxbrack?
                  nxurs_pt;
ev_cont_urs_vp -> nxurs_pt (cma? wdt)? (VCCONT | VCHAVE) in
                  nxbrack? (nxpg | nxgene | nxorf | nxprom)
                  | nxurs_pt (VCCONTTTO | VCHAVETO) nxbrack?
                  (nxpg | nxgene | nxorf | nxprom);
ev_cont_urs_va_n -> (nxpg | nxgene | nxorf | nxprom)
                  (cma? wdt)? (VCCONTNEG | VCCONTNEGTO |
                  VCHAVENEG | VCHAVENEGTO) nxbrack? nxurs_pt;
ev_cont_urs_vp_n -> nxurs_pt (cma? wdt)? (VCCONTNEG | VCHAVENEG)
                  in nxbrack? (nxpg | nxgene | nxorf | nxprom)
                  | nxurs_pt (VCCONTNEGTO | VCHAVENEGTO)

```

```

        nxbrack? (nxpg | nxgene | nxorf | nxprom);

#
# Containing binding site.
#
ev_cont_bs_va -> (nxgene | nxorf | nxprom) (cma? wdt)?
                (VCCONT | VCCONTTO | VCHAVE | VCHAVETO) nxbrack?
                (nxbs | nxbs_pt);
ev_cont_bs_vp -> (nxbs | nxbs_pt) (cma? wdt)? (VCCONT | VCHAVE)
                in nxbrack? (nxgene | nxorf | nxprom)
                | (nxbs | nxbs_pt) (VCCONTTO | VCHAVE)
                nxbrack? (nxgene | nxorf | nxprom);
ev_cont_bs_va_n -> (nxgene | nxorf | nxprom) (cma? wdt)?
                 (VCCONTNEG | VCCONTNEGTO | VCHAVENEG |
                 VCHAVENEGTO) nxbrack? (nxbs | nxbs_pt);
ev_cont_bs_vp_n -> (nxbs | nxbs_pt) (cma? wdt)?
                 (VCCONTNEG | VCHAVENEG) in nxbrack? (nxgene |
                 nxorf | nxprom)
                 | (nxbs | nxbs_pt) (VCCONTNEGTO | VCHAVENEGTO)
                 nxbrack? (nxgene | nxorf | nxprom);

```

## B.28 Level 28

```

#####
#
:Level28
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
MODV   = ax | rb;
AUX    = vx | to;
RESTV  = vvz | vvn | vvd | vv;

VCCAUS = AUX* MODV* cauv;
VCCAUSNEG = AUX* MODV* neg MODV* cauv;

VCINV  = AUX* MODV* (invv | requv);
VCINVNEG = AUX* MODV* neg MODV* (invv | requv);
VCINVNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCINV;

```

```

VCINVTO = AUX* (MODV | RESTV)* (inf | to) VCINV;

NXnnpgx = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
          nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
          nxtrregr_ag | nxgeneprod | nxorf | nxuas_pt | nxurs_pt
          | nxbs | nxbs_pt | nxcmpx | nxmut | nxfunct;

#
# Involves transcriptional activator.
#
ev_expr_inv_actr_va -> nxexpr (cma? wdt)? (VCINV | VCINVTO)
                      nxbrack? (nxactr_ag | nxtractr_ag | nxuas_pt);
ev_expr_inv_actr_vp -> (nxactr_ag | nxtractr_ag | nxuas_pt)
                      (cma? wdt)? VCINV (for | from | in | to)
                      nxbrack? nxexpr
                      | (nxactr_ag | nxtractr_ag | nxuas_pt)
                      VCINVTO nxbrack? nxexpr;
ev_expr_inv_actr_va_n -> nxexpr (cma? wdt)? (VCINVNEG | VCINVNEGTO)
                          nxbrack? (nxactr_ag | nxtractr_ag | nxuas_pt);
ev_expr_inv_actr_vp_n -> (nxactr_ag | nxtractr_ag | nxuas_pt)
                          (cma? wdt)? VCINVNEG (for | from | in | to)
                          nxbrack? nxexpr
                          | (nxactr_ag | nxtractr_ag | nxuas_pt)
                          VCINVNEGTO nxbrack? nxexpr;

#
# Involves transcriptional repressor.
#
ev_expr_inv_repr_va -> nxexpr (cma? wdt)? (VCINV | VCINVTO)
                      nxbrack? (nxrepr_ag | nxtrrepr_ag | nxurs_pt);
ev_expr_inv_repr_vp -> (nxrepr_ag | nxtrrepr_ag | nxurs_pt)
                      (cma? wdt)? VCINV (for | from | in | to)
                      nxbrack? nxexpr
                      | (nxrepr_ag | nxtrrepr_ag | nxurs_pt)
                      VCINVTO nxbrack? nxexpr;
ev_expr_inv_repr_va_n -> nxexpr (cma? wdt)? (VCINVNEG | VCINVNEGTO)
                          nxbrack? (nxrepr_ag | nxtrrepr_ag | nxurs_pt);
ev_expr_inv_repr_vp_n -> (nxrepr_ag | nxtrrepr_ag | nxurs_pt)
                          (cma? wdt)? VCINVNEG
                          (for | from | in | to) nxbrack? nxexpr
                          | (nxrepr_ag | nxtrrepr_ag | nxurs_pt)
                          VCINVNEGTO nxbrack? nxexpr;

#
# Expression involves.
#
ev_expr_inv_va -> nxexpr (cma? wdt)? (VCINV | VCINVTO) nxbrack? NXnnpgx;

```

## The regular expression grammar

```
ev_expr_inv_vp -> NXnnpgx (cma? wdt)? VCINV
                 (for | from | in | to) nxbrack? nxexpr
                 | NXnnpgx VCINVTO nxbrack? nxexpr;
ev_expr_inv_va_n -> nxexpr (cma? wdt)? (VCINVNEG | VCINVNEGTO)
                   nxbrack? NXnnpgx;
ev_expr_inv_vp_n -> NXnnpgx (cma? wdt)? VCINVNEG
                   (for | from | in | to) nxbrack? nxexpr
                   | NXnnpgx VCINVNEGTO nxbrack? nxexpr;
```

## B.29 Level 29

```
#####
#
:Level29
#
# The level vreg chunks relations between previously recognised entities
#
#####
#
# Variable declaration
#
RESTV = vvz | vvn | vvd | vv;
MODV  = ax | rb;
AUX   = vx | to;

VCCAUS = AUX* MODV* cauv;
VCCAUSNEG = AUX* MODV* neg MODV* cauv;

VCINV = AUX* MODV* invv;
VCINVNEG = AUX* MODV* neg MODV* invv;
VCINVNEGTO = AUX* (MODV | RESTV)* (infneg | neg to | to neg) VCINV;
VCINVTO = AUX* (MODV | RESTV)* (inf | to) VCINV;

NXnnpgx = nxpg | nxgene | nxrna | nxprot | nxf | nxactr_ag |
          nxrepr_ag | nxregr_ag | nxtractr_ag | nxtrrepr_ag |
          nxtrregr_ag | nxgeneprod | nxorf | nxprom | nxcmpx |
          nxmut;

#
# Is transcriptional activator.
#
ev_be_tractr_va -> NXnnpgx (cma? wdt)? (bez | ber | to be)
                  nxbrack? nxtractr_pt;
```

```

ev_be_tractr_vp -> tractr_pt (cma? wdt)? (bez | ber) nxbrack? NXnnpgrx
    | tractr_pt to be nxbrack? NXnnpgrx;
ev_be_tractr_va_n -> NXnnpgrx (cma? wdt)? (bez neg | ber neg |
    neg to be | to neg be) nxbrack? nxtractr_pt;
ev_be_tractr_vp_n -> nxtractr_pt (cma? wdt)? (bez neg | ber neg)
    nxbrack? NXnnpgrx
    | nxtractr_pt (neg to be | to neg be) nxbrack? NXnnpgrx;

#
# Is transcriptional repressor.
#
ev_be_trrepr_va -> NXnnpgrx (cma? wdt)? (bez | ber | to be)
    nxbrack? nxtrrepr_pt;
ev_be_trrepr_vp -> nxtrrepr_pt (cma? wdt)? (bez | ber) nxbrack? NXnnpgrx
    | nxtrrepr_pt to be nxbrack? NXnnpgrx;
ev_be_trrepr_va_n -> NXnnpgrx (cma? wdt)? (bez neg | ber neg |
    neg to be | to neg be) nxbrack? nxtrrepr_pt;
ev_be_trrepr_vp_n -> nxtrrepr_pt (cma? wdt)? (bez neg | ber neg)
    nxbrack? NXnnpgrx
    | nxtrrepr_pt (neg to be | to neg be) nxbrack? NXnnpgrx;

#
# Is transcriptional regulator.
#
ev_be_trregr_va -> NXnnpgrx (cma? wdt)? (bez | ber | to be)
    nxbrack? nxtrregr_pt;
ev_be_trregr_vp -> nxtrregr_pt (cma? wdt)? (bez | ber) nxbrack? NXnnpgrx
    | nxtrregr_pt to be nxbrack? NXnnpgrx;
ev_be_trregr_va_n -> NXnnpgrx (cma? wdt)? (bez neg | ber neg |
    neg to be | to neg be) nxbrack? nxtrregr_pt;
ev_be_trregr_vp_n -> nxtrregr_pt (cma? wdt)? (bez neg | ber neg)
    nxbrack? NXnnpgrx
    | nxtrregr_pt (neg to be | to neg be) nxbrack? NXnnpgrx;

```

## B.30 Level 30

```

#####
#
# :Level30
#
# The level categorizes events according to biological conclusions
#
#####

```

## The regular expression grammar

```
#
# Activation of expression.
#
ev_act_expr_rx -> nxev_uas_in | nxev_expr | nxev_bind_uas
                 | nxev_actr_of_expr
                 | nxev_abil_act_expr
                 | ev_trregr_act_va | ev_tractr_reg_va
                 | ev_act_expr_va | ev_rep_expr_va_r
                 | ev_cont_uas_vp
                 | ev_expr_inv_actr_vp
                 | ev_be_tractr_va;
ev_act_expr_xr -> nxev_tractr
                 | ev_trregr_act_vp | ev_tractr_reg_vp
                 | ev_act_expr_vp | ev_rep_expr_vp_r
                 | ev_cont_uas_va
                 | ev_expr_inv_actr_va
                 | ev_be_tractr_vp;

#
# Repression of expression.
#
ev_rep_expr_rx -> nxev_urs_in | nxev_bind_urs
                 | nxev_repr_of_expr
                 | nxev_abil_rep_expr
                 | ev_trregr_rep_va | ev_trrepr_reg_va
                 | ev_act_expr_va_r | ev_rep_expr_va
                 | ev_cont_urs_vp
                 | ev_expr_inv_repr_vp
                 | ev_be_trrepr_va;
ev_rep_expr_xr -> nxev_trrepr
                 | ev_trregr_rep_vp | ev_trrepr_reg_vp
                 | ev_act_expr_vp_r | ev_rep_expr_vp
                 | ev_cont_urs_va
                 | ev_expr_inv_repr_va
                 | ev_be_trrepr_vp;

#
# Regulation of expression.
#
ev_reg_expr_rx -> nxev_bs_in | nxev_dep_expr | nxev_bind_prom
                 | nxev_regr_of_expr
                 | nxev_abil_reg_expr
                 | ev_trregr_reg_va
                 | ev_act_expr_va_l | ev_rep_expr_va_l | ev_reg_expr_va
                 | ev_bind_prom_va
                 | ev_cont_bs_vp
```

```

        | ev_expr_inv_vp
        | ev_be_trregr_va;
ev_reg_expr_xr -> nxev_trregr
        | ev_trregr_reg_vp
        | ev_act_expr_vp_1 | ev_rep_expr_vp_1 | ev_reg_expr_vp
        | ev_bind_prom_vp
        | ev_cont_bs_va
        | ev_expr_inv_va
        | ev_be_trregr_vp;
```



# Bibliography

- Aaronson, D. S. and Horvath, C. M. (2002). A road map for those who don't know jak-stat. *Science*, **296**, 1653–1655.
- Abecker, A. and van Elst, L. (2003). Ontologies for knowledge management. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, chapter 22, pages 435–454. Springer.
- Abney, S. (1996). Partial parsing via finite-state cascades. In *Proceedings of the ESLLI '96 Robust Parsing Workshop*, pages 8–15, Prague, Czech Republic.
- Alberts, B., Bray, D., Johnson, A., Lewis, J., Raff, M., Roberts, K., and Walter, P. (1998). *Essential Cell Biology: An introduction to molecular biology of the cell*. Garland Publishing Inc., New York.
- Allan, J. (2003). Hard track overview in trec 2003.
- Allen, J. F. and Ferguson, G. (1994). Actions and events in interval temporal logic. Technical Report TR521.
- Ashburner, M., Ball, C., Blake, J., Botstein, D., Butler, H., Cherry, J., Davis, A., Dolinski, K., Dwight, S., Eppig, J., Harris, M., Hill, D., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J., Richardson, J., Ringwald, M., Rubin, G., and Sherlock, G. (2000). Gene ontology: Tool for the unification of biology. *Nature Genetics*, **25**(1), 125–29.
- Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.
- Bairoch, A. (2003). Swiss-prot protein knowledgebase user manual. <http://www.expasy.org/sprot/userman.html>.
- Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., and Wheeler, D. L. (2002). GenBank. *Nucleic Acids Res.*, **30**, 17–20.
- Benthem, J. F. A. K. v. (1983). *The Logic of Time*. D. Reidel Publishing Company.
- Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M. C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S., and Schneider, M. (2003). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2003. *Nucleic Acids Research*, **31**, 365–370.

## Bibliography

- Brill, E. (1992). A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152–155, Trento, IT.
- Bußmann, H. (1983). *Lexikon der Sprachwissenschaft*. Kröner Verlag.
- Burgun, A. and Bodenreider, O. (2001). Mapping the umls semantic network into general ontologies.
- Casati, R. and Varzi, A. C. (1994). *Holes and Other Superficialities*. MIT Press, Cambridge, Massachusetts.
- Casati, R. and Varzi, A. C. (1997). Spatial entities. In O. Stock, editor, *Spatial and Temporal Reasoning*, pages 73–96. Kluwer Academic Publishers, Dordrecht.
- Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, **2**, 65–73.
- Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O., and Herskowitz, L. (1998). The transcriptional program of sporulation in budding yeast. *Science*, **282**, 699–705.
- Ciaramita, M., Gangemi, A., Ratsch, E., Šarić, J., and Rojas, I. (2005). Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence IJCAI*.
- Cimiano, P. (2002). On the resolution of bridging references within information extraction systems. Master's Thesis.
- Cimiano, P., Reyle, U., and Šarić, J. (2004). Ontology driven discourse analysis for information extraction. *Data and Knowledge Engineering Journal*.
- Coden, A. R., Pakhomov, S. V., Ando, R. K., Duffy, P. H., and Chute, C. G. (2004). Domain-specific language models and lexicons for tagging. Technical report, IBM Research.
- Cohn, A. (2001). Formalising bio-spatial knowledge. In C. Welty and B. Smith, editors, *Proceedings of the 2nd International Conference of Geographic Information Science (FOIS'01)*. ACM Press.
- Collier, N., Nobata, C., and Tsujii, J. (2002). Automatic acquisition and classification of terminology using a tagged corpus in the molecular biology domain. *Journal of Terminology, John Benjamins*, **7**(2), 239–257.
- Craswell, N., Hawking, D., Wilkinson, R., and Wu, M. (2004). Overview of the trec 2003 web track.
- Cycorp (2001a). Creators of the cycorp knowledge base. URL: <http://www.cyc.com/>.

- Cycorp (2001b). Features of cycl. URL: <http://www.cyc.com/cycl.html>.
- Demetriou, G. and Gaizauskas, R. (2003). Corpus resources for development and evaluation of a biological text mining system. In *Proceedings of the Third Meeting of the Special Interest Group on Text Mining*, Australia, Brisbane.
- DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, **278**, 680–686.
- Dingare, S., Finkel, J., Nissim, M., Manning, C., and Grover, C. (2004). A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. In *The 2004 BioLink meeting: Linking Literature, Information and Knowledge for Biology*. at ISMB 2004.
- Dwight, S. S., Harris, M. A., Dolinski, K., Ball, C. A., Binkley, G., Christie, K. R., Fisk, D. G., Issel-Tarver, L., Schroeder, M., Sherlock, G., Sethuraman, A., Weng, S., Botstein, D., and Cherry, J. M. (2002). Saccharomyces Genome Database (SGD) provides secondary gene annotation using the Gene Ontology (GO). *Nucleic Acids Research*, **30**, 69–72.
- Fasman, K. H., Letovsky, S., Cottingham, R. W., and Kingsbury, D. T. (1996). Improvements to the gdb human genome data base. *Nucleic Acids Research*, **24**(1), 57–63.
- Fellbaum, C. (1998a). Wordnet an electronic lexical database.
- Fellbaum, C. (1998b). *WordNet An Electronic Lexical Database*. MIT Press.
- Francis, W. N. and Kucera, H. (1979). *Brown corpus manual, Manual of Information to accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University, Providence, USA.
- Franzén, K., Eriksson, G., Olsson, F., Asker, L., and Linden, P. (2002). Protein names and how to find them. *International Journal of Medical Informatics*, **67**(3), 49–61.
- Frunder, B., Hillen, E., and Rohlf, U. (1995). *Wörterbuch der Chemie*. Deutscher Taschenbuch Verlag.
- Fukuda, K., Tsunoda, T., Tamura, A., and Takagi, T. (1998). Toward information extraction: Identifying protein names from biological papers. In *Proceedings of the Pacific Symposium on Biocomputing (PSB'98)*, pages 705–716, Maui, Hawaii.
- Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening ontologies with dolce. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, pages 166–181.
- Gavin, A.-C., Bösch, M., Krause, R., Grandi, P., Marzioch, M., Bauer, A., Schultz, J., Rick, J. M., Michon, A.-M., Cruciat, C.-M., Remor, M., Höfert, C., Schelder, M., Brajenovic, M., Ruffner, H., Merino, A., Klein, K., Hudak, M., Dickson, D., Rudi, T., Gnau,

## Bibliography

- V, Bauch, A., Bastuck, S., Huhse, B., Leutwein, C., Heurtier, M.-A., Copley, R. R., Edelman, A., Querfurth, E., Rybin, V., Drewes, G., Raida, M., Bouwmeester, T., Bork, P., Seraphin, B., Kuster, B., Neubauer, G., and Superti-Furga, G. (2002). Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, **415**, 141–147.
- Gerstenberger, C. (2001). Semantische analyse von namen organischer verbindungen.
- Goffeau, A. *et al.* (1997). The yeast genome directory. *Nature*, **387 suppl.**, 5–105.
- Gomez-Perez, A., Fernandez-Lopez, M., and Corcho, O. (2004). *Ontological Engineering*. Springer.
- Grefenstette, G. and Tapanainen, P. (1994). What is a word, what is a sentence? problems of tokenization. In *The 3rd International Conference on Computational Lexicography*, pages 79–87.
- Gruber, T. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *Formal Analysis in Conceptual Analysis and Knowledge Representation*. Kluwer.
- Guarino, N. (1997). Understanding, building and using ontologies. *International Journal of Human and Computer Studies*, **46**, 293–310.
- Guarino, N. (1998). Formal ontology in information systems. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98*, Trento, Italy. IOS Press.
- Guarino, N. and Giaretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, Amsterdam.
- Harman, D. (2003). The development and evolution of trec and duc. In *Proceedings of the Third NTCIR Workshop*.
- Hermjakob, H., Fleischmann, W., and Apweiler, R. (1999). Swissknife — 'lazy parsing' of swiss-prot. *Bioinformatics*, **15**(9), 771–772.
- Hermjakob\*, H., Montecchi-Palazzi, L., Lewington, C., Mudali, S., Kerrien, S., Orchard, S., Vingron, M., Roechert, B., Roepstorff, P., Valencia, A., Margalit, H., Armstrong, J., Bairoch, A., Cesareni, G., Sherman, D., and Apweiler, R. (2004). Intact: an open source molecular interaction database. *Nucleic Acids Research*, **32**.
- Hersh, W. and Bhupatiraju, R. T. (2003). Trec genomics track overview.
- Hertz-Fowler, C., Peacock, C. S., Wood, V., Aslett, M., Kerhornou, A., Mooney, P., Tivey, A., Berriman, M., Hall, N., Rutherford, K., Parkhill, J., Ivens, A. C., Rajandream, M.-A., and Barrell, B. (2004). Genedb: a resource for prokaryotic and eukaryotic organisms. *Nucleic Acids Research*, **32**(Database-Issue), 339–343. <http://www.genedb.org>.

- Hirschman, L. (2003). Using biological resources to bootstrap text mining. Presentation to the Massachusetts Biotechnology Council Informatics Committee.
- Hirschman, L., Colosimo, M., Morgan, A., and Yeh, A. (2005). Overview of biocreative task 1b: normalized gene lists. *BMC Bioinformatics*, **6 (Suppl.1)**.
- Hobbs, J. R. (1985). Granularity. In *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI*.
- Hobbs, J. R. (2003). Information extraction from biomedical text. *J. Biomedical Informatics*, **35**, 260–264.
- Int. Human Genome Sequencing Consortium (2001). Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.
- IUPAC (1979). *Nomenclature of Organic Chemistry. Sections A, B, C, D, E, F, and H*. Pergamon Press, Oxford.
- IUPAC, C. o. N. o. O. C. (1993). *A Guide to IUPAC Nomenclature of Organic Compounds. Recommendations 1993*. Blackwell Scientific Publications, Oxford.
- Jones, K. S. and Willet, P. (1997). *Readings in Information Retrieval*. Morgan Kaufmann Publishers.
- Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluwer.
- Kanehisa, M. (1997). A database for post-genome analysis. *Trends in Genetics*, (13), 375–376.
- Kanehisa, M. and Goto, S. (2000). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, (28), 27–30. KEGG: Kyoto Encyclopedia of Genes and Genomes.
- Karp, P., Riley, M., Saier, M., Paulsen, I., Collado-Vides, J., Paley, S., Pellegrini-Toole, A., Bonavides, C., and Gama-Castro, S. (2002). The ecocyc database. *Nucleic Acids Research*, **30**(1), 56–58.
- Keen, G., Burton, J., Crowley, D., Dickinson, E., Espinosa-Lujan, A., Franks, E., Harger, C., Manning, M., March, S., McLeod, M., O'Neill, J., Power, A., Pumilia, M., Reinert, R., Rider, D., Rohrlich, J., Schwertfeger, J., Smyth, L., Thayer, N., Troup, C., and Fields, C. A. (1996). The genome sequence database (gsdb): meeting the challenge of genomic sequencing. *Nucleic Acids Research*, **24**(1), 13–16.
- Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, **19 suppl. 1**, i180–i182.
- König, E. and Lezius, W. (2002a). The TIGER language - a description language for syntax graphs. Part 1: User's guidelines. Technical report, IMS, University of Stuttgart.

## Bibliography

- König, E. and Lezius, W. (2002b). The TIGER language - a description language for syntax graphs. Part 2: Formal definition. Technical report, IMS, University of Stuttgart.
- Krymolowski, Y., Alex, B., and Leidner, J. L. (2004). BioCreative task 2.1: The Edinburgh-Stanford system. In *Proceedings of the first BioCrEatIvE Workshop*.
- Kumar, A. and Smith, B. (2003). The universal medical language system and the gene ontology: Some critical reflections. In *KI*, pages 135–148.
- Lackie, J. M. and Dow, J. (2000). *The dictionary of cell & molecular biology*. Academic Press, London, GB, 3rd edition.
- Lascarides, A. and Asher, N. (1993). Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy*, **16**(5), 437–493.
- Levesque, H. and Brachman, R. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational intelligence*, (3), 78–93.
- Lezius, W. (2002a). *Ein Suchwerkzeug für syntaktisch annotierte Textcorpora*. Ph.D. thesis, Universität Stuttgart.
- Lezius, W. (2002b). TIGERSearch—ein Suchwerkzeug für Baumbanken. In S. Busemann, editor, *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*, pages 107–114, Saarbrücken, Germany.
- Lezius, W., Biesinger, H., and Gerstenberger, C. (2002). *TIGER-XML Quick Reference Guide*. IMS, University of Stuttgart.
- Maedche, A. and Staab, S. (2000). Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th International Conference on Software and Knowledge Engineering*.
- Marsh, E. and Perzanowski, D. (1998). Muc-7 evaluation of i.e. technology: Overview of results. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.
- Monz, C. (2003). *From Document Retrieval to Question Answering*. Ph.D. thesis, Institute for Logic, Language and Computation, Amsterdam, Netherlands.
- Nardi, D. and Brachman, R. J. (2003). An introduction to description logics. In *The Description Logic Handbook – Theory, Implementation and Applications*, chapter 1, pages 5–44. Cambridge University Press.
- Nenadić, G., Rice, S., Spasić, I., and Ananiadou, S. and Stapley, B. (2003). Selecting text features for gene name classification: from documents to terms. In S. Ananiadou and J. Tsujii, editors, *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 121–128.
- Netzel, R., C., P.-I., Bork, P., and Andrade, M. A. (2003). The way we write. *EMBO Rep.*, **4**, 446–451.

- Ogren, P., Cohen, K., Acquah-Mensah, G., and Eberlein, J. (2004). The compositional structure of gene ontology terms. In *Proceedings of the 9th Pacific Symposium on Biocomputing (PSB 2004)*.
- Peter, K., Vollhardt, C., and Schore, N. E. (1999). *Organic Chemistry – Structure and Function*. WH Freeman and Company, 3rd edition.
- PubMed (2001). The pubmed database. <http://www.ncbi.nlm.nih.gov/pubmed/>.
- Reyle, U. (2005). Understanding chemical terminology. *Terminology*.
- Reyle, U. and Šarić, J. (2002). Corpus driven information extraction. In R. Baud and P. Ruch, editors, *Proceedings of the EFMI Workshop on Natural Language Processing in Biomedical Applications*, Nicosia, Cyprus. unknown.
- Robinson, P., Brown, E., Burger, J., Chinchor, N., Douthat, A., Ferro, L., and Hirschman, L. (1999). Overview: Information extraction from broadcast news. In *Proceedings of the DARPA Speech and Language Technology Workshop*.
- Sanguino, R. (2001). Evaluation of cyc. Lef grant report, CSC. URL: [www.csc.com/aboutus/lef/mds67\\_off/uploads/sanguino\\_eval\\_cyc.pdf](http://www.csc.com/aboutus/lef/mds67_off/uploads/sanguino_eval_cyc.pdf).
- Santorini, B. (1990). Part-of-speech tagging guidelines for the penn treebank project. Technical Report 3rd Revision, 2nd Printing, University of Pennsylvania.
- Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK. unknown.
- Schmid, H. (2000). Unsupervised learning of period disambiguation for tokenisation. Technical report, Institut für Maschinelle Sprachverarbeitung, University of Stuttgart.
- Schomburg, I., Chang, A., Hofmann, O., Ebeling, C., Ehrentreich, F., and Schomburg, D. (2002). Brenda: a resource for enzyme data and metabolic information. *Trends Biochem Science*, **27**(1).
- Schulze-Kremer, S. (1998). Ontologies for molecular biology. In *3rd Pacific Symposium on Biocomputing*, pages 705–716.
- Shah, P. K., Perez-Iratxeta, C., Bork, P., and Andrade, M. A. (2003). Information extraction from full text scientific articles: Where are the keywords? *BMC Bioinformatics*, **4**, 20.
- Soboro, I. and Harman, D. (2003). Overview of the trec 2003 novelty track.
- Sowa, J. F. (1999). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co.

## Bibliography

- Stein, L., Sternberg, P., Durbin, R., Thierry-Mieg, J., and Spieth, J. (2001). WormBase: network access to the genome and biology of *Caenorhabditis elegans*. *Nucleic Acids Res.*, **29**, 82–84.
- The FlyBase Consortium (2003). The flybase database of the drosophila genome projects and community literature. *Nucleic Acids Research*, **31**, 172–175. URL: <http://flybase.org/>.
- US Dept of Health and Human Services, NIH, N. L. o. M. (2002). Unified medical language system. URL: <http://www.nlm.nih.gov/research/umls/>.
- van Rijsbergen, C. J. (1979). *Information Retrieval*. Number 2nd edition. Butterworth and Co, London.
- Vendler, Z. (1957). Verbs and times. *Philosophical Review*, **66**, 143–160. Revised in Vendler (1967), 97-121; reprinted in Schopf (1974), 217-234.
- Venter, J. *et al.* (2001). The sequence of the human genome. *Science*, **291**, 1304–1351.
- Voorhees, E. M. (2003a). Overview of the trec 2003 question answering track.
- Voorhees, E. M. (2003b). Overview of the trec 2003 robust retrieval track.
- Šarić, J., Jensen, L. J., Ouzounova, R., Rojas, I., and Bork, P. (2004a). Extraction of regulatory gene expression networks from pubmed. In *Proceedings of the ACL 2004 Conference*, pages 192–199, Barcelona, Spain.
- Šarić, J., Jensen, L. J., and Rojas, I. (2004b). Large-scale extraction of gene regulation for model organisms in an ontological context. *In Silico Biol.* **5**, 0004.
- Šarić, J., Rojas, I., Ratsch, E., Kania, R., Wittig, U., and Gangemi, A. (2004c). Modelling gene expression. In *Proceedings of Nettab 2004*, Camerino, Italy.
- Šarić, J., Jensen, L. J., Ouzounova, R., Rojas, I., and Bork, P. (2005a). Extraction of regulatory gene/protein networks from Medline. doi:10.1093/bioinformatics/bti597.
- Šarić, J., Jensen, L. J., Ouzounova, R., Bork, P., and Rojas, I. (2005b). Large-scale extraction of protein/gene relations for model organisms. In *Proceedings of the First International Symposium on Semantic Mining in Biomedicine*.
- Wain, H., Lovering, R., Bruford, E., M.J., L., Wright, M., and S., P. (2002). Guidelines for human gene nomenclature. *Genomics*, **79**(4), 464–470.
- Wain, H., M.J., L., F., D., V.K., K., and S., P. (2004). Genew: the human gene nomenclature database, 2004 updates. *Nucleic Acids Research*, **32**.
- Weeber, M., Schijvenaars, R., van Mulligen, E., Mons, B., Jelier, R., van der Eijk, C., and Kors, J. (2003). Ambiguity of human gene symbols in locuslink and medline: Creating an inventory and a disambiguation test collection. In *Proceedings of AMIA Symposium*, pages 704 – 708.

- Wheeler, D., Chappey, C., Lash, A., Leipe, D., Madden, T., Schuler, G., Tatusova, T., and Rapp, B. (2000). Database resources of the national center for biotechnology information. *Nucleic Acids Research*, **1**(28).
- Wheeler, D. L., Church, D. M., Edgar, R., Federhen, S., Helmberg, W., L., M. T., U., P. J., D., S. G., M., S. L., Sequeira, E., Suzek, T. O., Tatusova, T. . A., and Wagner, L. (2004). Database resources of the national center for biotechnology information: update. *Nucleic Acids Res.*, **32**, D35–40.
- Wittig, U., Weidemann, A., Kania, R., Peiss, C., and Rojas, I. (2004). Classification of chemical compounds to support complex queries in a pathway database. *Comparative and Functional Genomics*, (5), 000–000.
- Yamamoto, K., Kudo, T., Konagaya, A., and Matsumoto, Y. (2003). Protein name tagging for biomedical annotation in text. In S. Ananiadou and J. Tsujii, editors, *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 65–72.
- Yeh, A., Hirschman, L., Morgan, A., and Colosimo, M. (2005). Biocreative task 1a: gene mention finding evaluation. *BMC Bioinformatics*, **6** (Suppl.1).