Natural Language Understanding (NLU)

Automatic Information Extraction (IE) from Biomedical Texts

Author: Rune Sætre

IDI, NTNU
Sem Sælands vei 7-9
N-7491 Trondheim
NORWAY

CIS, LMU
Oettingenstraße 67
D-80538 München
GERMANY

Abstract

This diploma thesis is about Natural Language Understanding (NLU) in general and more concretely about applications to microbiological texts on the topic "gene- and protein-activations". The first part is a review of different current research approaches in the field of NLU and "bio-linguistics". The second part will look into the bottom-up grammar building approach that is sketched in the article **"The Construction of Local Grammars**" by Maurice Gross. The visualization system **"Unitex**", made by Sébastien Paumier, will be used to construct these local grammars. The results will be compared to the full-parsing approach used in GeneTUC. In the third and last part a plan for future work will be given.

The preliminary results suggest that the medical language is constrained enough for the Local Grammar approach to work. 38 graphs were constructed to capture the essence of 59 "activate-sentences", and 18 graphs where created to capture all the different entity names that were used in the sentences. When the graphs were applied to a new text for testing, many of the constructed "activation-patterns" also matched in the new text.

Acknowledgements

First of all I would like to thank my two supervisors for this project: Professor Dr. Franz Guenthner at "Centrum für Informationsverarbeitung und Sprachwissenschaft" (CIS) for inviting me to do the project at Ludwig-Maximilian University in Munich, and associate Professor Tore Amble at NTNU for encouraging me to go. Thanks to my co-supervisor Professor Arne Halaas at NTNU, for introducing me to Franz Guenthner in the first place.

During the work with chapter 1 I had many video-conferences with Henrik Tveit at NTNU to discuss the advantages and disadvantages of stochastic based methods versus rule-based methods. After the chapter was finished, Nancy Lee Eik-Nes provided valuable feed-back on the language and form of the chapter. She also deserves thanks for being very flexible about my final exam dates last semester, and thus making it possible for me to do this exchange project.

Chapter 2 and the parts about linguistics were written after valuable discussions with the co-workers at CIS. Petra Maier, Holger Bosk, Jörg Schuster and the other members of the Biopath project group, and Sebastian Nagel, Mariya Vitusevych, Felix Golcher, Svetlana Stasiuk and the other students in the tagging seminar provided good input to this material. I am very grateful for the opportunity Petra Maier gave me when she invited me to participate in the weekly Biopath meetings at CIS every week. That gave me a great chance to present and think about my own work, and also participating in discussions about some of the common problems in this kind of information extraction.

Franz Guenthner deserves extra mentioning, because he was the one who convinced me to go ahead with the Unitex approach, described in Chapter 3. He also invited Sébastien Paumier (the creator of Unitex) to a tagging seminar at CIS to give a first hand presentation of the system. Sébastien Paumier has been very helpful, and has answered all questions about Unitex promptly by e-mail.

Astrid Lægreid is our key player in the cooperation between computer science and micro-biology. Without her help, it would have been much

harder to find the “right” (or interesting) extraction problems, good text-sources for “training” and testing the grammars, and the correct semantics to go with the sometimes extremely complex sentences of micro-biology.

During the final writing-phase Zoran Constantinescu-Fülöp, Sobah Petersen, Jörg Cassens and Martin Thorsen Ranang provided very good feed-back and spell-checking assistance.

And, finally, I would like to thank all my friends (including all the people I forgot to mention by name in these Acknowledgements) in Munich and Trondheim, for helping me having a good time and for “charging my batteries” when I am away from work.

Preface

This diploma thesis serves both as a large final project after 5 years of computer science education and as the beginning of a PhD work that will last another 3 years. The thesis contains three main parts. Part 1 (Chapter 1 and 2) is a literature review with the goal of determining where the current research barriers in computational linguistics applied to micro-biological texts are. Part 2 (Chapter 3, 4 and 5) describes the project part of the work, and part 3 (Chapters 6 and 7) describes how the PhD work should proceed in the next phase.

Because of a "sudden" invitation to spend this semester in Munich, part 2 turned out to be quite different from what was originally expected. To begin with, the plan was to focus on how to develop GeneTUC further, but my advisor in Munich is very interested in Local Grammars and he convinced me to try using that approach instead of following the original idea. That means that my part in the GeneTUC project was put on hold for half a year, and I learned how to use the Unitex system to solve the same kind of tasks that the parser in GeneTUC solves during the database build-up phase (the tell-phase, in a tell-and-ask system).

GeneTUC and Unitex use two different approaches to solve the parsing problem, but after this project I believe that GeneTUC could benefit greatly from using parts of the Unitex system. Especially, the use of graphs when constructing grammars is very promising, and makes it easy also for people with little computer knowledge to produce grammars to fit their needs. How the two systems should be integrated is an open question that will be decided in the following PhD work.

This thesis is written in Microsoft Word, using font *Palatino Linotype* 11pt.

Rune Sætre, April 25, 2003

Content

List of Figures

1 "Natural Language Understanding" Articles Reviewed

The purpose of this chapter is to give an overview of existing literature and methods used in the field of Natural Language Processing (NLP) with a special focus on Natural Language Understanding (NLU) and Information Extraction (IE). The domain of the IE will be biomedical texts describing gene and protein interactions. 15 NLP articles have been selected, and together they cover most of the recent advances in biomedical IE. This diploma thesis is leading up to a PhD thesis that involves working with a system called GeneTUC, and the idea in GeneTUC is that the text is to be fully parsed. Only one other article has been found that describes full parsing of biomedical texts [18], and that article will therefore receive extra attention in this review. The other articles in the collection use various methods of shallow (partial) parsing, or stochastic (statistical) calculations to analyze the language. One of the 15 articles is an interesting article on Local Grammars [8]. It describes the use of Finite State Transducers (Pattern Matching) to extract exact knowledge from texts. This represents a bottom-up approach that will be compared to the top-down approach used in GeneTUC.

1.1 Introduction

There are two current main approaches to information extraction from biomedical texts. One approach is the rule-based and grammatical one that is often called Natural Language Understanding (NLU). The other approach is the statistical or pattern matching one, usually referred to as Natural Language Processing (NLP). Future systems are likely to be hybrid systems, including techniques from both of these approaches, since NLU and NLP often offer complementary solutions to the same problem. Sometimes NLU is thought of as a subset of NLP, since "understanding" is also really just some kind of processing. The way GeneTUC *understands* a text is by translating it into an event-logic form called TUC Query Language (TQL).

This chapter is split into several sections, each dealing with a special topic regarding NLU/NLP of microbiological texts: The next section will discuss the *terminology* of the field, and give some definitions of common terms. It is followed by sections about *full parsing*, what the common *goals* of microbiological IE are, what specific *systems* are implemented around the world, and last a short *conclusion*.

1.2 Terminology

One of the goals in the GeneTUC project [16] is to do full parsing of microbiological texts. This section will briefly explain the terminology of full parsing and all the other approaches that are being used to reach the end goal of automatic Information Extraction (IE) in the medical domain. Specifically the following will be explained: The difference between IE and IR, NLU and NLP, full parsing and partial parsing, global and local grammars, and finally the difference between robust and non-robust parsers. The last section provides an explanation of what is meant by corpus-based approaches.

1.2.1 Information Retrieval and Information Extraction

While Information Retrieval (IR) and IE are both dealing with some form of text searching, they are quite different in terms of what output or results they produce. IR is the simple classical approach to text searching, as it is done e.g. in Google [28] and other search engines on the Internet. In IR the user enters some words of interest, and then all the documents containing these words are listed. The document list can be ordered

accordingly to how many times each search word occurs, how close the different search words are clustered in the document and so on. In this approach, the user has to run many different searches to cover all the possible different search words to describe the fact that she is actually looking for. Also, for every search she might have to read all the articles returned by the search engine, just to see if they really are of interest or not.

Information Extraction (IE) seeks to reduce the user's workload by adding reasoning to the IR process. With IE the computer will have some knowledge about synonyms and different sentence forms that actually express the same basic facts. That means that the user only has to specify the question that she has, and then the computer will do the tedious work of running several different IR searches, and skimming every single retrieved article to see whether or not it is of interest. The end result from IE can be simple yes/no answers to different questions or it can be specific facts that are extracted from various articles and then used to build databases for quick and easy lookup later.

1.2.2 NLP and NLU, Statistical and Rule-based Approaches

In the literature, full parsing and other symbolic approaches are commonly called Natural Language Understanding. Symbolic approaches means using symbols that have a defined meaning both for humans and machines. The other approaches, e.g. statistical, are often called Natural Language Processing. This use of terms tells us that NLU seeks to do something more than just process the text from one format to another. The end goal is to transform the text into something that computers can "understand". That means that the computer should be able to answer natural language (e.g. English) questions about the text, and also be able to reason about facts from different texts. The field of NLU is strongly connected to the field of Artificial Intelligence (AI).

1.2.3 Partial Parsing or Full Parsing

Regardless of whether a symbolic or sub-symbolic approach is being used, there is a distinction between full and partial parsing. Full parsing means that every sentence must be completely analyzed from the beginning to the end. The output from full parsing is usually a parse tree saying what Part-Of-Speech (POS) each word has, how words are connected to one another in phrases, and how the phrases together make up the entire sentence. Quite often there will be more than one possible legal parse tree, and then the sentence must be disambiguated (possibly in a larger context) to find the one intended parse tree (with the right semantics). Another possibility is to simply list all legal parse trees without considering semantics. Partial parsing, on the other hand, means that the output is not a complete parse tree for the entire sentence. Instead it can be

smaller parse trees for specific phrases that are recognized in the sentence, or simply a POS-tag for each word, saying nothing about how they connect to each other.

1.2.4 Local or Global Grammars

The difference between local and global grammars is somewhat similar to the difference between partial and full parsing. With a global grammar, the dependencies between words far away from each other are modelled explicitly with complex high-level grammatical rules. In the local grammar approach [8], pattern-recognizing automata are built to deal with neighbouring word dependencies. Later these automata can be group into larger units and thereby implicitly solve the long range constraints.

1.2.5 Robust Parsers

Another criterion under which a parser is evaluated is whether it is robust or not. Robust in this sense means if the parser is able to deal with all reasonable inputs. All parsers are constructed with specific sentence constructions and words in mind, or they are trained (statistically) on a corpus of relevant and already correctly parsed/annotated sentences. However, the human language is so flexible that new and previously unseen constructs or names are bound to appear all the time. When a parser is able to deal in some intelligent manner also with all the examples that it was not specifically constructed or trained for, it is called a robust parser. Most full parsers are not robust, since they are built on the premises that all possible sentence constructs must be known in advance.

1.2.6 Corpus-Based Approaches

In both NLP and NLU many researchers are now trying different corpus-based approaches. That means that they take some collection of actual texts from the domain (e.g. Medline) as a starting point. Then, this text must be manually analyzed by experts in the domain (e.g. Biologists), and tagged by linguistic experts. This pre-processed text can then act as source for learning rules etc., or it can be used as a golden standard when testing parsers, saying exactly what the desired results are for this specific collection of texts.

1.3 Full Parsing

Searching **Medline** [39], **Cite-seer** [24] and **Google** [28], only one article that describes full-parsing of microbiological texts was found. This article is discussed in depth below, and then the remaining sections will describe various other methods that are used to achieve the goal of IE in the microbiological domain.

Yakushiji et al. [18] was the only paper found that describes the pure full parsing approach to biomedical texts. It is an early report on an experiment that the authors carried out to see if this approach can be used, even when the texts are more complex than e.g. newspaper texts. Their long term goal is to build an information extraction system that can extract specific facts from Medline abstracts. Their short-term experimental goal was to automatically extract 133 (already known) facts from 97 manually annotated test sentences.

The reason for trying full parsing is that current information retrieval and IE methods are not scalable enough. Today, extraction of a fact is done by syntactic (surface form) pattern matching against all possible ways of expressing that fact. That means that for every type of fact (relation) many handmade patterns are needed, and this technique is too expensive when the number of different relations gets bigger.

The Yakushiji et al. system is based on a general purpose (domain independent) parser. The parser transforms each sentence into an argument structure (AS). Each AS contains a verb as the title, the semantic subject and object(s) of the verb, and possibly adjective modifiers. The AS is a canonical structure, and that means that the *parser* has already taken care of all the variations that can occur in the text because of for example passivization and nominalization in the verbal phrases.

Next comes the domain specific part of the system. For each type of AS, a transformation rule (pattern matching) must be written, that converts the AS into a corresponding frame representation (FR). The FR is a possible end result of IE, and contains the semantics of the original verbal phrase. This technique scales better with large number of different relations, since the parser deals with the different syntactic ways of writing a verbal phrase, and only a few IE transformation rules must be written for each type of relation.

The article deals with three well-known problems of full parsing: Inefficiency, ambiguity and low coverage. These problems are partially solved with the use of pre- and post-processors. One pre-processor is the shallow parser. It introduces local constraints (a little stronger than Part-Of-Speech tagging) whenever possible in the text, and this increases the efficiency of the parser since obviously illegal (and computationally expensive) parse attempts can then be avoided. The other pre-processor is a term recognizer. It is not yet implemented, but it was simulated by hand-annotating the complex names in the sentences as units belonging to a

given class. This gave a 10-fold increase in parsing speed, and also reduced the coverage problem since failure to recognize a complex term is often the reason that the parse fails.

The results of the experiment are not extremely good (23% success rate), but they give hope that this method can work (67% success rate) when more pre-/post-processing techniques are applied. 23% of the facts were uniquely (correctly) extracted. 24% of the facts were extracted with more than 1 possible FR (ambiguity) and 20% of the facts were extractable (without modifiers) from the partial results of the failed parses.

1.4 Goals

The goal of Information Extraction (IE) in the medical domain is as follows: We need to automate the task of IE from biomedical papers, because there are simply too many new papers every day for the researchers to keep up with. On the way to solving this goal many sub-problems must first be solved. Most of these sub-problems have already been identified by others, for example in the review that is summarized in this section.

1.4.1 Other Reviews

Text-based knowledge discovery is discussed in the review by Mack & Hehenberger [11]. They identify several of the common goals for the search and mining of life-sciences documents. Both Mack and Hehenberger work for IBM, and in the article they naturally also present IBM's solutions to the tasks that they are discussing. They begin by stating that the main point of biomedical Knowledge Discovery (KD) is to create an interpretive context for biology researchers, and that text-mining is of great use when modelling complex biomedical structures and processes. Quite often an important part of the puzzle exists only as written text, in some publication somewhere. Currently many databases are being built to contain these facts in an organized way, and IE can help speeding up this work.

Mack & Hehenberger also point out that there is currently a shift from simple Information Retrieval (IR) to more advanced IE techniques. Theses techniques include both stochastic (statistic) and symbolic (rule-based) methods. Until only recently the IE community has been focusing mainly on extraction of named entities (e.g. protein and gene names) from the medical texts. Now there is a shift towards extracting concrete relations between these entities, and we are getting one step closer to the next goal of

building more complex structures (such as networks of connected facts). What is really needed is a standard way of describing these facts, so that the databases don't become as unstructured and inaccessible as the huge amount of free text was in the first place!

1.4.2 Tagging

No matter what method is used to analyze the text, the first sub-goal of NLU is usually to mark-up the single words in the text with tags (e.g. labels such as N for Noun, and V for Verb). One very popular tool for doing this is the Brill tagger [3], and it was used in many of the projects that are being reviewed in this article. The Brill tagger is a robust, statistical, transformation-based POS-tagger. Unlike other pure statistical approaches, this tagger tries to *learn* common sense *rules* about how the tags should be applied, based on a given correctly tagged example text. These rules can then be manually modified later, in order to optimize the performance of the tagger on specific texts.

1.5 Systems

In this section three different types of systems will be reviewed: Term recognizers, Relation discovery systems and Visualization systems.

1.5.1 Term Recognizers

Results from full-parsers are much better when some pre-parser can recognize and cluster long names/Noun Phrases (NP) in advance [18]. One such NP-recognizer is implemented in Bennett et al. [2]. Their approach is to use the Brill tagger [3] to assign POS-tags to all the words in the text, and then they build patterns to say which combinations of POS-tags are legal NPs. They have taken ideas and techniques from different commercial software systems and implemented them in a publicly available free system. In this way they save other researchers from having to make or buy their own NP-extractors, and thus they free resources that instead can be used for *further* research in the field of biomedical IE. The article also describes how the software must be run on a multiprocessor supercomputer with tapes as the input medium. Thanks to Moore's law [35], the same system can today effectively be run on a PC with a cluster of large hard drives.

In **"Contrast and Variability in Gene Names"**, 2002, Cohen et al. [4] found common patterns among gene names and symbols from the LocusLink Database [33]. Based on these regularities, they suggest four heuristics for clustering different variations of the same gene, protein or RNA names together: equivalence of vowel sequences, optional hyphens,

optional parenthesized material, and case insensitivity. The researchers ran their heuristics against Medline abstracts and, as expected, got better results than what is achieved with normal strict pattern matching. Unfortunately, manual examination showed that there were many false positives, i.e. names and symbols were clustered even if they were not related. Cohen et al. argue that discerning between documents on different organisms may improve the results. They are to research for more contrast features, which discerns similar name for different genes and more closely examine the false positives. This is an example of a project where no NLU is involved, but the system can still be interesting as a pre-parser for a NLU system.

Another approach to recognizing gene names and gene symbols in biomedical texts was investigated by Proux et al. in **"Detecting gene symbols and names in biological texts: A first step toward pertinent information extraction"** [14]. Proux et al. built a cascade of transducers to extract gene names from biological documents. The first transducer tokenizes the input before a probabilistic HMM part-of-speech tagger assigns categories to known words. This leaves most gene names and symbols in an undefined group. Two error-correcting steps clean the unknown words: First a biological dictionary removes common biological terms and adds common language words that potentially are gene names when occurring in Medline abstracts. Then, another error-recovery algorithm removes nucleotide and peptide sequences, components and special terms with the help of special sequence detection rules succeeded by suffix and prefix recognition. Before validating the results, the system performs a contextual analysis where it interprets any unknown word preceding "gene" as a gene name. Proux et al.'s system showed good results, but it only studied gene names from the fly Drosophila. This may introduce problems when generalizing the method since the Drosophila database, FlyBase, uses only standard gene symbols as opposed to the case of normal biological texts and genes from other organisms. Scaling will also be of concern when moving from small, directed circumstances to the amounts of data in e.g. Medline. Nevertheless, using statistical methods Proux et al. have introduced a gene name allocation method that works well within its domain.

The last term-recognizing system to be reviewed is the PROPER system [7]. Their approach is purely syntactic, and they claim that this is the only sensible approach since new terms are created quicker than the dictionaries can be updated. In their test they got 99% recall and 95% precision. They

have already identified one of the major sources of precision problems and they plan to get rid of that problem as future work.

1.5.2 Relation Discovery

The first example of relation extraction uses pattern-matching, and is from the article **"Robust Relational Parsing over Biomedical Literature: Extracting Inhibit Relations"** by Pustejovsky et al. [15]. It was presented on Pacific Symposium on Biocomputing 2002 (PSB02) [38], which is probably the most important conference for IE from biomedical texts (among other topics), since it includes many potential users (biologists) and the most significant recent papers in the field. The parser presented here is a robust, shallow, corpus-based parser. Relational parsing means that they extract information on the form X relates to Y. In this case the specific relations are all inhibiting relations, and the X and Y can be entities (genes and proteins) or processes (e.g. binding). Their results are much better than previously published results, with 90% precision and 57% recall plus 22% partial recall. Partial recall means that just X or Y, but not both, was extracted. The way they get these good results is by their use of cascades of Finite States Automata (FSA), more or less in the same way that is done with local grammars [8] in Unitex [36]. One important step in getting good results was to realize that nominal-based relations (Predicative Nouns) had to be dealt with separately from normal verbal-based relations. All this work is a part of the Medstract project [34], building on the old Acromed system.

Another interesting approach to doing relation extraction is presented by Park [13]. They use a parser with combinatory categorial grammar to parse the relatively complex biomedical sentences, and they combine this with the corpus-based approach. In the end they do a gold standard test, with 48% recall and 80% precision, and these numbers are better than any other previously published comparable attempts. The conclusion of the article more or less agrees with [18], in that full-parsing can be made to work, and it is worth the effort, because then we can extract more specific and meaningful facts from the abstracts. One example where full-parser usually performs better is anaphoric resolution, meaning the ability to recognize what is pointed to by terms such as "it".

1.5.3 Visualization

Visualization is another important area, because the biologists (end users) need to understand the information that is extracted from the biological texts. In the article **"A literature network of human genes for high-throughput analysis of gene expression"** [10], Jenssen et al. introduce a program called *PubGene*. It creates and visualizes an overview network of possibly related genes. The network is built on the assumption that gene

names co-occurring in Medline abstracts also have a related function or another relevant connection. The network is especially useful in Microarray experiments, because then many genes must be explored simultaneously. The methodology includes a database of gene names, a gene-to-article index, a gene-to-gene network, a gene network browser, and a gene expression and literature score. To handle the gene name problems the authors collected gene name variations from LocusLink [33], Human Gene Nomenclature Committee [30], the Genome Database [27] and GENATLAS [25]. The resulting gene identifier database contained 13712 different genes, and each became a node in the gene-to-gene network. Using the accumulated identifiers, the authors searched Medline and found 7512 co-occurring genes. Each co-occurrence linked two network nodes or added one to the weight of an existing link. The finished network allowed searches for individual nodes, resulting in a sub-network of the gene's closest neighbours, or an expression set from e.g. a Microarray experiment. The sub-networks of the searches indicate functional relations that the biologist should consider in her further work. Jenssen et al. proved their concept with a subset of well-known expressions. According to error analysis, most false positive errors stem from gene identifier problems, e.g. the gene names are too general.

The visualization of gene-interaction networks, e.g. as in [10], is very important for the biologists who are trying to understand what the role of a single gene is. Another field where visualization is very important is in the construction of local grammars. The idea behind local grammars is that you cannot write *general* rules about how *nouns* and *verbs* combine into phrases and sentences, because there are simply too many irregularities or exceptions. In the end, you really need an exhaustive list of specific rules for every single possible use of a given verb: Normally accepted complements (e.g. nouns), all legal adverbial phrases for the verb, idiomatic uses with their allowed complement structures, and so on. This is an enormous work, since there is more than 10^{50} ways to build a sentence with at most twenty words [8], and therefore it is very important with a good visualization tool so that all these rules can be built fast with a minimum of extra work. The kind of local grammars described here are implemented in the visualization system Unitex [36].

Other examples of systems that include some sort of visualization are described in [9, 12, 17, and 5]. These articles describe complete approaches, with all the necessary steps from plain texts via knowledge bases to actually useful systems for the end users. They are written in the early stages of IE from biomedical papers, and they are giving general pointers

and plans about what has to be done. It is also interesting that Internet is pointed out as a new kind of "Corpus" for IE systems to take advantage of, especially as databases such as Medline [39] become more accessible and structured.

1.6 Conclusion

There are mainly two current approaches to Information Extraction from Biomedical Texts. One is the search for golden language rules and good heuristics as presented in [18], and the other is the rather tedious work of collecting all necessary examples from maybe 10^{50} possible different 20-words sentences [8]. Almost all the articles criticise the *other* approach, namely the one that they are not using, but future systems will probably include techniques from both of these approaches, since they really just try to solve the same problem from two complementary sides.

2 Computational Linguistics

This chapter gives the background information and terminology that is needed to understand the rest of this thesis. It gives definitions of important linguistic and computational linguistic terms, and starts by giving a definition of the main term: "Computational Linguistics".

2.1 Introduction

Simply put, computational linguistics is the scientific study of language from a computational perspective. Computational linguists are interested in providing computational models of various kinds of linguistic phenomena. These models may be "knowledge-based" ("hand-crafted") or "data-driven" ("statistical" or "empirical"). Work in computational linguistics is in some cases motivated from a scientific perspective in that one is trying to provide a computational explanation for a particular linguistic or psycholinguistic phenomenon; and in other cases the motivation may be purely technological in that one wants to provide a working component of a speech or natural language system [17]

Computational linguistics is a pragmatic approach to the field of language. In the end we want to build a working system that solves a particular task. In the case of GeneTUC, this task can be the automatic generation of gene interaction databases or the construction of a "Tell and Ask" gene oracle.

2.2 Linguistic Terminology

An important part of analyzing a sentence is to tag each word with its correct Part-of-Speech (POS) tag. The challenge then is that many words are highly ambiguous, and different approaches must be used to find the correct tag. The definition and meaning of some none-trivial POS tags are given below.

2.2.1 Pronoun

Definition: A pronoun always stands in the place of a noun in the sentence. The pronouns can be tagged with information such as

Is it a Possessive (Poss) Pronoun?

The Person (1st, 2nd, 3rd)

The Number (Singular, Plural)

For third person singular pronouns; the gender: Masculine, Feminine or Neutral (m, f or n)

The Unitex tag for pronouns is “PRO”.

2.2.2 Determiner

Determiners are said to "mark" nouns. That is to say, you know a determiner will always be followed by a noun. Some categories of determiners are limited (there are only three articles, a handful of possessive pronouns, etc.), but the possessive nouns are as limitless as nouns themselves. This limited nature of most determiner categories, however, explains why determiners are grouped separately from adjectives even though both serve a modifying function. We can imagine that the language will never tire of inventing new adjectives; the determiners (except for those possessive nouns), on the other hand, are well established, and this class of words is not going to grow in number. These categories of determiners are as follows: the articles (an, a, the); possessive nouns (Joe's, the priest's, my mother's); possessive pronouns (his, your, their, whose, etc.); numbers (one, two, etc.); indefinite pronouns (few, more, each, every, either, all, both, some, any, etc.); and demonstrative pronouns. Notice that the possessive nouns differ from the other determiners in that they, themselves, are often accompanied by other determiners: "my mother's rug," "the priest’s collar," "a dog's life" [44].

The Unitex tag for determiners is “DET”.

2.2.3 Adverb

Adverbs are words that modify

- A verb (He drove slowly. — How did he drive?)
- An adjective (He drove a very fast car. — How fast was his car?)
- Another adverb (She moved quite slowly down the aisle. — How slowly did she move?)

Adverbs often tell when, where, why, or under what conditions something happens or happened. Adverbs frequently end in *-ly*; however, many words and phrases not ending in *-ly* serve an adverbial function and an *-ly* ending is not a guarantee that a word is an adverb [45].

One of the highly ambiguous words is "that", and in a few rare occurrences it should also actually be tagged as adverb. One example is "It was that easy!"

The Unitex tag for adverbs is "ADV".

2.2.4 Conjunctions

A conjunction is a word that connects (conjoins) parts of a sentence. Conjunctions are split into two classes: coordinating and subordinating conjunctions [46]. The coordinating conjunctions are used to put two sentences of equal importance together, while a subordinating conjunction promotes one of the two sentences as being more important.

There are seven coordinating conjunctions: "And, but, or, yet, for, nor, so".

There are several subordinating conjunctions, and the most common ones are: "after, although, as, as if, as long as, as though, because, before, even if, even though, if, if only, in order that, now that, once, rather than, since, so that, than, that, though, till, unless, until, when, whenever, where, whereas, wherever, while."

The Unitex tag for all conjunctions is "CONJ"

2.2.5 Pre-determiners

Pre-determiners are small words (e.g. prepositions) that can stand directly before the main determiner, like "about" in the following example: I have bought about 2000 candles.

The Unitex tag for pre-determiners is "PRED".

2.2.6 Particle

Particles often occur in *phrasal verbs*. Phrasal verbs consist of a verb and another word or phrase, usually a preposition. The word that is joined with a verb in this construction should then be tagged as a *particle*.

The most common particle is the infinitive particle "to". A golden rule for disambiguation is that when "to" is found directly in front of a verb in infinite form, it should always be tagged as a "particle".

The Unitex tag for particles is "PART".

2.3 Computational Linguistic Terminology

In the following sub sections some important computational linguistic terms will be given: Ontology, Dictionary, Multi-Word Unit (MWU) and different word-counts.

2.3.1 Ontology

Ontology is defined as follows in WordNet [42]: "the metaphysical study of the nature of being and existence". Ontology is a structured description of what we know about "the world" that we are interested in modelling.

The "correct" classification of Nouns (and other words) into ontologies is tricky. The decisions to be made are largely dependent on what the end use of the ontology is. Choices must be made about the granularity and about how much ambiguity that should be allowed.

In the biomedical texts acronyms are being heavily used, and two different acronyms can often refer to the same entity. Therefore, a strategy should be selected so that all acronyms are represented in a uniform way. One important point is to decide what the main entry in the dictionary should be. The problem with storing acronyms as main entries is that one acronym often has several different meanings, and thus extra ambiguity problems are introduced. The problem with storing the expanded (full) forms as main entries in the dictionary is that one acronym can often be expanded in many different ways where the difference is very small. E.g. different authors have different styles, but they still refer to the same entity. In this case, all the possible full forms must be stored in the dictionary, perhaps with a common reference to a unique identifier (e.g. the "right" or standardized full form).

Another approach that was tested in this project was to build a small grammar for each acronym that has several different full forms. Such small "dictionary grammars" are called *lexicon grammars* in the Unitex terminology. Figure 1 shows a lexicon grammar that will recognize all occurrences referring to the CREB protein. The bottom line in this graph is not really a part of the *name* of the CREB protein, but it is still included because it is a unique remark that describes only this CREB protein. In addition, there are also other ways to talk about the CREB protein, so the graph is not complete (yet), but in the Unitex approach the idea is to keep adding new entries to the graphs until no more new entries are discovered. Then the graph will be "complete enough" for our purposes.

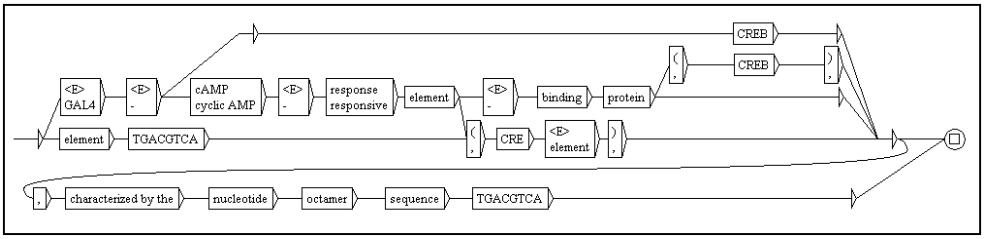


Figure 1. Lexicon Grammar for CRE

Synonyms raise similar problems as the entities that have multiple full form names do. When two words are truly "identical" (within the ontology) they should then be linked in some way, for example with the help of a unique concept-id number or similar techniques. By marking the word entries like that, some semantics will be brought into the dictionary. Normally a dictionary should contain only syntactic and morphologic information, but if such simple semantic information is needed by the system at a later stage anyway, it is easiest to store this information during the dictionary building. An example is while importing the gene name lists, all the entries can automatically receive the tag "+Gene".

2.3.1.1 TUC Ontology

The ontology in TUC is, like the WordNet ontology, built as a heterarchy. That means that each node can have multiple parent nodes in addition to having multiple children nodes as in a normal hierarchy. One constraint that is imposed in the TUC ontology is that all nodes must have the node "Thing" as their single highest ancestor in the network. That means that all concepts are ultimately classified as "Things", and "Thing" is the most general concept in the TUC ontology. The TUC Ontology is partly based on the WordNet ontology (see below).

2.3.1.2 WordNet Ontology

The WordNet ontology is a little more general and comprehensive, and it contains more concepts than the TUC ontology. With a few exceptions, the WordNet is a superset of the GeneTUC ontology. The main difference is that WordNet have several different top nodes, while TUC requires all concepts to be "Things" ultimately. Since these ontologies are so similar, it should be possible to automatically update the TUC ontology based on changes in WordNet Ontology. This could save a lot of work, since classifying unknown words and storing them in the TUC ontology (files: "semantic.pl" and "facts.pl") is always a necessary first step in the TUC approach, and somewhat time demanding. The creation of a good ontology is also necessary using the Unitex approach, so it would be well worth the

effort trying to make a good interface between WordNet and the other systems. WordNet already has an application programming interface (API, e.g. for java) to ease this work.

Another advantage of fully integrating WordNet with GeneTUC is that WordNet contains glosses for every concept (see Figure 2). That makes it easier for others (e.g. biologists) to do the classification of new unknown words into existing concepts, or new ones when needed.

One challenge during the integration work will be that some WordNet concepts have quite complex names, compared to the single word terms in GeneTUC. Several different TUC concepts can sometimes be seen clustered together as one long comma-separated WordNet concept. One possible solution to this "unification problem" is to use only the words before the first comma in the WordNet entry as the unique identifier for that (sub-) concept in GeneTUC.

salt -- (a compound formed by replacing hydrogen in an acid by a metal (or a radical that acts like a metal))
=> compound, chemical compound –
((chemistry) a substance formed by chemical union of two or more elements or ingredients in definite proportion by weight)
=> substance, matter -- (that which has mass and occupies space; "an atom is the smallest indivisible unit of matter")
=> entity, physical thing -- (that which is perceived or known or inferred to have its own physical existence (living or nonliving))

Figure 2. WordNet definition of Salt

2.3.2 Dictionaries and MWUs

The work with dictionaries is somewhat overlapping with the ontology-work, because both are dealing with the fact that all words need to be known by the system, before any meaningful processing can be done. The creation of a dictionary can be almost as tricky as the ontology work because new words are created all the time, and especially in the biomedical domain. This is especially true in the naming of new genes and proteins.

There is also the question of what granularity the dictionary should have. A phenomenon called Multi-Word Units (MWUs) deals with the fact that quite often a group of words only has the right meaning as a group, and not as single words. One example is "Bananas *as well as* apples" (Ref Sentence 44 and 58). In this sentence the three words "as well as" just take the place of the single word "and" (or maybe "and also"), and therefore

they should also be tagged as one MWU (namely conjunction). In many current systems "as well as" would be tagged as for example "PREP ADV PREP", but this is not a correct POS sequence between two sentences, and it gives no hints that we are semantically dealing with a conjunction. In Unitex this can be solved by using the DELACF dictionary format to store "as well as" on one line as a MWU, and tag it as conjunction.

2.3.3 Word Counting

Unitex lists some basic statistics every time a new text is pre-parsed, and every time patterns are located in the text using *regular expressions* or *search graphs*. To understand this statistics, it is helpful to know how the word count is done. Words are counted in these three different ways:

Running Words: Everything between two word separators (normally space) is counted as a word, regardless of whether that particular word has already been counted.

Unique Words: Only count the first occurrence of every word. This gives the number of *different words used in the text*

Frequency Count: Tells you how many times every given word is used.

2.4 Biological Terminology

This section will describe some of the particular language constructs and problems that were encountered during the work with texts from the Medline database.

2.4.1 Protein

Proteins are the agents/actors of interest. See Figure 3 and Figure 4 for the definition.

protein -- any of a large group of nitrogenous organic compounds that are essential constituents of living cells; consist of polymers of amino acids; essential in the diet of animals for growth and for repair of tissues; can be obtained from meat and eggs and milk and legumes; "a diet high in protein"

Figure 3. WordNet definition (with gloss) for Protein

```
Protein
    => Macromolecule, super-molecule
      1=> molecule
        => Unit, building block
          => Entity, physical thing
      2=> organic compound
        => Compound, chemical compound
          => Substance, matter
            => Entity, physical thing
```

Figure 4. WordNet ontology entry for Protein

2.4.2 Genes

For every protein there are one or more genes that "code for it". Quite often the gene and the protein will then have the same name. When such ambiguities arise in the semantic tagging of the text, the gene-tag should be selected, for the sake of consistency. Exceptions should be made for example when the sentence contains the word "protein" directly after the protein/gene name.

2.4.3 Enzymes

Here is the definition of enzyme from Kimball's Biology Pages [22]:

Enzymes are catalysts. Most are proteins. (A few ribonucleoprotein enzymes have been discovered and, for some of these, the catalytic activity is in the RNA part rather than the protein part; Link to discussion of these **ribozymes**.) Enzymes bind temporarily to one or more of the reactants of the reaction they catalyze. In doing so, they lower the amount of **activation energy** needed and thus speed up the reaction.

Many enzyme names are encountered during classification of unknown words in Medline texts, and to speed up the process it helps with a few good heuristics. One such heuristic is that all names with "ase"-ending are enzymes (and therefore protein) names.

2.4.4 Latin: Cis and Trans

Dealing with medical texts, one always encounters many Latin words that are not so common in other texts. Two of these rare words will be discussed in this sub-section, namely cis and trans.

In the activate sentences (See Appendix B, Sentence 6) "cis-acting elements" are mentioned. Cis is a Latin prefix that means something like "on this side". Its Latin counterpart is "trans", which means "on the other

side". The MIT Bio-pages [23] has a more medically precise definition of the two terms. Figure 5 shows how this works on the molecular level:

"We can use these techniques to see if a DNA sequence can act from afar on another DNA sequence. If it can, then it is a diffusible protein. These sites are called **trans-acting** sites, since they act from afar. If the site cannot act from afar, then it is a DNA binding site that needs to be near other DNA sites (such as coding sequences) in order to function. These sites are called **cis-acting sites**, since they need to be next to other DNA to work."

The Unitex semantic tag for prefix is PFX.

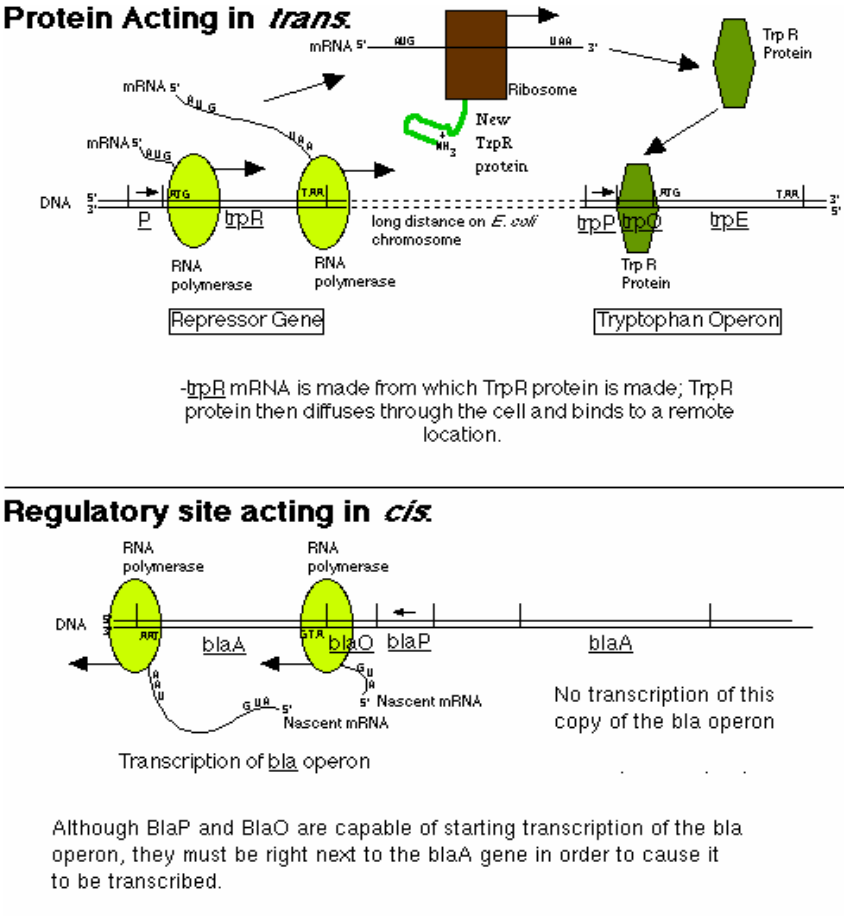


Figure 5. Cis- and trans-acting proteins. Courtesy of the MIT Bio-pages

2.4.5 Greek Letters

Another typical feature of biomedical texts is that they contain many Greek letters that have no standard plain-text coding format. Many of the proper nouns (e.g. protein names) contain Greek letters, and various types of sub and superscripts. This problem is encountered when PS, PDF or

Word files are converted into other formats, and especially into plain text format.

The activate-sentences (Appendix B) used in this experiment were collected from various PDF files from the internet (via Medline), but when these PDF files were downloaded as Word (Doc) files, a special coding was used for Greek letters. Instead of α, the text "small alpha, Greek" was inserted, instead of β "small beta, Greek" and so on. This creates some extra problems in the tokenisation of the text, since e.g. "Greeksmall" is an unknown word in our language. In this experiment the original texts was changed manually to solve this problem, but when dealing with large amounts of text, some automatic way should be found to solve these problems.

3 Unitex Tutorial

This chapter describes the Unitex system, and can act as a preliminary English translation of the manual. It will explain the basic functionality in Unitex in the form of a step-by-step tutorial convenient for new users.

3.1 Introduction

Most of the "tricks" in this chapter had to be learned by trial and error, since the manual only exists in French and a partial Portuguese translation [37]. Writing an English translation of the manual is one of top priorities of the Unitex team, but there is no official deadline given for this task yet. The main programmer of Unitex, Sébastien Paumier [40], has been very helpful and answers all questions about the program rapidly by email. He also came to LMU Munich (to the CIS institute) at the beginning of this Diploma work (November) to give a guest lecture on Unitex. The main points from that introduction will be summarized in this chapter.

Sébastien Paumier worked for the LADL institute in France [32], where both the INTEX and the Unitex systems were developed. Unitex is more or less just a GNU General Public License (GPL) version of INTEX, but it still lacks some of the semantic disambiguation support that INTEX gives.

3.2 Installation

Unitex [41] can be downloaded and installed together with the Java Runtime Environment (JRE) version 1.4.1. The installation takes only a few minutes, and any standard PC should be adequate in terms of system requirements.

Version 1.1 has been available since the beginning of February 2003. It was released as a beta version, but it was already running stable and has been used in this project since then without any particular problems. The biggest change from version 1.0 is that the graph editor GUI has been

improved, and a few general bugs have been removed. All the data files can also be transferred between the two different versions without any problems (Tested for text, graph and dictionary files).

3.3 Text Format

All the texts that shall be used by Unitex must be stored in Unicode format, and with the little endian byte order (standard for Intel-type processors). The text format conversion must be done before the file can be opened in UniTex, and can be done by most semi advanced text editors, including Word and TextPad (but not Notepad). There is also a support program in the Unitex package that will do this conversion. This program is located in the "Unitex\App" directory and is called "asc2uni". The input file must be ASCII-coded, and then a Unicode output file will be created.

3.4 Pre-processing and Lexical Analysis of the Text

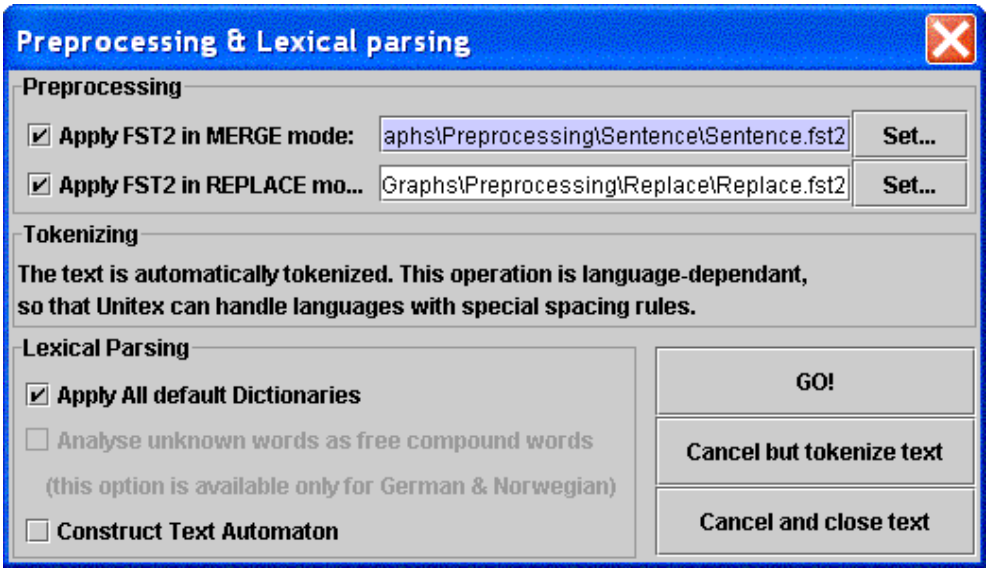


Figure 6. After opening a new text

The first time a Unicode text-file is opened in UniTex, one is given the option to pre-process the text. During pre-processing, sentence delimiters ({S}) are inserted into the text. At the same time, lexical analysis can be done ("Apply All default Dictionaries"); meaning that every word is tagged with all the possible tags it can have according to the dictionaries.

Unitex has two already compiled English system dictionaries: One for single words (delas.bin or delaf.bin), and one for compound terms

(delac.bin or delacf.bin). Other self-made dictionaries (e.g. gene and protein names) can also be used during the pre-processing phase. The difference between DELAS/DELAC and DELAF/DELACF is that DELAF dictionaries contain inflected forms of all the words. Normally a DELAF can be automatically created from DELAS/DELAC with a few good inflection rules.

In connection with the lexical analysis it is also possible to construct Finite State Text (FST) automatons for the sentences, by using the bottom left checkbox in Figure 6: "Construct Text Automaton". FST (see section 3.6) is a good idea, because it will give you a feeling about how ambiguous the sentences are, based on the current dictionaries. FST automatons can also be constructed later by choosing "Construct FST-text" from the "Text" menu in Unitex.

When pre-processing and lexical analysis is done, some common files are automatically created:

DLF	Contains all possible simple word dictionary entries matching the text
DLC	Contains all the possible compound lexical entries that were recognized
ERR	Contains all the simple words that were not found in any dictionary
Tokens	Contains an unsorted list of different tokens from the text
Tok_by_freq	Contains different tokens sorted by frequency of use
Tok_by_alph	Contains all different tokens sorted alphabetically

3.5 Graphs

In this section, a simple description of the syntax that is used to read and write graphs is given. The description is largely derived from Chapter 4 in the Unitex Manual [37].

+	Separate (OR) different word (or sequences of words) choices in a box
-	Logical "And Not" with the next grammatical/semantic code in the <X> forms. For example, <N-hum> recognizes all *nouns* that do not have the *human* mark
/	Separate box input from box output
<X>	Match all the words with ground (lemma) form X, or with X as a grammatical/semantic code. Examples: <be> recognizes be, are, am, is, etc… <N> recognizes any simple or compound word that has the noun mark
()	Group elements (for regular expressions)

*	Kleene Closure: Match none ore more instances of the previous (grouped) term

Figure 7. Graph writing syntax

<E>	Empty string
<MOT>	Token with only letters
<MIN>	Token with only lower case letters
<MAJ>	Token with only upper case letters
<PRE>	Token with only letters (like MOT), but starting with an upper case letter
<DIC>	matches any simple or compound word in the dictionary
<SDIC>	matches only simple words from the dictionary
<CDIC>	matches only complex words from the dictionary
<NB>	Token with only contiguous numbers (no spaces within)
<PNC>	Punctuation (; , ! ? :) is only available during pre-processing.
<^>	Matches a new-line (\n) and is only available during pre-processing.
#	forbids the presence of a space at the given position. For example, <MOT>#-#<MOT> will recognize "tam-tam" but not "tam – tam".

Figure 8. Special lemma-forms (from [37], chapter 4.3.1)

While looking at a graph with subgraphs, a specific subgraph can be opened by pressing the <alt> key while clicking on the subgraph's named box representation.

A graph can be used to recognize/locate given fragments (local grammars) in a text, and it can be applied with different output modes: Ignore, Merge or Replace.

In ignore mode the original text is not changed, and all the matches are given in a separate list. This is called a concordance list and contains the context for each of the matches. The concordance list is also hypertext marked-up, so it is easy to jump to the right place in the original text, and to the appropriate FST-text (see section 3.6) if it has already been made and is currently visible in another open window.

When searching for a pattern in merge mode, the output from a graph will be inserted in the original text, just *before* the starting point of whatever the graph matched.

Finally in replace mode, everything that matches the graph is removed from the input text, and only the output from the graph is left in that place.

3.5.1 Variables

In Unitex graphs, variables can be used to produce just the output that is wanted. In Figure 9 two variables (*$activator* and *$activated*) is being used to transform the extracted facts from natural language into a very general predicate logic form.

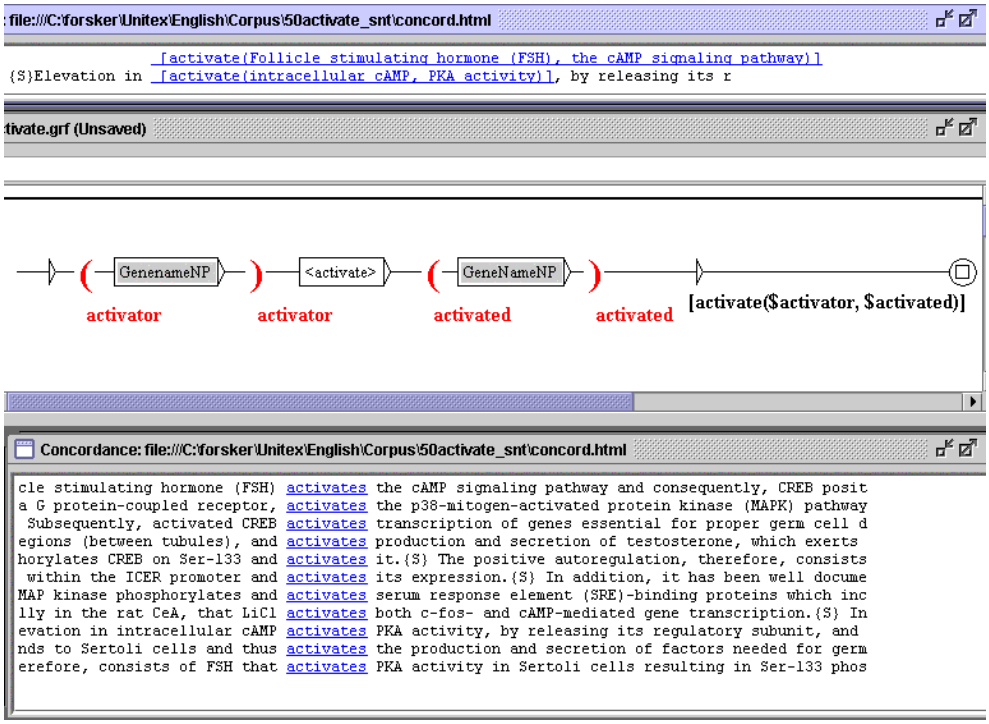


Figure 9. Variables in Unitex

The middle window in this figure shows a graph were the sub-graph "GeneNameNP" is called two times. The text that matches the sub-graph the first time is stored in the *$activator* variable, and the "*Gene Noun-Phrase*" that matches the sub-graph the second time is store in the variable *$activated*. At the end is just an empty box that produces the desired output in the form of a logic predicate (*activate*) with two arguments, and the entire output is placed in angle brackets, in order to make it easier to separate it from the remaining text later.

The top window shows a concordance structure with the results of applying this simple graph in *replace* mode, and the bottom window show a concordance structure for all occurrences of the word "activates" found in the 59 sentences.

3.6 FST-text

FST-text is a special kind of graphs. These can be automatically created (see section 3.4), there is one FST-graph per input sentence, and they

represent every possible parse of the sentence given the entries in the inflected dictionaries.

FST is an excellent starting point for disambiguating the sentences manually, since one only has to delete the boxes that do not fit. When an FST sentence is modified, the changes will be stored in the working directory as a temporary file called "sentenceX" (X is the number of the current sentence). When the "Rebuild FST-text" button is pushed, the "sentenceX" files are used to update the *more* permanent file "text.fst2". If you later choose "build FST-text" again from the menu, also this file will be overwritten.

3.7 Dictionaries

This subsection will give a brief overview of the different dictionary formats and explain how they are used in Unitex.

3.7.1 Dictionary Format and Syntax

There are two main types of dictionaries in the Unitex system: Inflected (DELAF/DELACF) and uninflected (DELAS/DELAC). These can be further divided into simple-form (DELAF or DELAS) and complex-form (DELAC or DELACF) dictionaries. The only extra feature in complex form dictionaries compared to the simple ones, is that they accept MWUs, i.e. the complex forms can have the space character in them.

Figure 10 shows the format of the *un*inflected and Figure 11 shows the format of the inflected dictionaries.

```
"Word surface form, POS + Semantic: Morphology: Syntactic: Inflection"

Diacyl-glycerol, N+Glycerol: s
Down regulate, V
```

Figure 10. Format for uninflected (DELAS/DELAC) dictionaries

```
 "Word surface form, lemma form. POS + Semantic: Morphology/Syntactic/Inflection"

Diacyl-glycerol, .N+Glycerol: s
Down regulates, down regulate.V+P3s
```

Figure 11. Format for inflected (DELAF/DELACF) dictionaries

For word entries without any morphological information the last ":" should be dropped, and when the lemma form in a complex dictionary is equal to the word (surface) form the lemma place should be left empty.

Not all characters can be used freely when writing the dictionaries. Figure 12 gives a list of the special characters. These characters must be protected by a \ (backslash) when they are used as "themselves" as a part of a dictionary entry. If it is possible to avoid entering these characters into a dictionary, it would save a lot of trouble later, because the dictionary, FST-text and graph modules of Unitex do not deal with escaped characters 100% consistently.

/	Comment (Chapter 3, page 24 in manual, examples)
=	Hyphen or Space (Can be used to represent both forms with just one entry, page 23)
,	between token, lemma
.	Between lemma. Grammatical info
:	Between grammatical info (lexical info) and Inflexion information (info about inflected forms)
+	Between parts of lexical info
\ \	Used as folder separator in graphs on Windows systems

Figure 12. Special dictionary characters

3.7.2 Dictionary Tags

Here the different standard tag types are explained with examples. Also, a few self-defined tags from the dictionaries Genes and ProtHum are used. The Gene and Protein dictionaries were created by converting the GeneTUC files "genebase.pl" and "protbase.pl" into Unitex format.

A	Adjective
ADV	Adverb
CONJC	Conjunction, Coordinating
CONJS	Conjunction, Subordinating
DET	Determiner
INTJ	Interjection
N	Noun
PART	Particle, including infinitive particle
PRED	Pre-determiner (I bought *about* 200 balloons)
PREP	Preposition
PRO	Pronoun
V	Verbs
X	Words that cannot stand alone, without prefix, suffix etc (in *situ*)
XI	Parts of MWUs (et, al.)

Figure 13. Word class dictionary tags

3.7.2.1 Morphological tags

This list includes all syntactic and morphological tags, and the information they can contain

:m	Masculine
:f	Feminine
:n	Neutral
:s	Singular (Number)
:p	Plural (Number)
:1	First (Person)
:2	Second (Person)
:3	Third (Person)
:P	Present Indicative
:I	Imperfect Indicative
:S	Present Subjunctive
:T	Imperfect Subjunctive
:Y	Present Imperative
:C	Present Conditional
:J	Past (simple)
:W	Infinite
:G	Present Participle (Gerundive)
:K	Past Participle
:F	Future

Figure 14. Morphological dictionary tags

3.7.2.2 Semantic tags

Here are the semantic tags that are described in the Manual [37]

+z1	Common word
+z2	uncommon word
+z3	Very rare word
+Abst	Abstract
+Anl	Animal
+AnlColl	Animal Group
+Conc	Concrete
+ConcColl	Concrete Group
+Hum	Human
+HumColl	Human Group
+t	Transitive Verb
+i	Intransitive Verb

Figure 15. Semantic dictionary tags

There also are also a few other system-defined semantic tags that are not described in the manual:

```
+A   Adjectively used adverb (ADV that can be used as A)
+DA      A or DA (as in "poorly understood")
+2X A+2X (in situ). X always marks words that should not stand alone

For PRO and DET:
+Dadj    Det Adjective
+Ddem   Det Demonstrative
+Nomin Nominative (for PRO)
+Pdem    Pronoun Demonstrative
+Poss3ns          Possessive 3.person neutral single (for PRO and DET)
+PR Province (Monaco et al., here it is a Personal Name!)
```

Figure 16. "Undefined" semantic dictionary tags

And here are two self-made semantic tags, particular for Microbiology-linguistics.

```
+Gene   Gene
+ProtHum          Human Protein
```

Figure 17. Self-defined semantic dictionary tags

3.7.3 Dictionary Update

After the lexical resources (dictionaries) have been applied to the text, all unknown words will be stored in the file "ERR". In order to be able to do intelligent parsing of the text, all these words must first be added to a dictionary.

When new words have been added to a dictionary file (e.g. name.dic), this file must then be compiled into a .bin file, so that it can be used together with the system dictionaries the next time lexical parsing is done. The main reason for this precompiling is that binary compiled graphs give a much higher parsing speed than one can get when interpreting the graphs on-the-fly. This is more or less in analogy with the fact that compiled (e.g. C++) programs run much quicker than interpreted (e.g. Java) programs. The compiling of new dictionaries can be done by opening the Unicode "file.dic" from the "DELA" menu in Unitex. Then, from the same menu one has the choice to check the format or to compress the dictionary into a finite state automaton (FST). It is also possible to use the program "Compress" in the "Unitex\App" folder to do this compiling. The result in both cases is that a file called "name.bin" will be created.

To use the new compiled dictionaries every time a new text is being pre-processed, they must first be added as default lexical resources. This can be done from the "Text" menu, but only when an arbitrary text is already open. So the first step is to open the text, and choose to skip the preprocessing. Then, choose from the "Text" menu "Apply Lexical Resources" (see Figure 18). In this window, all the .bin files in the

"Unitex\English\Dela" folder are listed. New dictionaries can be selected in addition to the old ones, by pressing the <ctrl> key while clicking the new names with the mouse pointer. After all the standard dictionaries have been selected, it is important to click the "Set Default" button, so that the changes are made permanent and all the selected dictionaries will be used next time preprocessing and lexical parsing is being done.

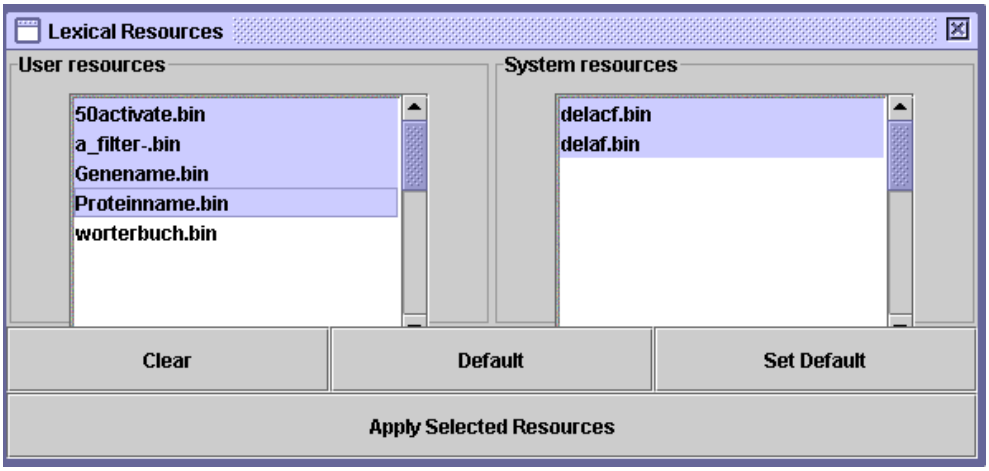


Figure 18. Menu: Text->Apply Lexical Resources

The selected dictionaries can be directly (re-) applied from the window in Figure 18, or when the "Set Default" button has been clicked, from the Preprocessing and Lexical Parsing window, see Figure 6.

3.7.3.1 Creating a context for "unknown words"

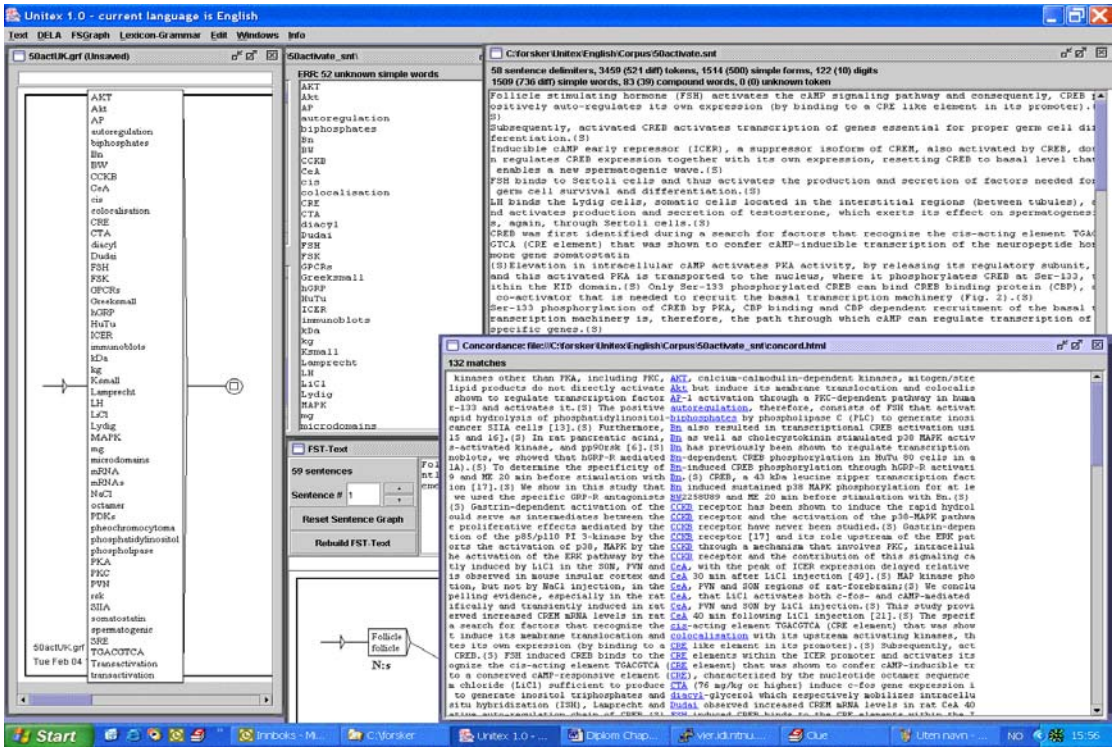


Figure 19. Creating a context for "unknown words"

A simple way to speed up the process of classifying unknown words is by making a graph that contains one box with all the unknown words in it (See Figure 19, left side). When this graph is stored, it can later be applied for locating patterns in the text, and for building concordance structures, as it was done in Figure 19, bottom right side. This is very useful, since the unknown words are then highlighted and display together with their *context* (approximately the 12 nearest words) in the text. This can save a lot of time since unknown words are often declared (explicit or implicit) in the text where they are first used, and by using these definitions one can save valuable time that would otherwise have to be spent searching the Internet or other dictionary sources.

When searching for the 52 unknown words in the text, it was discovered that they constitute almost 4% of the total text. That means that every 26th (running) word is an unknown.

3.7.4 Filter Dictionaries

Filter dictionaries are dictionaries with a name ending with a – (minus) sign before the ".DIC" ending. They can be used for example to solve a very particular problem raised by the fact that "a" and "as" are treated as

nouns. First of all, there is a valid reason for treating “a” and “as” as *nouns*, for example in the sentence:

“I tell you that bazaar takes three as and bizarre just one a.”

Figure 20. A "bizarre" sentence

But even though “a” and “as” can be nouns in this particular sentence, they will usually not be in the kind of micro-biological sentences of this project. The *noun* interpretation is not just unlikely, but it actually causes some problems for example when locating all indefinite articles; it should be possible to searched for the lemma form <a> to matched both “a” and “an”, since these two indefinite articles have “a” as their lemma form. Unfortunately, also “as” has the lemma form “a” (because of the *many* “a”-s interpretation), and this causes a lot of noise in out search.

Therefore, to avoid this interpretation of a, you can use a filter dictionary: Put all the interpretations you want for a and as in a DELAF, for example:

```
a,.DET
as,.PREP
```

Figure 21. Filter dictionary: a_filter-.dic

Give the file a name with a - sign before .DIC, such as “filter-.DIC”.

Then, compile your dictionary and select it to be applied in the default dictionary selection (menu Text>Apply lexical resources). This will filter for “a” and “as” only your given solution and you won't have to worry anymore about the noun interpretation for a and as.

3.7.5 Complex Dictionary Terms

Some of the gene names are so complex that it would be better to build simple NP graphs for them, than to list all possibilities in a dictionary. This can be compared to the way that the stock exchange index-names were modelled in “the construction of local grammars” [8] by Maurice Gross. See Figure 1 for a biomedical example.

3.8 Disambiguation

When a text is opened in Unitex the first time an FST (see 3.6) can be automatically constructed for all the sentences. This FST will then contain

graphs showing every possible parse of every sentence given the current dictionaries. It is then possible to manually modify the FST-text, e.g. to do disambiguation on it by removing un-semantic paths from the graph. When a sentence is changed manually, the result will be stored in a temporary file in the text's working directory (e.g. …/English/50activate_sen/sentenceX, where X is the sentence number).

During the disambiguation work one can also choose to give semantic codes to important words, for example by tagging all genes and proteins with genes and protein tags. One problem with such semantic tagging is that a word (e.g. polymerase) can belong to several different semantic classes, and it is not practical to give every word a complete and very long list of semantic tags (e.g. Enzyme, Protein, Molecule and so on). In GeneTUC this problem is solved by building an ontology (a semantic network), then giving every word (e.g. polymerase) the most detailed description possible in this network (e.g. Enzyme), and by having a rule saying that it then also belongs to all the parent nodes above this description (e.g. Protein, Molecule, Thing). It is an open question how such rules could or should be implemented in Unitex.

4 GeneTUC

In order to appreciate the results made with Unitex and local grammars, it is important to think about how they can be used later. In our case, Unitex's local grammars could be used to do effective pre-processing on the biomedical texts, before they are parsed by GeneTUC. This is a good idea because the strengths of Unitex (Fast processing, and a graphical interface) matches the weaknesses of GeneTUC (slow parsing, and no graphical interface). The reason for not doing *all* the work in Unitex is that GeneTUC already implements the entire framework for a tell-and-ask system, and the grammar is much more expressive than the regular automata in Unitex.

This chapter will not give many details about GeneTUC, but the points most relevant to the work with Unitex will be discussed below. For more details the reader is referred to the last project report covering the GeneTUC system [16]. It includes a tutorial, and explains the basic need for pre-parsing due to punctuation, Greek letters and other such problems that Unitex can solve much easier.

4.1 Introduction

The end goal of GeneTUC is to build a question answering system about biomedical facts. Biomedical text, more often than not, contains Greek letters, punctuation in the middle of words, chemical formulae and so on. This constitutes a problem for GeneTUC, because it is not made to deal with anything else than standard ASCII-letters. This means one of two things: Either GeneTUC has to be changed in order to deal with all these new problems, or the input texts have to be modified in order to fit GeneTUC's requirements.

Work is being done with GeneTUC to make it deal intelligently with some simple punctuation (e.g. comma), but as long as the system remains ASCII-based, it can never deal successfully with the Greek letter problem.

Therefore, the approach with pre-parsing should also be explored, and Unitex is ideal for such "strange-letters" parsing, because it is Unicode-based.

4.2 Greek Letters

By now there exists several different ways that the authors of the biomedical texts deals with Greek letters. Some simply use "a, b, g" etc instead of "α, β, γ" and so on, and some spell it out like "alfa, beta, gamma" etc. Others use the actual Greek letters in their writing, but then these letters are later changed into more or less cryptic representations (e.g. "small beta, Greek" or "small chi, Greek") in the process of transferring documents between different formats such as LATEX, Word, PDF, PS and so on.

One solution to this problem is to use Unitex to build local grammars that would recognize all these different representations and transform them into a standard ASCII-representation that GeneTUC could process further. For example, "PI3Ksmall beta, Greek" and "PI3Kβ" could both be replaced by "PI3Kbeta" or "PI3Kb". This is very easy to do in Unitex, because the local grammars are built in the form of *transducers*, which means they can produce *output* in addition to reading input like normal *automata*. There are three ways that the local grammars (transducers) can be applied to the text: Normal, Merge and Replace modes. When the Greek-letter term-recognizing graphs are applied in Replace mode, we get exactly what is needed by GeneTUC, namely a text file without Greek letters, and with only well defined protein and gene name IDs. Of course, this file must then be translated back to ASCII from Unicode, but that is a trivial task. For example, there is a program called Uni2Asc in the Unitex\App folder that does this conversion.

4.3 Dictionaries / Ontologies

Dictionaries are just as important in GeneTUC as they are in Unitex. In addition GeneTUC is dependent on its ontology (implemented in "semantic.pl" for the "*ako*" relations, and in "facts.pl" for the leaf nodes "*isa*" relations). The construction of ontologies is at least as time-consuming as the construction of dictionaries, and therefore new ways to automate this work should be found.

For example, it should be possible to use WordNet to automatically classify unknown words into the GeneTUC ontology system, but there are

a few problems. First of all, the categories in GeneTUC and WordNet are far from 100% equal, even though the GeneTUC ontology is strongly "inspired" by WordNet. That means that GeneTUC must be adapted to fit the existing WordNet categories, before automation can take place. Second, also the *formats* of the ontologies are quite different between the two systems. In GeneTUC two elementary relations (inheritance and attributes) can be represented: "A Kind Of" or "Is A" (E.g. salt ako substance, and NaCl isa salt) and "Has A" (DNA has_a region). Since GeneTUC is a running system, the relation can only hold between well defined concept classes. This is not the case with WordNet (see Figure 2. WordNet definition of Salt), where the ontology entries are written as free-text. Still, with a few transformations, the simplified structure can be implemented in GeneTUC. The glosses can easily be stripped away, since they are already put in parentheses (watch out for nested parentheses!), and the comma-separated lists of definitions can be implemented as multiple *ako* relations in GeneTUC. The only problem left then is the fact that it might be meaningless to have both *compound* and *chemical compound* as classes, but in the end that will depend on the actual use of the system.

5 Methods

This chapter describes the work done with the Unitex system, and how it compares to GeneTUC methodology.

5.1 Introduction

The current end goal for a working biomedical parsing system, whether it is GeneTUC, Unitex-based, or any other parser, is to do automatic information extraction (IE) from the medical abstracts or full texts. In this project we are particularly interested in gene *activation*, and we want to extract information such as what gene/protein is the activator, what gene/protein is being activated, how reliable are the extracted facts, and what extra conditions must be satisfied. In order to start somewhere, a micro-biologist was asked to find around 50 sentences that contain facts about activation. Most of these sentences also contain the actual word "activate" in some form, but there are also a few sentences that use other words (e.g. X *confer transcription* of Y).

After local grammars have been built for the given "activate-sentences", a test will be run on a biological reference corpus, to see how general and applicable the graphs are.

5.2 Text Sources

This section describes the different sources that were used to acquire the text to parse. Most of the text is from the Medline abstracts database [39]. The first source contains an entire abstract that was previously used to train GeneTUC. The second source contains "random" single sentences selected by a biologist with the criterion that they should all contain facts about activation of a gene, protein or hormone. The third source is a 19.000 sentences large biological reference corpus that will be used for testing the

finished local grammars in the end. This is the same text that was used to test GeneTUC earlier (*abs2.txt*).

5.2.1 The Medline Abstract

A Medline abstract about gastrin and CCK was used for preliminary testing, to see the number of unknown words, and how ambiguous the sentences are. It can be seen in Appendix A. This is the same abstract that was used to test GeneTUC in 2002, so it should be possible to compare the results, and the amount of work that is needed to successfully "understand" a text using either of the two different systems.

5.2.2 59 Activate-Sentences

Astrid Lægreid found more than 50 sentences describing the activation of different genes. Actually, the sentences are not only about *gene* activation. They also contain facts about *protein* and *hormone* activations. For the sake of testing the Unitex methodology, it is not so important whether they are genes, proteins or hormones. Genes and proteins quite often have the same names anyway, since a protein is usually made from one or more corresponding genes. Hormones actually have slightly different names, but the sentences about hormone activation have the same form and context as the gene/protein activation sentences. That means that we can merge the gene, protein and hormone dictionaries, and just treat all these names as subjects or objects of the *activation* relation (see Appendix D, Name-graph)

5.2.3 Micro-Biological Reference Corpus

The first micro-biological reference text was the same as that used to test GeneTUC earlier. It contained around 18.000 sentences, which is a little more than 5MB when it is stored in the Unicode format.

Later a reference corpus with about 25.000 *tagged* tokens was acquired from "Centrum for Informations und Sprachverarbeitung" (CIS) at LMU in Munich. This reference corpus was also originally extracted from Medline, and it was used as a cross reference and aid during the classification of different medical words (primarily names) in this project. A program could be built that automatically classifies or suggest classification of words based on what tags they are given in such an *already tagged* reference corpus. This would save a lot of work, since every entity name would then

only have to be manually processed one time. Right now, every researcher always has to start from scratch, and often ends up solving problems that have already been solved by others.

After all the graphs for the 59 activate sentences were finished, a test was run on "*abs2.txt*" to see how applicable the graphs were. The test results are given in 6.4.

5.3 Preliminary Work with a CCK Abstract

As a way of getting familiarized with Unitex, some preliminary work was done with a small familiar text sample (See Appendix A). The CCK abstract has previously been successfully parsed by GeneTUC, so resources such as the classification of previously "unknown words" were already available in the GeneTUC system. See Figure 22 for a step-by-step explanation of the transformations that were needed to import the gene names into Unitex via the DELA dictionary format.

```
Manually:
1) Delete header and footer
2) Delete all lines beginning with % (Comment = Removed entries)
3) Delete all gene('
4) Delete all ').
5) Substitute \n with  ,.N+Gene\n

Step 2-5 can be done with Perl Regular Expressions, like this:
2)    s/^%.*//;
3-5)  s/\('(.*)'\)/$1,.N+Gene/;
```

Figure 22. Conversion from genes.pl (GeneTUC) to genes.dic (Unitex)

After the gene, protein and substance names had been imported to give GeneTUC and Unitex a "common microbiological platform", the following steps were taken: The CCK abstract text was stored in Unicode format, and then it was opened with Unitex. It was pre-processed in Unitex, with lexical parsing, but only using the standard dictionaries. 12 unknown words were found among a total of 137 words.

5.3.1 Dictionary Update

Since all the words from the CCK abstract had already been classified in GeneTUC, the possibility of automatic importation was explored. As many as 10 of the unknown words were gene/protein/substance names or identifiers, and should therefore be found in one of the GeneTUC files "*genebase.pl*", "*protbase.pl*" or "*substance.pl*" in the "genes" or "database" folders. There are also two files, "*genecmpl.pl*" and "*protcmpl.pl*", that

contains "Multi Word Units" (MWUs, or full forms) that mapp to each of these identifiers. All these files were auto generated from a nomenclature resource on the Internet [30].

The converted gene, protein and substance names dictionary were applied as default lexical resources, and text pre-processing were done again, but this time with all dictionaries applied (see 3.7.3). Unfortunately, most of the unknown identifiers were not found in new dictionary files. The reason for this is that all the "handcrafted" additions (during the GeneTUC work with this abstract in 2002) were put directly into the system file "*facts.pl*". This practice should be changed in later versions of GeneTUC, to ensure that the system remains as modular as possible, and to allow for better "cooperation" with other systems. For now, the missing identifiers were manually copied from "*facts.pl*" to Unitex, and stored in the dictionaries that were automatically extracted from GeneTUC (genes, proteins and substances), see Figure 22.

5.3.2 Disambiguation

After all the unknown words had been put into the right dictionaries, "Finite State Transducers" (FSTs) were (automatically) constructed for the 11 sentences. This turned out to be a good starting point for the disambiguation task, since all possible parses are then represented in the graph, and after removing all the invalid nodes from the graph, only the right path (the disambiguated sentence) will be left. There were a few words where the correct dictionary entry did not exist, and therefore none of the suggested paths through the graph were the correct one. This was then solved by adding the correct entry into the graph manually, to avoid having to do re-preprocessing of the text, since that would delete the disambiguation work that had already been done. In addition, the word-entry was added to the appropriate dictionary so that it would be taken into account when preprocessing was done the next time.

5.4 Work on the "Activate" Sentences

This section describes the experiments that were done with the Unitex system on the collection of Medline sentences about gene/protein/hormone activation listed in Appendix B.

5.4.1 Preliminary Work

The "activate-sentences" were first stored in Unicode text format, and pre-processed with Unitex, just like the CCK abstract. All standard dictionaries were used, including the imported gene, protein and substance dictionaries from GeneTUC (see 5.3.1). FST automata were constructed to see how ambiguous each sentence was. Some facts about the sentences are given below:

- The 59 sentences consisted of 1514 words (27 words per sentence), but just 500 *different* words.
- Of the 500 words, 52 were unknown words (not already in the dictionaries). See Figure 24 for a list of these words and the sources that were utilized to find their right semantic classes etc.
- Different forms of "activate" (activates, activated, activating) occurred 23 times in the text, "activation" occurred 25 times, and activator 1 time. That means that at least 10 of the activate-sentences did actually not contain any use of the word activate.
- The original text was slightly modified, but only in order to avoid the Greek-letters errors.

5.4.2 Sentence Delimiters

During pre-processing, the 59 sentences were (wrongly) split into 61 sentences. Three extra sentences were made because of periods in abbreviations (e.g. "Fig. 3"), and two separate sentences were joined because the second sentence was started by the name "p53" (with a small "p"). There were two occurrences where a semicolon was (correctly?) interpreted as a sentence-delimiter.

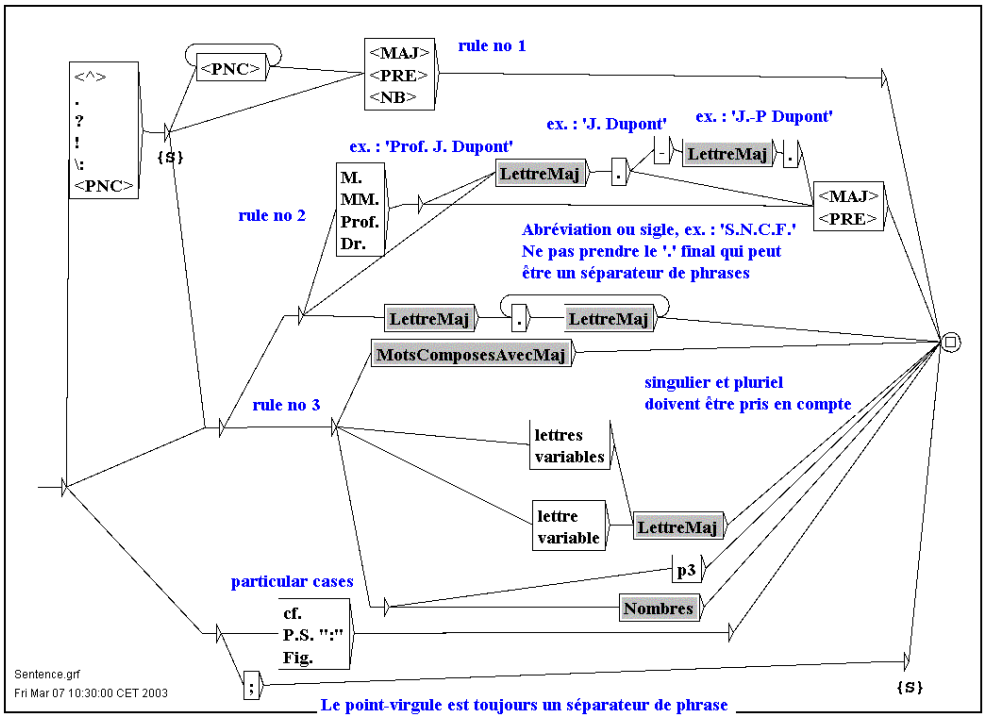


Figure 23. Modified Sentence.grf to correctly detect sentence boundaries

The way to solve these delimiting problems is by modifying the Sentence.grf file (in the Graphs/Preprocessing/Sentence folder) before doing pre-processing again. "Fig." was added as a "*cas particuliers*" (particular case) in this graph (see Figure 23). This means that a period does not count as a sentence delimiter when it follows directly after the word "Fig". "p53" were also added to the graph, as a legal sentence starter, even though it is written with a small p. These changes will not have any effect on the original text before the Sentence.grf file is recompiled into an fst2 file and pre-processing is done again. After this was done on the original "50activate.txt" file, 59 sentences were correctly recognized.

It is worth noting that every time pre-processing of a text is done, all the files that are stored in the corresponding text_snt folder are deleted, and new ones are created. That means that no useful external data should be store in the text_snt folders!

5.4.3 Dictionary Update

The 52 unknown words (except for a few obvious errors in the original text) were added in a new dictionary file in the "Dela" directory. This file

was later compiled into a .bin file, so that it could be used together with the system dictionaries during lexical parsing. Figure 24 shows the sources that were used in order to classify the unknown words with correct semantic tags. See Figure 26 for the actual semantic classifications.

Word	Source
AKT	http://www.sigmaaldrich.com/Area_of_Interest/Life_Science/Cell_Signaling/Pathway_Slides_and_Charts/Akt_Signaling.html
Akt	See AKT
AP-1	http://journals.endocrinology.org/joe/169/joe1690447.htm http://www.biochemj.org/bj/360/0599/3600599.pdf
Autoregulation	http://www.cogsci.princeton.edu/cgi-bin/webwn1.7.1?stage=2&word=autoregulation&posnumber=1&searchtypenumber=-2&senses=&showglosses=1
Bn	http://www.nature.com/cgi-taf/DynaPage.taf?file=/mp/journal/v7/n1/full/4001974a.html
BW2258U89	http://www.alzet.com/bibliography/bib_pages/ia.htm
CCKB	http://www.uni-mainz.de/FB/Chemie/Biochemie/Fahrenholz/abstr_16.html
CeA	http://www.biomeda.com/site/cat/K052/specsheet.html
Cis	Astrid Lægreid
Colocalisation	WordNet
CRE	GeneTUC
CTA	http://www.conditionedtasteaversion.net/
Diacyl-Glycerol	WordNet: http://www.cogsci.princeton.edu/cgi-bin/webwn1.7.1?stage=2&word=glycerol&posnumber=1&searchtypenumber=-2&senses=&showglosses=1
Dudai	Original text
FSH	Original text, WordNet
FSK	GeneTUC
GPCRs	GeneTUC. www.gpcr.org/7tm/
hGRP-R	In the text by Qu
HuTu 80	http://www.biotech.ist.unige.it/cldb/cl1780.html http://www.aphis.usda.gov/ppq/manuals/pdf_files/APM%20in%20PDF/APM.pdf (cell "part of" organism) Definition in the text by Qu
ICER	Definition in the text by Don
Immunoblots	http://www.ndif.org/Terms/immunoblots.html. Useful: Automatic dictionary creator.
kDa	From Astrid Lægreid
Kg	Kilogram
Ksmall	Error because of bad Greek letter handling
Lamprecht	Name, Author
LH	From original text
LiCl	Chloride = salt
Lydig	Used as Adjective
MAPK	From original text
Mg	Common sense: milligram
Microdomains	Obvious: Domain
mRNA	Obvious: RNA

mRNAs	See mRNA
NaCl	Chloride = salt
Octamer	From original text
PDKs	From original text
Pheochromocytoma	Used as Adjective
Phosphatidylinositol	http://www.lipid.co.uk/infores/Lipids/pi/
Phosphatidylinositol-biphosphates	http://acer.gen.tcd.ie/cgi-bin/khwolfe/gene.pl?name=PIP2&junk=no
Phospholipase	http://www.biochem.ucl.ac.uk/bsm/enzymes/ec3/ec01/ec04/ec0003/ (Good Enzyme Classification reference!)
PKA	From text
PKC	From text
PVN	From text
pp90rsk	http://www.colorado.edu/Chemistry/directory.dir/faculty.dir/biochem.dir/ahn.dir/ahnres.html
SIIA	Used as Adjective
Small beta,Greek	ERROR: Because of conversion from PDF to Word-format. Greek letters were translated into text-strings such as this one. See **2.4.5**.
Somatostatin	http://arbl.cvmbs.colostate.edu/hbooks/pathphys/endocrine/otherendo/somatostatin.html
Spermatogenic	Used as Adjective
SRE	From original text
TGACGTCA	Obviously a DNA sequence
Transactivation	Obvious=Activation

Figure 24. Information sources for unknown words

5.4.4 Greek Letter Problem

To deal with the Greek-letters problem the original text was changed in the following way: "Small beta, Greek" was replaced by "b" and "Small gamma, Greek" was replaced by "g". For a discussion about this problem see section 4.2.

5.4.5 Re-Preparsing

After the original text and dictionaries had been changed to account for unknown words and Greek letters, pre-parsing had to be done again. During pre-parsing, all the preliminary work that was already done was deleted from the working directory "50activate_sen". Of course, it was much faster doing the job the second time, since the new dictionary was then already in place.

5.4.6 Disambiguation

Working with the automatically constructed FST-Text (graphs), it is possible to quickly disambiguate the input sentences manually (see section 3.8). When the FST-text is modified in this way, each modified sentence will be stored in the working directory (e.g. 50activate_sen). NB: Be sure to copy this files out of the working directory to a safe place before you reset, rebuild or pre-process your graphs or text, because then **all** files in the working directory will be deleted.

Several problems were encountered during the disambiguation work. These include things such as MWUs that are not listed in the dictionaries, single words that are highly ambiguous and cases where it is not really clear which tag that should actually be used. One example is sentence number 3 (see Appendix B): "ICER down regulates CREB expression". The reason for this is that "down regulates" is really a compound verb, and should be represented as that in the DELACF-dictionary. This addition and others like it were done in the third iteration of pre-processing the text. Ideally, the text should only have been pre-processed one time, but because of "learning-by-doing", it is sometimes necessary to start all over again.

5.4.7 Building Graphs

After the text was finally disambiguated, the next step was to build graphs that would match all the "activate"-facts in the text. This is tedious work and a lot of choices have to be made about modularity, recursion/iteration and semantics. The entire process is described in this section.

5.4.7.1 Modularity

A main goal during the graph building is to build separate units that can be reused later. That means that the graphs will be modified and "added on to" later, but never in such a way that they no longer work in their original settings. This requires some clever choices about which sentences should be merged in one graph, and which ones should be modelled in different graphs. In the beginning this will require some trial-and-error attempts, but after a while one gets a better intuition about the problem, and some golden heuristics can be made. For example, sentences that seem to have similar parse trees should be kept in the same graph.

Another criterion for how graphs should be split and merged is according to what semantic relations they express. This will be discussed in the semantics sub-section below.

5.4.7.2 Recursion

Since the microbiological language is usually very complex and contains quite long sentences, it is necessary to use many sub-graphs in order to fit an entire sentence into one graph. The graph size can be somewhat increased to account for this problem, but if the graphs get much bigger than A4 paper size, the risk of losing track of the details increases dramatically. Of course, too much recursion can also cause problems, so one has to strike a balance between graph size, and recursion depth.

Another problem with recursion is the risk of ending up with non-terminating automata. Recursion is accepted in Unitex, because there is a mechanism to handle it. This mechanism is probably just a cut-off limit, so after a certain amount of nesting, the next recursive call is simply not executed.

An alternative to recursion is iterations, meaning that a loop is formed in a graph. This has more or less the same advantages and disadvantages as recursion. Iterations allow for longer sentences to be put into one graph, without making too many sub-graphs. This is especially true for sentences of the form "X activates A, B, C, D and E". The A-E entities in this sentence usually have a lot in common, and should therefore be modelled in the same graph. Then, this sub-graph can be iterated, just by inserting "," or "and" between each iteration. This works well in order to *recognize* sentences of this form, but if the graphs were to be "run in reverse" to *produce* all such sentences, two problems will appear. First, the production will produce sentences of the form "X activates A, A, A, A and B". Second, it will go on doing that forever, without ever terminating. In other words, a choice must be made about what is more important, easy functional graph-building contra time-consuming but 100% versus graph-building. Again, this really depends on what the end goal and the intended use of the system is.

5.4.7.3 Semantics

The end goal of the system is to extract meaningful semantic relations from the graphs, or actually from the sentences that the graphs recognize. It is important to keep this in mind as the graphs are being built. For example, when one graph is getting too big and must be split into multiple separate graphs or recursive/iterative sub-graphs, it is important to think about things such as *anaphoric resolution*.

"Activated PKA localizes to the nucleus where it phosphorylates CREB on Ser-133 and activates it"

Figure 25. Anaphoric sentence

In a (quite short) example sentence as in Figure 25, there are two occurrences of the word "it". In order to determine what "it" means, anaphoric resolution must be done. This is usually quite easy for humans to do, and it is more or less obvious to us that the first "it" refers to "(activated) PKA" and the second "it" refers to "CREB". For the computer, on the other hand, explicit rules must be made about how anaphoric resolution should be done, and these rules are usually not straightforward. In any case, it will be much easier to make such rules when "it" occurs in the same graph as the antecedent (what is being pointed to).

When the antecedent occurs in the same graph as the anaphoric reference, Unitex can use variables to solve the problem of anaphoric resolution (see 3.5.1). The box that will match the antecedent can be identified already when constructing the graph and the value that matches this box can be stored in a variable. Then, the box that matches the anaphoric reference could be identified, and the anaphoric reference could be replaced by the value in the antecedent-variable. This would give a disambiguated sentence in terms of anaphoric resolution, because all references have been replaced by semantically meaningful antecedents.

If the sentence in Figure 25 was to be split between two different graphs, at the "and"-conjunction for example, then anaphoric resolution would be a little bit harder. First of all, because it is a technical challenge to pass variable values between different graphs, and second, because it would be much harder for the graph creator to keep track of what is going on. Some of the modularity of the system would also be lost in the process of splitting this sentence between different graphs, because there would then be a connection between the two new graphs. The second graph would then always expect the first graph to store an antecedent value in a given variable, and this does not agree with the "modularity thinking", which says that all graphs should be separate, reusable modules.

6 Results and Discussion

This chapter will discuss the results that were acquired with the Unitex system when it was applied to the different Medline text samples.

6.1 Introduction

The graphs that are being built must be general enough to also accept sentences that are not *explicitly* programmed. That means that if we have training examples such as "X activates A", "X activates B" and "X activates D", then the very similar sentence "X activates C" should also be recognized by the system. This means that we have to introduce abstract graphs such as "X activates <Noun>", but if too many such abstractions are introduced, the system will end up also recognizing incorrect or "false" sentences.

The results from the tests and the lessons learned during the different work phases will be summarized in this chapter.

6.2 The CCK Abstract

Here are the results from the preliminary work on the CCK abstract summarized. During the importation of gene, protein and substance names and identifiers from the GeneTUC system, a flaw was discovered. These names should all reside in files that were automatically generated and updated based on different internet ontology resources. The problem was that several new entity names have later been added to the system, and they have then been put in other files. Most of the new entries can be found in the "*facts.pl*" file where they are mixed with all other entities (such as face, plasma, Værnes and politiet) making it harder to extract only gene, protein and substances names.

In the end, only the entities that were listed in the gene, protein and substance file were imported into Unitex, and the 12 unknown words in the

CCK abstract were then added manually to these dictionaries. In the future, the protein, gene and substance entries in the GeneTUC file "*facts.pl*" should be extracted and put into the right separate files, in order to keep GeneTUC as modular as possible, and making it easier to incorporate external sources.

It was much quicker building grammars for the 11 sentences using Unitex than it was with the TUC-grammar in the last GeneTUC project. This can be because the text was already well-known when it was processed in Unitex. Another reason can be the fact that in the GeneTUC project, considerations had to be made all the time about how to keep the whole existing system running as good as before, when changing it to add new rules. This is probably not just a problem of GeneTUC, but of any system that grows so big.

6.3 "Activate" Sentences

This section will summarize and discuss the results from the work on the 59 activate-sentences.

6.3.1 Sentence Boundary Detection

The 59 sentences were first split into 61 sentences, because of three periods in abbreviations (e.g. "Fig. 3") and one sentence starting by the name "p53" (with a small "p"). This means that the boundary detection graph (*Sentence.grf*) was 93% (55/59) accurate on these sample sentences. After two simple updates to the sentence delimiting preprocessing graph (see 5.4.2), it was working 100%. Semicolons were treated as sentence-delimiters, but that is really just an arbitrary choice that has to be made.

6.3.2 Dictionary Update

The 52 unknown words were added to a separate dictionary (*50activate.dic* in the *Dela* folder), according to Figure 26. This dictionary is listed in DELAC format in Appendix C. Figure 27 shows how the semantic classes of these entries fit into the GeneTUC (and WordNet) ontology. When searching for the 52 unknown words in the activate-sentences, it was discovered that they constitute almost 4% of the total text. That means that every 26th running word is an unknown.

The following tables explain what synonyms and hypernyms were found for each word.
Classes in parentheses () means that they are not part of the semantic network in GeneTUC, but they are listed in the WordNet ontology
Asterisk (*) before a term means that it is a complex term, consisting of multiple words or numbers

Word	**Synonyms**	**(GeneTUC) Class**
AKT	PKB	Kinase
Akt	PKB	Kinase
* AP-1	Activator Protein 1	Protein
Autoregulation		Process
* phosphatidylinositol-biphosphates	PIP2 PI(4,5)P2	Protein
Bn	Bombesin, GRP	Substance
BW2258U89		Antagonist
CCKB	cholecystokinin B (often as ADJ before receptor)	Receptor
CeA	Carcinoembryonic Antigen	Antigen
Cis	On this side	Prefix (PFX)
Colocalisation		Activity
CRE	CAMP Responsive Element	Element
CTA	Conditioned taste aversion	Aversion
* diacyl-glycerol	DAG	(Glycerol)
Dudai	Name	Author
Calcium	From periodic table	Element
FSH	Follicle stimulating hormone	Hormone
FSK	Forskolin	Substance
GPCRs	G protein-coupled receptors	Receptor
* hGRP-R	human gastrin-releasing peptide receptor	Receptor
HuTu 80 (ADJ before cells)	duodenal cancer	(Cell_line)
ICER	Inducible cAMP early repressor	Protein
Immunoblots		Technique
kDa		Measure
Kg	Kilogram	Gram (!)
Ksmall	ERROR	N/A
Lamprecht	Name	Author
LH	luteinizing hormone	Hormone
LiCl	lithium chloride/salt	Salt
Lydig	Often ADJ before cells	
MAPK	mitogen-activated protein kinase	Kinase
Mg	Milligram	Gram (!)
Microdomain		Domain
mRNA	Messenger RNA	RNA
mRNAs	See mRNA	See mRNA
NaCl	Natrium Chloride	Salt
Octamer	8-mer	Sequence
PDKs	Phosphoinositide dependent kinases	Kinase
Pheochromocytoma		Cell_line
Phosphatidylinositol		Lipid
Phospholipase		Enzyme

PKA	Protein Kinase A	Kinase
PKC	Protein Kinase C	Kinase
PVN	Para ventricular nucleus (of the hypothalamus)	Nucleus
* Pp90rsk	pp90 ribosomal S6 kinase	Kinase
SIIA	human gastric cancer	ADJ to cell
Small beta, Greek	Error	N/A
Somatostatin		Hormone / Gene
Spermatogenic	ADJ->wave, process, stage	
SRE	serum response element	Element
Testosterone		Hormone
TGACGTCA	DNA-sequence	Sequence
Transactivation	"other side" activation	Activation

Figure 26. Unknown words semantic classifications

(GeneTUC) Class	Hypernyms (WordNet entries in parentheses)
Activation	Activity, Thing
Activity	Thing
Antagonist	(Drug), Agent, Thing
Antigen	Substance, Thing
Author	Person, Animate, Agent, Thing
Aversion	(Dislike), Feeling, Abstract
Cell_line	Cell, thing
Domain	Region, surface, place, thing
Element	Part, Thing
Enzyme	protein, component, part, thing
Glycerol	(Alcohol), Liquid, (Fluid, Substance), Mass, Thing
Gram	Measure, Thing
Hormone	(secretion), Thing
Hormone / Gene	(Secretion / Agent), Thing
Kinase	Enzyme, protein, component, part, thing
Lipid	Substance, Thing
Measure	Thing
Nucleus	Place, Thing
Process	Activity, Thing
Protein	Component, Part, Thing
Receptor	Protein, Component, Part, Thing
RNA	Agent, Thing
Salt	(Compound), Substance, Thing
See mRNA	See mRNA
Sequence	Set, Thing
Substance	Thing
Technique	Science, theory, abstract, thing

Figure 27. GeneTUC (and WordNet) ontology extract

6.3.3 Disambiguation

All the sentences were disambiguated and automatically stored in files called sentenceX, where X is the current sentence number. During this work, a lot of "meaningless" dictionary entries created a bit of extra work. One of the strange cases was that "and" is classified as a verb. This means that every time "and" is used (and that happens in almost all the medical sentences), there will always be an extra verb-box in the graph. Each such extra box doubles the amount of possible paths through a sentence graph, and also requires an extra amount of time during the manual disambiguation process. So far, there have been no sentences where "and" is actually used as a verb. Therefore, it would probably be a good idea not to classify it as a verb until this is really needed. This can be accomplished by applying filter dictionaries (see 3.7.4) similar to that done to avoid tagging "a" and "as" as Nouns.

It turned out later in the project that it is not really necessary to disambiguate the training text in order to build local grammars, because when the local grammar graphs are applied, e.g. to locate patterns in a text, the disambiguated version of the text is not taken into account. This means that when a search is done for all verbs (<V>), "and" will also be listed, even though it is no longer marked as a verb in the disambiguated FST automaton. Only the untagged text and *all* matching dictionary entries are used in order to find such pattern matches. So to avoid un-meaningful results such "as" matching the *indefinite article* lemma form <a> or "and" matching <V>, the changes must be done using (filter-) dictionaries, and not by using FST disambiguation.

Even though disambiguation of the sentences was not necessary, it was still a good thing to do, since it shed some light on the complexity and some common problems of the training sentences. Two such problems will be described below, with some examples.

6.3.3.1 Multi Word Units (MWUs)

During the disambiguation process, it became apparent that some of the single words could not be meaningfully tagged at all. One such example is the phrase "as well as". This phrase is obviously used as a conjunction, but when trying to tag each word, one ends up with something like "<PREP> <ADV> <PREP>" which contains no clue that we are dealing with a conjunction. One could also imagine tagging the phrase as "<PREP> <ADV> <CONJ>" but that would mean that "as" is a conjunction, which is not really true. Other examples of phrases that should be added as MWUs to the dictionaries or modelled using *lexicon grammars* (see below) are given in Figure 28. The sentence numbers in this figure refer to the number given to each sentence in Appendix B.

Sentence#: MWU (POS)	
6: Cis-acting	(Adjective)
19: cAMP-activated	(Adjective)
38: Phospholipase C	(Noun)
40: Fig. 1A	(Noun)
41: p85/p110 PI 3-kinase	(Noun)
44: as well as	(Conjunction)
47: PI3K	(Noun)
50, 53: ser133	(Noun)
51: Bn-induced	(Adjective)
51: GRP-R	(Noun)
54: FSH induced	(Adjective)
54: Stress-activated	(Adjective)
56: GAL4-CREB	(Noun)
57: PC12	(Noun)

Figure 28. Sample MWUs

"cAMP-activated", "Mitogen-activated" and "stress-activated" belong to a class of special MWUs. The size of this class has the same infinite nature as the class of *personal nouns*. New names are being created all the time. Therefore, instead of trying to put all these MWUs directly into the dictionaries, a (lexicon grammar) rule should be made, and this rule should state that all *nouns* with the semantic tag "+activator" could stay in the place of X in the following MWU *adjective*: "X-activated". This is easy to accomplish with graph building in Unitex.

6.3.3.2 Gene/Protein Ambiguities

The ambiguity between genes and proteins is a particularly tricky ambiguity (see 2.4.2). So far in the disambiguation work the distinction between these to groups has not been very firm. However, as more and more facts are extracted, this distinction could be used, for example, to limit the number of hits from a database search. Therefore, some firm rules should be used later to make this distinction correctly. The golden heuristic that was used in this project was that "gene" was chosen as the correct tag for a named entity when it was listed as both gene and protein in the dictionaries. Exceptions were made when it was obvious from the context that the name was referring to a protein.

6.3.3.3 Disambiguation Heuristics

A few other "arbitrary" choices had to be made during the disambiguation work, and to keep the results consistent it is necessary to make the *same* choice every time. Therefore, golden heuristics were written, every time such a general choice had to be made. Two examples of such heuristics are given here.

The first example is the choice between preposition (PREP) and particle (PART) tags. This applies most often to the word "to" which is the infinite verb particle: When the word and the following phrase can be removed without breaking the *completeness* of the sentence, then the word should always be tagged as a *preposition* in a prepositional phrase (also called a complement, which means it is not a necessary part to form a whole sentence). When the word is standing directly before a verb in infinite form, the tag should always be *particle*.

The second problem is when dealing with MWUs. Unitex will as always suggest all alternatives, meaning the words can be tagged separately with separate tags, or together with one MWU tag. In these cases the MWU tag would normally be preferred, but sometimes the first word of the MWU is connected with a hyphen to the previous word in the sentence. Then, closer inspection is needed, but usually separate tags for all the words will be preferred, because the hyphen only *connects* semantically to the first word, and not the entire MWU.

6.3.4 Semantic Problems

During the first graph construction work with the sentences, problems were encountered regarding how to split the sentences into modular parts, to make them fit into the graph representation. Later, the challenge was to find out what semantic output the different sentences were really supposed to produce. The reason for this is that some of the sentences are very technical, and not always easy to understand for a non-microbiologist. A few of the semantic questions and the answers from our micro-biologist are listed in Figure 29, with reference to the corresponding sentence number in Appendix B.

Sentence# Question
Answer

6 This sentence does not use the word "activate" in any form. What is actually being activated by what? Which word signals that activation is going on?

Answer: "CREB activates transcription". This means that the sentence does not really fit the criteria that it should be about gene or protein activation, since "transcription" is a cellular *process*.

7 The first part is ok. Second part: Is "phosphorylates" equal to "activates"?

Answer: "PKA phosphorylates CREB". Proteins (such as CREB) are often *activated* by phosphorylation, **but** there are also many examples where proteins are being *de-activated* by phosphorylation.

8 **Without** the word "activate" (See #6). What is being activated? Is "Recruit the transcription machinery" equal to "activate" or is "phosphorylated CREB" equal to "activated CREB"?

Answer: phosphorylated CREB = activated CREB!

9 **Without** the word "activate". What is being activated, and by what?

Answer: "cAMP activates PKA", "PKA activates CREB" and "CREB activates transcription"

10 "Activate it". What is "it"? (Is "it" Ser-133?)

Answer: "PKA activates CREB".
"It" is probably CREB.

13 What is being activated, and by what?

Answer: "FSH activates ICER isoform of CREM"

14 What is being activated, and by what?

Answer: "FSH activates ICER"

17 The question in all the remaining examples is basically always: "*What is being activated, and by what?*" so only the answers are given below…

Answer: "LiCl activates c-fos (transcription)" and "LiCl activates ICER (transcription)

18

Answer: LiCl activates c-fos (expression)

19

Answer: "Lithium (Chloride) activates c-fos (expression)", "Lithium activates cAMP signalling pathway (PKA, actually)", "cAMP signalling pathway (PKA, actually) activates CREB" and "CREB activates c-fos (expression)"

20

Answer: "LiCl activate CREM (expression)"

21

Answer: "LiCl activates CREM (expression), is implied by the sentence"

22

Answer: "LiCl activates ICER"

24

Answer: "LiCl activates c-fos" and "LiCl activates ICER" (CREM gene, actually)

25

Answer: Answer: "LiCl activates c-fos" and "LiCl activates ICER" (CREM gene, actually)

26

Answer: "LiCl activates c-Fos"

27

Answer: "LiCl activates c-fos"

29

Answer: "LiCl activates MAP kinase"

31
Answer: "LiCl activates ICER (CREM gene, actually)"
32
Answer: "LiCl activates c-fos" and "LiCl activates ICER" (CREM gene, actually)
37 Extracted fact: "In particular, activation of the p38-MAPK pathway by gastrin." Is the condition "…have never been studied" important?
Answer: "Gastrin activates p38-MAPK (signalling) pathway"
55
Answer: Bn activates transcription factor AP-1"

Figure 29. Semantic challenges

In Figure 29 many of the sentences express two or more different activation facts, and often without using the word "activate" at all. In order to simplify the information extraction somewhat, it would be a good idea to focus *only* on gene and protein interaction, because then there would usually be only one fact to extract per sentence. The other activation-facts are about processes, secondary messengers and so on (for example, sentence 9).

Here are some other conclusions that are based on the answers to the general semantic questions above:

- Transcription (of a gene) = (gene) expression (Example #17 and #18)
- "Activation" is just as *interesting* as "de-activation"
- The same fact is usually expressed more than one time in the same abstract/article, sometimes as many as 10 (See sentences #17-#32)!

When asking the questions in Figure 29 the hope was to get simple facts back, and then implement these facts as output into the already existing graphs. It turned out, however, that many of the *answers* were even more complicated than the original sentence itself. This gives reasons to believe that simple "black and white" answers might not be what the biologists are looking for in the first place. In the future, we really need to work much closer together with the biologists, and try to understand what their needs are, before implementing a system to try and solve these needs.

6.4 Biomedical Corpus

In the end the completed graphs were tested on the 5MB biomedical reference corpus, containing about 18.000 sentences. This was to see if the graphs were applicable also on texts that the system was not specifically trained to handle.

From the 18.000 sentences about 400 facts were extracted. That means that one activation-fact was found for every 50th sentence. The real ratio is probably much higher, but this number (2% activation-fact per sentence) is still good, because it shows that the graphs have some generality in them,

after being trained on only 59 sentences. A problem with the extracted facts was that many of them only contained the *activated* entity, and not the *activator*. After closer inspection it is evident that this is because many of the *activators* in the test set were not in the Unitex dictionaries, and therefore could not be matched by any Unitex graph.

6.5 Building Graphs

This section contains a general discussion about the graph-building work, and a few notes that can be useful to others undertaking such work later.

6.5.1 Different Stages

The building of the graphs went through three more or less well defined iterations. The first iteration felt like putting different sentences together almost at random, but it was soon discovered that many sentences were too long for all the words to fit beside each other in one graphpage, so the second iteration consisted of constructing subgraphs to cluster groups of words together and represent them as just one box (subgraph). These graphs were made so that words that often stood together in different sentences were put into the same subgraph. That allowed entire sentences to be represented in the main (top node) graph while still maintaining the desired left-to-right reading property. As the number of subgraphs grew, it became obvious that a good naming scheme was needed. It took some time to work this out, and that means that some graphs had to be completely rearranged later, and a few of the graph names had to be changed. It is a good idea to avoid this, because it will definitely introduce some new errors into the system, e.g. sentences that were recognized by the old graph might not be recognized by the new graph, and it will often take a lot of debugging to figure out exactly why. This phenomenon also happens for example when a function name is being changed in the code of a big program: When the name of a function is changed, all the places that call this function must also be updated. There is no support for such name changes in Unitex, so a lot of time will be spent doing this manually, and it is very easy to miss something, and get strange errors.

The third iteration was caused by the fact that too many different semantic meanings often ended up in a single graph, making it hard to extract meaningful facts from these specific graphs later. So another rearrangement consisting of splitting the graphs with ambiguous meanings

into separate disambiguate graphs had to be done. This caused the "height" of the graphs (number "of lines" or "parallelism" in the graph) to increase, since different paths leading into one "ambiguous" box, now had to go to new separate disambiguated boxes. That also means that some boxes had to be duplicated, which is generally not a good thing, because then all subsequent work on the specific boxes must also be duplicated. Still, this is necessary, since the semantic output from the different boxes in the end must be different. For example "*activation of X*" can mean that X is being activated in one sentence, but that X is the activator in another sentence.

6.5.2 Naming Scheme

Different abstractions were tried in order to find good names for the subgraphs during the construction work. This consisted of splitting the sentences into well defined semantic and/or POS-based units. This turned out to be harder than expected, because quite often there would be an overlap between the units, and this would often be discovered long after the choice was made and the graph already constructed. The most successful abstractions were those including Gene Name Noun Phrases (POS), and the Activator/Activated (Semantic) sub-graphs. Since Protein/Gene name discovery in biomedical texts is considered a more or less solved problem [6], it is not necessary to be too careful about the explicit content of these graphs. In this work, these graphs were simply manually filled with the explicit coding of the names as they appeared in the text. For the sake of building a complete system later, it is very important to find one of these systems that does protein/gene name discoveries in medical texts, because the current solutions are either too slow (manually coding every entry) or not accurate enough (importing *probable* entity names from nomenclature resources on internet).

Another very successful abstraction/naming scheme for the graph-building work was to make separate subgraphs for every *prepositional phrase* (PP), based on what leading preposition they contained. This was very practical when new sentences were added into the graph system, because one only had to identify the prepositions of the sentence, and then it was already obvious how the sentence should be split into subgraphs. The problem with this approach was that it sometimes led to "collisions" with the "*gene name, activator and activated*" naming scheme. Many sentences are on the form shown in Figure 30, and then a choice must be made whether "of Y by Z" should be coded by the "of" and "by" PP-graphs, or by the "activated" and "activator" sub-graphs. Regardless of what choice is being made, these different forms should be located close to

each other in the parent graph, to ease the work later of debugging and add-ons to the system.

Ser-133 phosphorylation of CREB by PKA

Figure 30. X Activation of Y by Z

Another problem with the PP-graphs naming abstraction became evident later, as semantics were incorporated into the graphs. For example, the PP-graph called *"inPP"* (Appendix D), contains prepositional phrases with very different semantic meanings, and the only thing they have in common is the fact that all this phrases start with the word "in". Because the semantics of these phrases are so different from each other, it would be better to spread them across different graphs. This is already partly done for example with the "InResponseToPP" graph.

6.5.3 Time Representation

Many sentences in the biomedical domain contain sentences describing the timing between different events, or the duration of one specific event. It would therefore be useful to build dedicated subgraphs to represent such sub-sentence fragments. This was not done in this project, because too few "time" examples occurred for any useful generalizations to be made. E.g. the occurrences that appeared were just "hard coded" into the appropriate higher-level graphs.

7 Future Work

This chapter will give an overview of what tasks that should be solved during the rest of this PhD work.

7.1 Results and Standards

Before this work can be taken any further, with GeneTUC or Unitex, it is important to decide exactly what we want to extract from the biomedical texts. It is apparent that simple "X activates Y" facts are usually not what should be extracted, because the reality is almost always more complex than that (see 6.3.4).

In this project an attempt has been made to parse entire sentences, just like it is being done in GeneTUC, but this might be a waste of time if the real goal is just to extract activators and their activated entities. Additional facts that could be extracted with the activation facts include:

- …by *method*
- …in *cell/area*
- …with *certainty*

7.2 Unitex Integration with GeneTUC

The thesis work has been an effort to identify specific ways that "interesting facts" are actually written in the texts at our disposal. This was done with the graph drawing tool Unitex, but the results are also directly relevant to GeneTUC, as the same sentences should ideally also be recognized by TUC's grammar. Different ways of integrating the two systems will be discussed below.

7.2.1 TQL-Code

Since Unitex consists of Finite State Transducers (FSTs), it can produce output as well as "understanding" the input. This means that one way of integrating the two systems, is to let Unitex do the preliminary parsing, and then produce TQL-code that can be further processed by GeneTUC, e.g. in question-answering tasks. This would be very efficient, since FSTs are much faster than context sensitive grammars of TUC. The problem is that FSTs are not as expressive as the TUC grammar, and they might not recognize all the sentences of interest (Or it would take a huge amount of work to build all the corresponding local grammars!)

Another problem with the TQL-code is that no formal specification exists, because the format has been made bit by bit, in a slightly ad-hoc and pragmatic way. And since the TQL-code "standard" is subject to modification at any time, it is a little like shooting a moving target. In other words, if TQL-code is chosen as the future language of choice to formulate golden standards for example, the first step should be to make a more formal description of the "TQL-language" in order to avoid the problems mentioned above.

7.2.2 Pre-Processing

Another way of integrating the two systems is by letting Unitex do some simple pre-processing of the text, and then let GeneTUC build the TQL-code. This would greatly increase GeneTUC parsing rate, because one of the major reasons for parsing failures on unseen material is the fact the TUC "crashes" when unknown names are being used in the sentences. The only way around this problem is to explicitly encode every possible name into the TUC grammar, and this is more or less impossible, as new names are being "generated" every day.

Using Unitex, the problem can be solved in a slightly different way. In Unitex it would be necessary, or at least a great advantage, to build one local grammar for every single *entity* (Gene, protein, hormone etc.) covering all the different ways the specific entity can be named. However, after enough examples have been collected, it is possible to generalize the local grammars using semantic tags or regular expressions so that they also recognize similar but not already explicitly coded names. The idea behind this approach is that there is some sort of system or conventions for making up new names, and such conventions can easily be expressed using local grammars. A lot of research has been done in this kind of *named entity extraction* the last years, and it should be possible to build on some of this

work when constructing the local grammar *entity* graphs. The main idea for future work in this area is to contact the authors of for example [6], since they also focus on protein names. Contact was already made with this research group last year, as we all were at the same conference (ACL HLT2002, [19]). A question should be forwarded to this group to see if they would be interested in any sort of cooperation, or if we could base our entity-naming algorithm on the results that they got in their work.

7.3 General Linguistic Topics

After discussions with people at the CIS group at LMU Munich, it has become obvious that many of the challenges of parsing biomedical texts are just the same as are encountered in parsing in almost any other domain. A few such example problems are "predicative nouns", "sentence conjunction handling", "anaphoric resolution" and "the use of *not* and negated facts". All these problems were encountered during this project work, and they were solved in an ad-hoc fashion, using the Unitex graph-tools. In order to reduce the time needed to parse new training sentences, and to improve the quality of the graphs being built, the problems above should be attacked in a systematic manner in the future. Each of these problems is complex enough to be the topic of its own PhD thesis, so it should not be attempted to solve all of them. But many other researchers are already starting to find solutions to these challenges, so a very important part of the future work is to keep up-to-date on these four topics and keep thinking about how new solutions can be integrated with or benefit the GeneTUC system.

Another topic that is starting to get a lot of attention is the question: "How can the construction of (local) grammars be automated?" It is clear after spending a lot of time with only 59 training sentences, that the manual construction of local grammars is too time-consuming, and without some automation the goal of complete coverage will probably not be reached in our life-time. The "induction" of such grammars, based for example on a (semantically) tagged corpus, is therefore another idea for future research.

The last linguistic problem that should be solved using a general methodology is the problem of representing synonyms (for example PKB = AKT). This problem is not as complex as some of the others, but it can lead to a lot of trouble and bad results at later stages if it is not handled properly from the beginning. In the GeneTUC system a solution to this problem is the predicate *synword*. It is currently largely used to handle spelling errors, but can also be used to handle synonyms. In Unitex synonyms could be handled for example by using one common lemma form for all the words that have the same meaning. Another alternative is using distinctive semantic tags for groups of synonyms. The main challenge will anyway be to find good sources of already identified synonyms in the medical domain.

Good starting points for this work will be existing online ontologies, for example Gene Ontology [26].

7.4 Conclusion

In this thesis GeneTUC and Unitex have been compared. They represent two different approaches to the same text-parsing problem, even though GeneTUC should be more than just a parser in the end. Unitex is based on the view that we have to collect all valid sentence examples from a domain, before we can hope to do successful parsing in this domain. This has been done with good success in the domain of stock market news from newspapers, but it might not be possible to "collect all sentences" in the medical domain, because there are simply *too many* "different sentences". GeneTUC uses the approach of building a context sensitive grammar that should cover all valid biomedical sentences, but this is also hard, because the sentences are usually quite complex, and the grammar must be tailored to fit also the tricky cases. The claim behind Unitex is that there are usually more *differences* between two sentences than there are common features, so trying to make *general* rules will usually not be good enough.

Sooner or later, in a working system these two approaches should meet each other "somewhere in the middle", because they are working on the same problem, but from two different ends. After enough local grammars have been built with Unitex, they should act together in a high-level super-graph as a complete grammar. And after the GeneTUC grammar has been adopted to fit enough different sentence examples, it should be able to parse as good as the grammars that were built in the bottom-up fashion.

Since it is not clear whether the coding of (possibly infinitely) many specific example sentences is more effective than the constant refining of a general grammar, it makes sense to keep pulling at the rope from both ends. In the future we will have to keep collecting actual sentences from the domain in question, and these sentences can be easily modelled with Unitex. After enough examples have been coded in Unitex, general rules or patterns are bound to emerge, and these rules could then also be implemented in the general TUC grammar.

Before any grammar can be built, both systems are dependent on good dictionaries and ontologies, and the only way to make them is by collecting all the specific examples from the domain, and grouping them together in some meaningful fashion. GeneTUC already has a pretty good microbiological ontology that has been built in a very pragmatic way; a new entry is added whenever it is needed to parse a new training or test

sentences. A part of the future works would be to keep adding to this ontology, with the goal of one day making it a “complete” ontology. The speed of this work can be greatly increased with the help of already existing ontologies such as WordNet and Gene Ontology [26], and other online resources.

References

1. Andenæs, A. (2000) **GeneTUC - /j&'-ne-tük/**. Master's Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology. URL: http://www.idi.ntnu.no/~natlang/GENETUC/thesis.ps
2. Bennett, H. A., He, Q., Powell, K., and Schatz, B. R. (1999) **Extracting noun phrases for all of Medline**. In *AMIA '99 (American Medical Informatics Assoc) Conf*, Washington, DC.
3. Brill, E. (1992) **A simple rule-based part-of-speech tagger**. In *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, Trento, Italy pages 152-155
4. Cohen, K. B., Dolbey, A. E., Acquaah-Mensah, G. K. and Hunter, L. (2002) **Contrast and variability in gene names**. In *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*. Pages 14-20.
5. Collier, N., Park, H., Ogata, N., Tateishi, Y., Nobata, C., Ohta, T., Sekimizu, T., Imai, H., Ibushi, K. and Tsujii, J. (1999) **The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers**, in Proc. of the Annual Meeting of the European Association for Computational Linguistics (EACL-99), pp.271-272, Norway
6. Eriksson, G., Franzén K., Olsson, F., Asker, L., Lidén, P. (2002). **Using Heuristics, Syntax and a Local Dynamic Dictionary for Protein Name Tagging**. Human Language Technology Conference 2002, San Diego, USA. URL: http://www.sics.se/~fredriko/papers/hlt02_abstract.pdf
7. Fukuda, K., Tsunoda, T., Tamura, A., and Takagi, T. (1998) **toward information extraction: Identifying protein names from biological papers**. In *Proc. of the Pacific Symposium on Biocomputing '98 (PSB'98), Hawaii*. Human Genome Center, Institute of Medical Science, University of Tokyo. Email: ichiro@ims.u-tokyo.ac.jp.
8. Gross, Maurice. (1997) **the Construction of Local Grammars**. In *Finite-State Language Processing*, E. Roche & Y. Schabès (eds.), Language, Speech, and Communication, Cambridge, Mass.: MIT Press, pages 329-354
9. Hishiki, T., Collier, N., Nobata, C., Ohta, T., Ogata, N., Sekimizu, T., Steiner, R., Park, H. S., and Tsujii, J. (1998) **Developing NLP tools for genome informatics: An information extraction perspective**. In *Proc. of*

Genome Informatics, pages 81-90, Tokyo, Japan. Universal Academy Press, Inc., Tokyo, Japan.

10. Jenssen, T. K., Lægreid, A., Komorowski, J., and Hovig, E. (2001) **a literature network of human genes for high-throughput analysis of gene expression**. In *Nature Genetics*, 28(1):21-28.
11. Mack, R. and Hehenberger, M. (2002) **Text-based knowledge discovery: search and mining of life-sciences documents**. In *Drug Discovery Today* 7(11):S89-S98
12. Ohta, Y., Yamamoto, Y., Okazaki, T., Uchiyama, I., and Takagi, T. (1997) **Automatic Construction of Knowledge Base from Biological Papers**. In *Proc. of the Fifth International Conference on Intelligent Systems for Molecular Biology (ISMB'97)*, pages 218-225.
13. Park, Jong C. (2001) **Using Combinatory Categorial Grammar to Extract Biomedical Information**. In *IEEE Intelligent Systems*, 6. Korea Advanced Institute of Science and Technology. URL: http://computer.org/intelligent/ex2001/x6toc.htm, pages 62-67
14. Proux, D., Rechenmann, F., Julliard, L., Pillet, V., and Jacq, B. (1998) **Detecting gene symbols and names in biological texts: A first step toward pertinent information extraction.** In Miyano, S. and Takagi, T., editors, *Ninth Workshop on Genome Informatics*, volume 9, pages 72-80, Tokyo, Japan.
15. Pustejovsky, J., Castaño, J., Zhang, J., Kotecki, M. and Cochran, B. (2002) **Robust Relational Parsing over Biomedical Literature: Extracting Inhibit Relations.** In *Proceedings of the Pacific Symposium on Biocomputing*, pages 362-373. Hawaii, USA
16. Sætre, Rune. (2002) **GeneTUC**. Technical report, Department of Computer and Information Science, Norwegian University of Science and Technology, 2002. URL: http://www.idi.ntnu.no/~satre/genetuc/GeneTUC.pdf
17. Tanabe, L., Scherf, U., Smith, L. H., Lee, J. K., Hunter, L. and Weinstein, J. N. (1997) **MedMiner: An Internet Tool for Filtering and Organizing Gene Expression and Pharmacological Information**. *Biotechniques submitted*
18. Yakushiji, A., Tateisi, Y., Miyao, Y. and Tsujii, J. (2001) **Event Extraction from Biomedical Papers Using a Full Parser**. In *Proceedings of Pacific Symposium on Biocomputing 2001*, pages 408-419, Hawaii, USA
19. Proceedings of the Second International Conference on **Human Language Technology Research, HLT2002**. ISBN: 0-12-470870-6
20. Session Proposal (2001). **Large coverage dictionaries and grammars for text processing: the INTEX system**: URL: http://www.nyu.edu/its/humanities/ach_allc2001/papers/fairon/
21. Association of Computational Linguistics. URL: http://www.aclweb.org/archive/what.html
22. Biology Definitions. URL: http://biology-pages.info/

23. Cis- and trans-acting. URL: http://web.mit.edu/esgbio/www/pge/mutants.html
24. CiteSeer. URL: http://citeseer.nj.nec.com/
25. GENATLAS. URL: www.dsi.univ-paris5.fr/genatlas/
26. Gene Ontology. URL: http://www.geneontology.org/
27. The Genome Database. URL: http://gdbwww.gdb.org/
28. Google. URL: www.google.com
29. The Human Genome Organisation (HUGO). URL: www.gene.ucl.ac.uk/hugo/
30. The Human Genome Organisation (HUGO) Gene Nomenclature Committee. URL: http://www.gene.ucl.ac.uk/nomenclature/
31. INTEX Tutorial. URL: http://grelis.univ-fcomte.fr/intex/downloads/Notes.pdf
32. Laboratory for Information Retrieval Systems and Linguistics, France. URL: http://ladl.univ-mlv.fr/index.html
33. LocusLink. URL: www.ncbi.nlm.nih.gov/LocusLink/
34. Medstract Project. URL: www.medstract.org
35. Moore's Law. URL: www.webopedia.com/TERM/M/Moores_Law.html
36. Unitex. URL: http://www-igm.univ-mlv.fr/~unitex/
37. Unitex, Manual. URL: http://www-igm.univ-mlv.fr/~unitex/manuel.html
38. Pacific Symposium on Biocomputing. URL: http://psb.stanford.edu/psb02/
39. PubMed Medline. URL: www.pubmed.gov
40. Sébastien Paumier. E-Mail: paumier@univ-mlv.fr
41. Unitex Home Page. URL: http://www-igm.univ-mlv.fr/~unitex/index.html
42. WordNet. URL: http://www.cogsci.princeton.edu/~wn/
43. Pronouns. URL: http://webster.commnet.edu/grammar/pronouns.htm
44. Determiners. URL: http://webster.commnet.edu/grammar/determiners/determiners.htm
45. Adverbs. URL: http://webster.commnet.edu/grammar/adverbs.htm
46. Conjunctions. URL: http://webster.commnet.edu/grammar/conjunctions.htm

A CCK Abstract

The CREM gene encodes both activators and repressors of cAMP-induced transcription.

ICER (Inducible cAMP Early Repressor) isoforms are generated upon activation of an alternative, intronic promoter within the CREM gene.

ICER is proposed to down-regulate both its own expression and the expression of other genes that contain cAMP responsive elements (CREs) such as a number of growth factors.

Thus, ICER has been postulated to play a role in proliferation and differentiation.

Here we show that ICER gene expression is induced by gastrin, cholecystokinin (CCK) and epidermal growth factor (EGF) in AR42J cells.

The time course of gastrin- and CCK-mediated ICER induction is rapid and transient, similar to forskolin- and PMA- induced ICER expression.

The specific CCK-B receptor antagonist L740093 blocks the gastrin- but not the CCK -response. This indicates that both the CCK-B receptor and the CCK-A receptor can mediate ICER gene activation.

Noteworthy, CREB is constitutively phosphorylated at Ser 133 in AR42J cells, and ICER induction proceeds in the absence of increased CREB Ser 133 -P.

Gastrin-mediated ICER induction was not reduced in the presence of the PKA inhibitor H-89. This indicates a PKA independent mechanism.

This is the first report on ICER inducibility via Gq-G11 protein coupled receptors.

B 59 Activate Sentences

1. Follicle stimulating hormone (FSH) activates the cAMP signaling pathway and consequently, CREB positively auto-regulates its own expression (by binding to a CRE like element in its promoter).
2. Subsequently, activated CREB activates transcription of genes essential for proper germ cell differentiation.
3. Inducible cAMP early repressor (ICER), a suppressor isoform of CREM, also activated by CREB, down regulates CREB expression together with its own expression, resetting CREB to basal level that enables a new spermatogenic wave.
4. FSH binds to Sertoli cells and thus activates the production and secretion of factors needed for germ cell survival and differentiation.
5. LH binds the Lydig cells, somatic cells located in the interstitial regions (between tubules), and activates production and secretion of testosterone, which exerts its effect on spermatogenesis, again, through Sertoli cells.
6. CREB was first identified during a search for factors that recognize the cis-acting element TGACGTCA (CRE element) that was shown to confer cAMP-inducible transcription of the neuropeptide hormone gene somatostatin
7. Elevation in intracellular cAMP activates PKA activity, by releasing its regulatory subunit, and this activated PKA is transported to the nucleus, where it phosphorylates CREB at Ser-133, within the KID domain.
8. Only Ser-133 phosphorylated CREB can bind CREB binding protein (CBP), a co-activator that is needed to recruit the basal transcription machinery (Fig. 2).
9. Ser-133 phosphorylation of CREB by PKA, CBP binding and CBP dependent recruitment of the basal transcription machinery is, therefore, the path through which cAMP can regulate transcription of specific genes.

10. Activated PKA localizes to the nucleus where it phosphorylates CREB on Ser-133 and activates it.
11. The positive autoregulation, therefore, consists of FSH that activates PKA activity in Sertoli cells resulting in Ser-133 phosphorylation and hence activation of CREB.
12. Activated CREB bound to its own promoter amplifies CREB transcription, leading to CREB dependent transactivation of genes important to support spermatogenesis (Fig. 4).
13. Monaco et al. (1995) have shown that FSH stimulates expression of the ICER isoform of CREM in primary culture of rat Sertoli cells.
14. This implies that the FSH induced expression of ICER, is a link in a negative auto-regulation chain of CREB.
15. FSH induced CREB binds to the CRE elements within the ICER promoter and activates its expression.
16. In addition, it has been well documented that FSH binding to Sertoli cells results in elevation of cAMP levels and activation of the PKA signaling pathway.
17. We hypothesized that c-fos gene transcription is rapidly stimulated by LiCl, followed later by the expression of the inducible cAMP early repressor (ICER) transcription factor, a negative modulator of cAMP-mediated gene transcription.
18. Several studies have shown by either in situ hybridization or immunohistochemistry that doses of lithium chloride (LiCl) sufficient to produce CTA (76 mg/kg or higher) induce c-fos gene expression in rat brain.
19. Induction of c-fos gene expression by lithium may be mediated by the cAMP signaling pathway and cAMP-activated transcription factors such as cAMP response element-binding protein (CREB).
20. LiCl has been shown to induce CREM gene expression in vivo;
21. using in situ hybridization (ISH), Lamprecht and Dudai observed increased CREM mRNA levels in rat CeA 40 min following LiCl injection [21].
22. The specific induction of ICER expression after LiCl has not been previously described, however.
23. These prior studies provide compelling evidence, especially in the rat CeA, that LiCl activates both c-fos- and cAMP-mediated gene transcription.
24. In summary, we have shown that (1) expression of the immediate-early genes c-fos and ICER was induced by LiCl injection, but not by NaCl injection, in the CeA, PVN and SON regions of rat-forebrain;

25. We conclude that (1) c-fos and ICER mRNAs are rapidly and transiently induced by LiCl in the SON, PVN and CeA, with the peak of ICER expression delayed relative to the peak of c-fos expression;
26. The pattern of c-fos mRNA expression induced by LiCl agrees with the pattern of LiCl-induced c-Fos protein expression observed by others
27. As expected, there was a rapid and transient induction of c-fos mRNA expression by LiCl.
28. Although the co-activation of c-fos and ICER suggests activation of cAMP pathways, evidence exists for the activation of other pathways by LiCl;
29. for example, phosphorylation of MAP kinase is observed in mouse insular cortex and CeA 30 min after LiCl injection [49].
30. MAP kinase phosphorylates and activates serum response element (SRE)-binding proteins which increase c-fos transcription through the SRE site in the c-fos promoter [35].
31. Our results showed that ICER mRNA was specifically and transiently induced in rat CeA, PVN and SON by LiCl injection.
32. This study provides evidence that LiCl induces gene expression of c-fos and ICER transcription factors within that 6-h window.
33. We present evidence that gastrin, next term binding to a G protein-coupled receptor, activates the p38-mitogen-activated protein kinase (MAPK) pathway.
34. Our results demonstrate that gastrin-induced DNA synthesis requires p38-MAPK activation through mechanisms that involve calcium mobilization, PKC and Src family kinases.
35. Several laboratories, including ours, have reported the activation of the ERK pathway by the CCKB receptor and the contribution of this signaling cascade in growth-promoting effects mediated by this receptor [7, 8 and 9].
36. p38-MAPK has also been shown to be activated by GPCRs [15 and 16].
37. In particular, activation of the p38-MAPK pathway by gastrin and its role in the proliferative effects mediated by the CCKB receptor have never been studied.
38. Gastrin-dependent activation of the CCKB receptor has been shown to induce the rapid hydrolysis of phosphatidylinositol-biphosphates by phospholipase C (PLC) to generate inositol triphosphates and diacyl-glycerol which respectively mobilizes intracellular calcium and stimulates protein kinase C (PKC).
39. Since we have previously reported that ERK activation by gastrin is mediated by a signaling cascade including the phosphorylation of Shc proteins by Src-like tyrosine kinases [7], we have also analyzed the possibility that Src family kinases could serve as intermediates between the CCKB receptor and the activation of the p38-MAPK pathway.

40. Here, we report the activation of p38-MAPK by gastrin through a mechanism that involves PKC, calcium mobilization and Src family kinases.
41. We have previously reported the activation of the p85/p110 PI 3-kinase by the CCKB receptor [17] and its role upstream of the ERK pathway induced by gastrin [21].
42. In summary, our study reports the activation of p38, MAPK by the CCKB through a mechanism that involves PKC, intracellular calcium mobilization and Src family kinases.
43. From a molecular point of view, two types of PI3Ks can be activated in response to LPA stimulation.
44. Transactivation of the epidermal growth factor receptor (EGFR), as well as G protein bg subunits, is thought to play an important role in PI3Kb activation [6 and 7], but the molecular determinants of these processes have remained elusive.
45. Because the mechanisms underlying PI3K activation by LPA in non-haematopoietic cells remain poorly understood, we have explored whether lipid rafts could participate in this process.
46. Because the mechanisms coupling LPA stimulation to PI3K activation remain poorly understood, we searched for a participation of detergent-resistant membrane microdomains, putative regulatory platforms for proximal signalling events.
47. We thus observed that cholesterol level strongly modulated the activation of PI3K in response to LPA.
48. In brief, PI3K lipid products do not directly activate Akt but induce its membrane translocation and colocalisation with its upstream activating kinases, the phosphoinositide-dependent kinases (PDKs) that are also controlled by PI3K lipid products.
49. Here, we demonstrated hGRP-R activation stimulated sustained cyclic AMP response element binding protein (CREB) phosphorylation and transactivation in duodenal cancer cells through a protein kinase C and partially p38 mitogen-activated protein kinase-dependent pathway.
50. Using a specific antibody against the phosphorylated form of CREB at Ser133 in immunoblots, we showed that hGRP-R mediated Bn-dependent CREB phosphorylation in HuTu 80 cells in a dose-dependent manner (Fig. 1A).
51. To determine the specificity of Bn-induced CREB phosphorylation through hGRP-R activation, we used the specific GRP-R antagonists BW2258U89 and ME 20 min before stimulation with Bn.

52. CREB, a 43 kDa leucine zipper transcription factor, is a main regulator of gene expression which mediates the activation of cAMP-responsive genes by binding as a dimer to a conserved cAMP-responsive element (CRE), characterized by the nucleotide octamer sequence TGACGTCA [6 and 9].
53. We showed that CREB phosphorylation at Ser133 occurred in response to FSK, a stimulus known to produce CREB phosphorylation via a cAMP-dependent pathway, in duodenal cancer cells.
54. CREB is a substrate for many kinases other than PKA, including PKC, AKT, calcium-calmodulin-dependent kinases, mitogen/stress-activated kinase, and pp90rsk [6].
55. Bn has previously been shown to regulate transcription factor AP-1 activation through a PKC-dependent pathway in human gastric cancer SIIA cells [13].
56. Furthermore, Bn also resulted in transcriptional CREB activation using a GAL4-CREB luciferase reporter system.
57. Insulin-like growth factor I and nerve growth factor were shown to elicit p38 MAPK activation and result in CREB phosphorylation in PC12 cells, a pheochromocytoma cell line [15 and 16].
58. In rat pancreatic acini, Bn as well as cholecystokinin stimulated p38 MAPK activation [17].
59. We show in this study that Bn induced sustained p38 MAPK phosphorylation for at least 4 h.

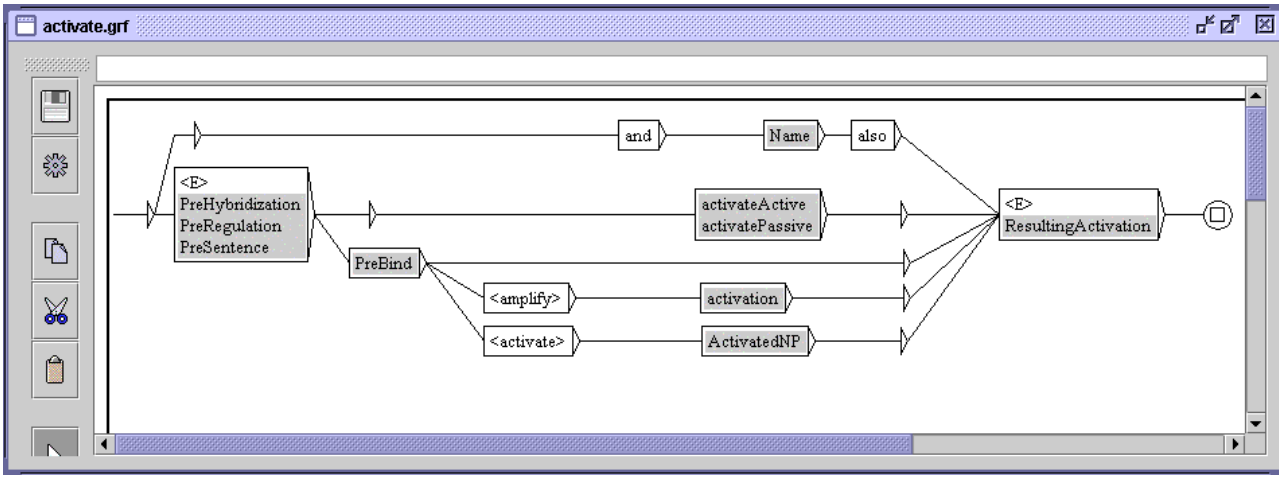
C Unknown Words Dictionary

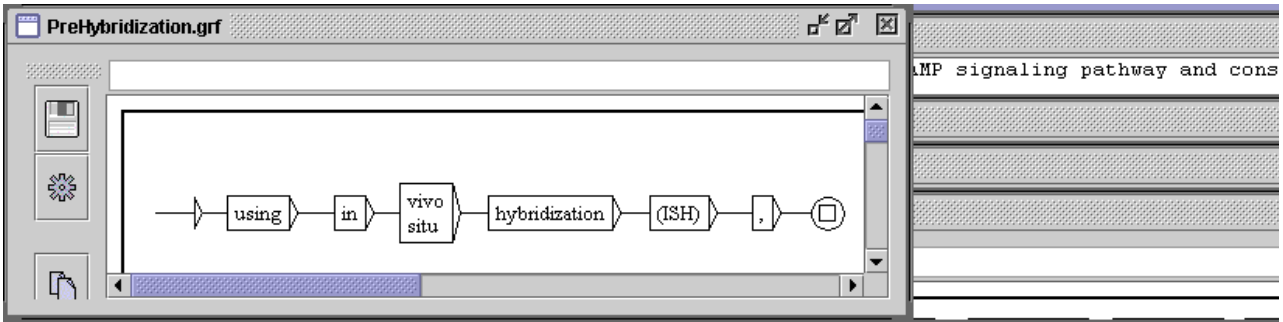
AKT,.N+Kinase:s
Akt,AKT.N+Kinase:s
AP-1,.N+ProtHum:s
autoregulation,.N+Process:s
bg,.A
Bn,.N+Substance:s
BW2258U89,.N+Antagonist:s
calcium,.N+Element:s
CCKB,.N+Receptor:s
CeA,.N+Antigen:s
c-fos,.N+Gene:s
cis,.PFX
colocalisation,.N+Activity:s
CRE,.N+Element:s
CTA,.N+Aversion:s
diacyl-glycerol,.N+Glycerol:s
down regulates,down regulate.V+P3s
Dudai,.N+Author:s
FSH,.N+Hormone:s
FSK,.N+Substance:s
gastrin,.N+Substance:s
GPCRs,GPCR.N+Receptor:p
hGRP-R,.N+Receptor:s
HuTu 80,.A
ICER,.N+ProtHum:s
immunoblots,.N+Technique:s
kDa,.N+Measure:s
kg,.N+Measure:s
Lamprecht,.N+Author:s
LH,.N+Hormone:s
LiCl,.N+Salt:s
Lydig,.A
MAPK,.N+Kinase:s
mg,.N+Measure:s
microdomains,microdomain.N+Domain:p
mRNA,.N+RNA:s
mRNAs,mRNA.N+RNA:p
NaCl,.N+Salt:s

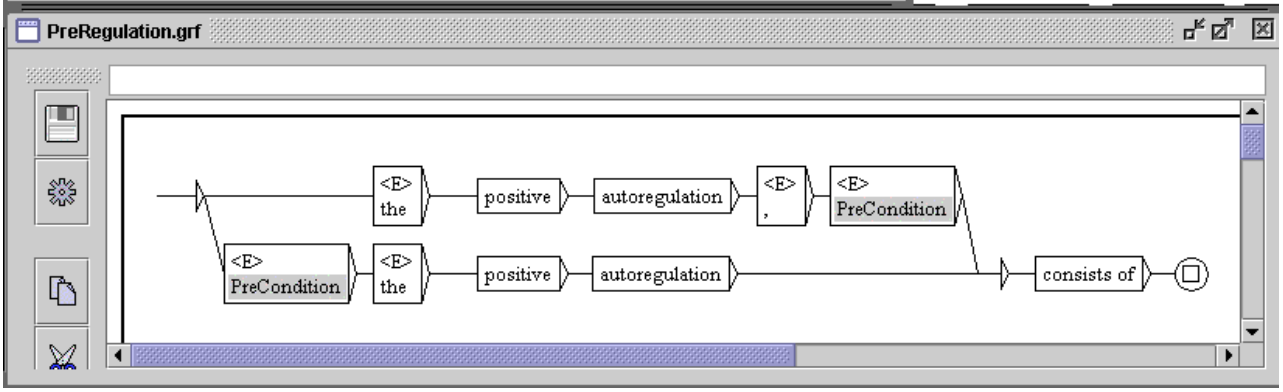
octamer,.N+Sequence:s
PDKs,PDK.N+Kinase:p
pheochromocytoma,.A
phosphatidylinositol,.N+Lipid:s
phosphatidylinositol-biphosphates,phosphatidylinositol-biphosphate.N+ProtHum:p
phospholipase,.N+Enzyme:s
PI3Kb,.N+Gene:s
PKA,.N+Kinase:s
PKC,.N+Kinase:s
pp90rsk,.N+Kinase:s
PVN,.N+Nucleus:s
SIIA,.A
somatostatin,.N+Hormone:s
spermatogenic,.A
SRE,.N+Element:s
testosterone,.N+Hormone:s
TGACGTCA,.N+Sequence:s
transactivation,.N+Activation:s

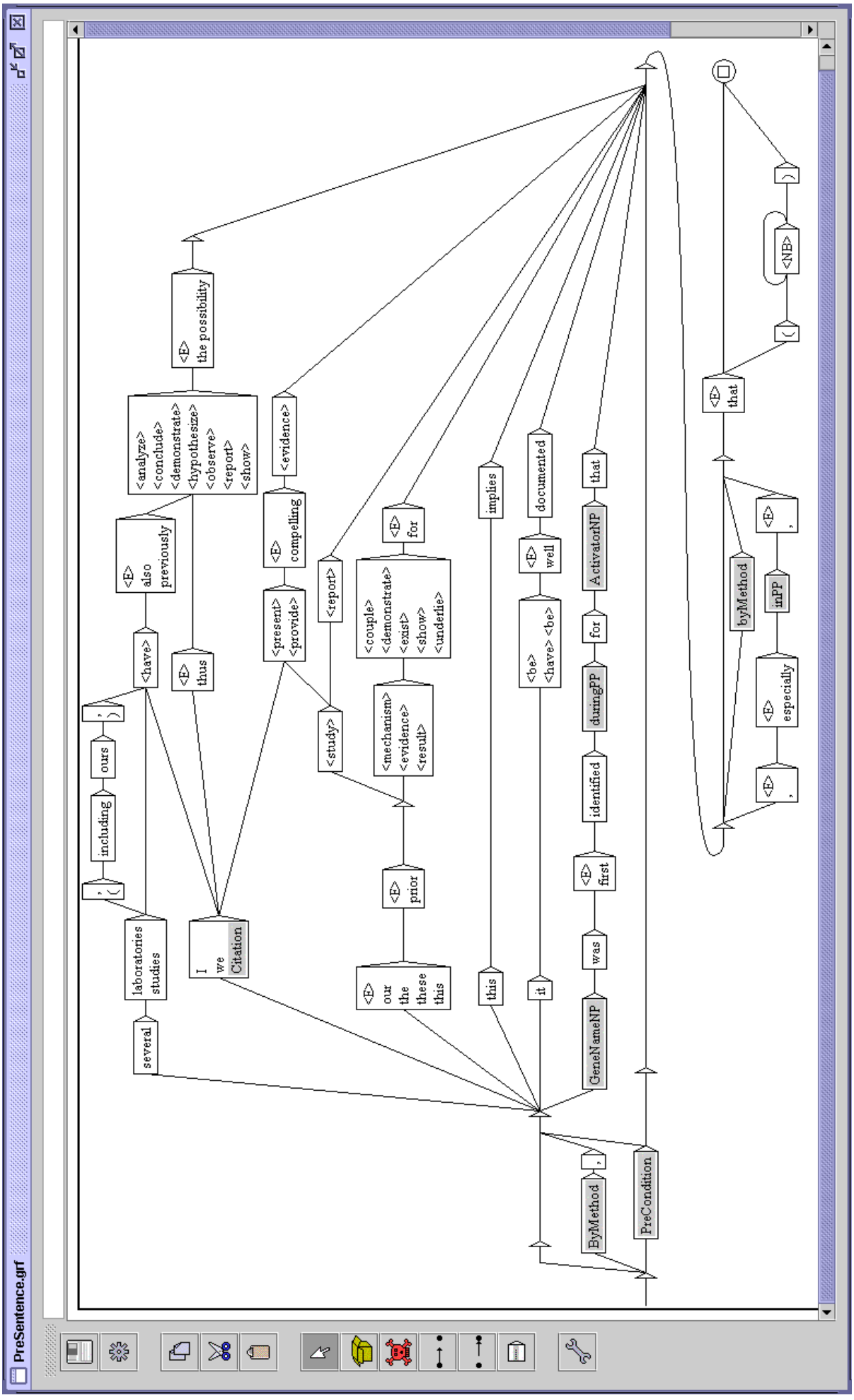
D Local Grammar Graphs

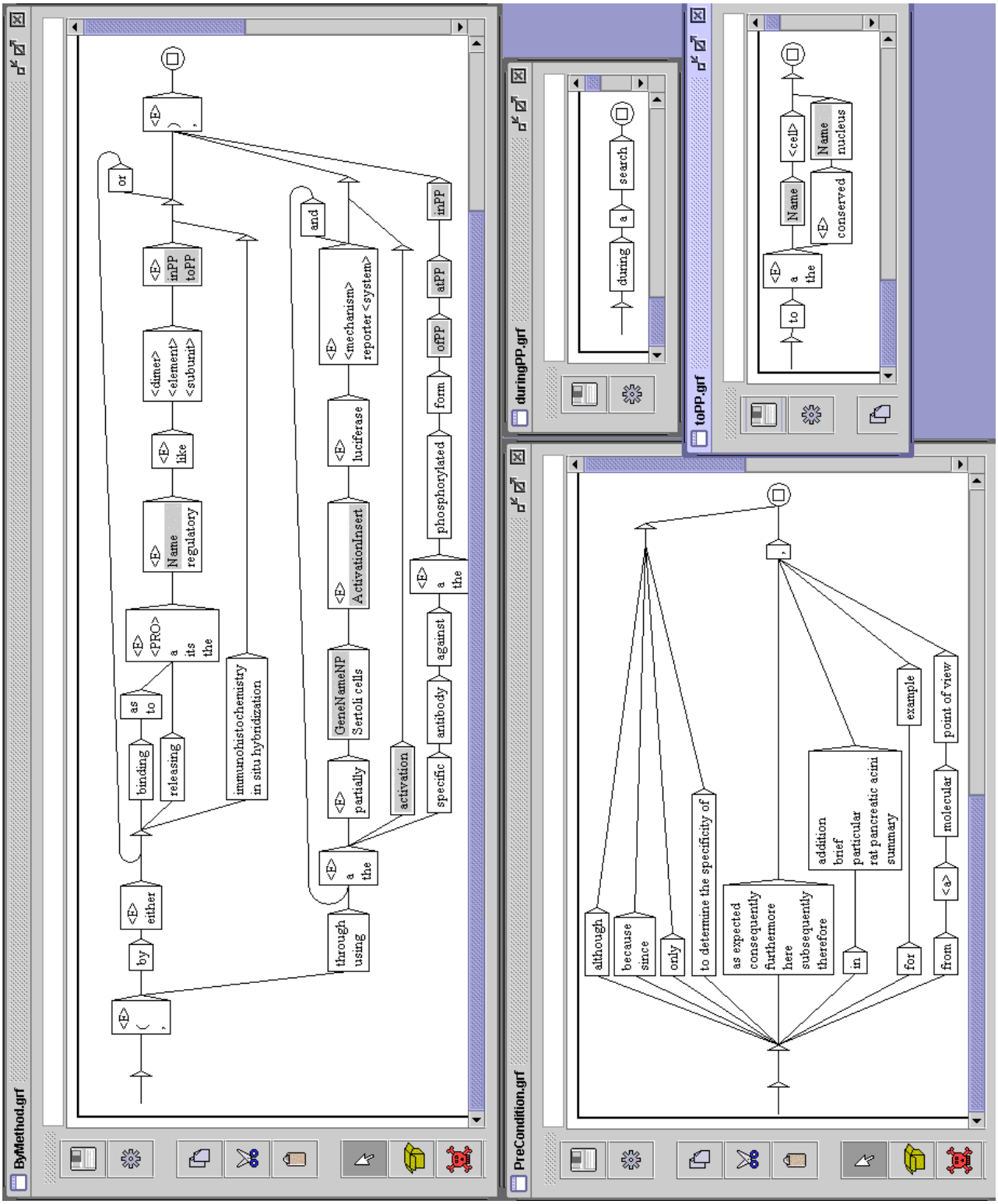
In this appendix a representative sample of all the graphs that were constructed is given.

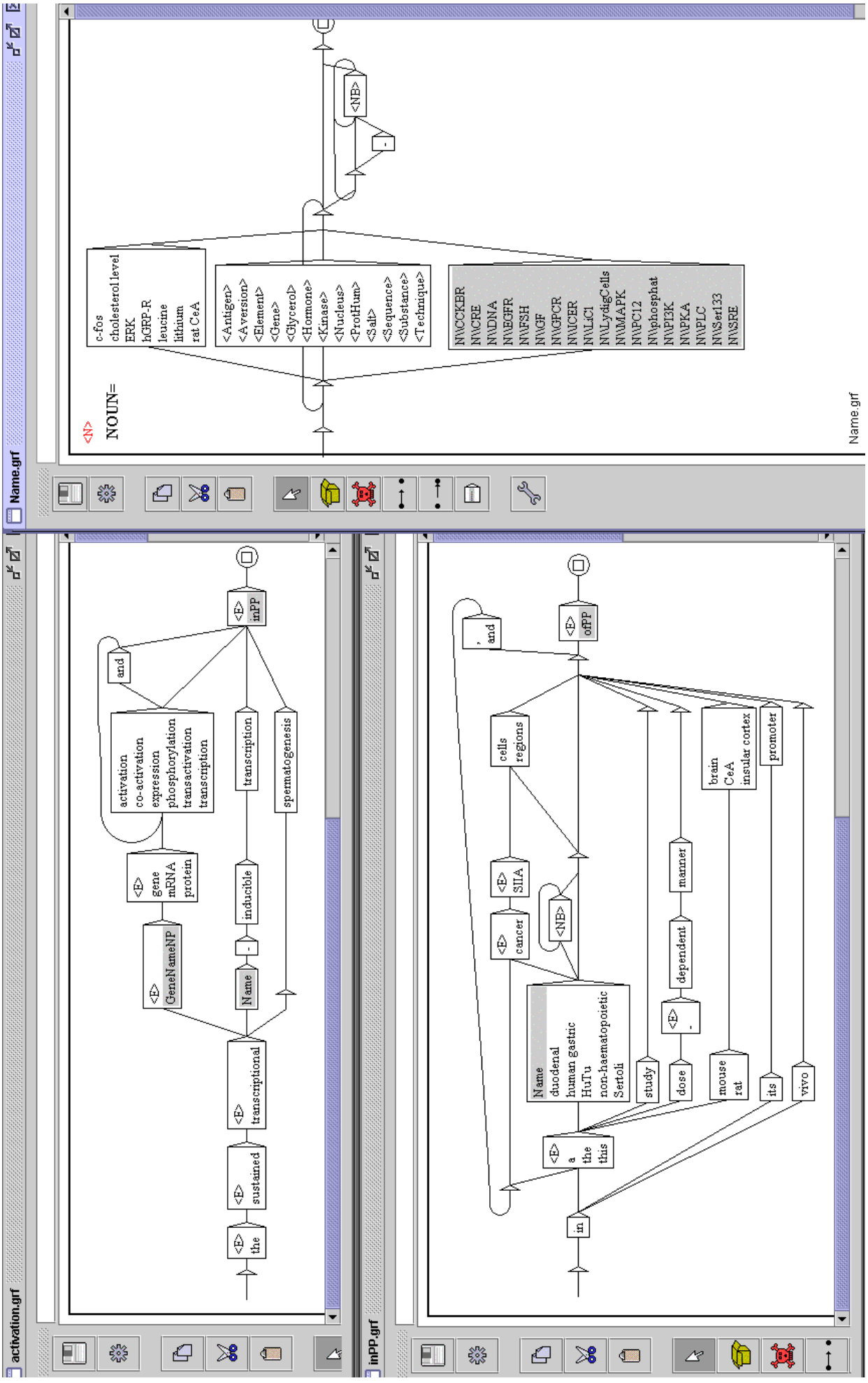


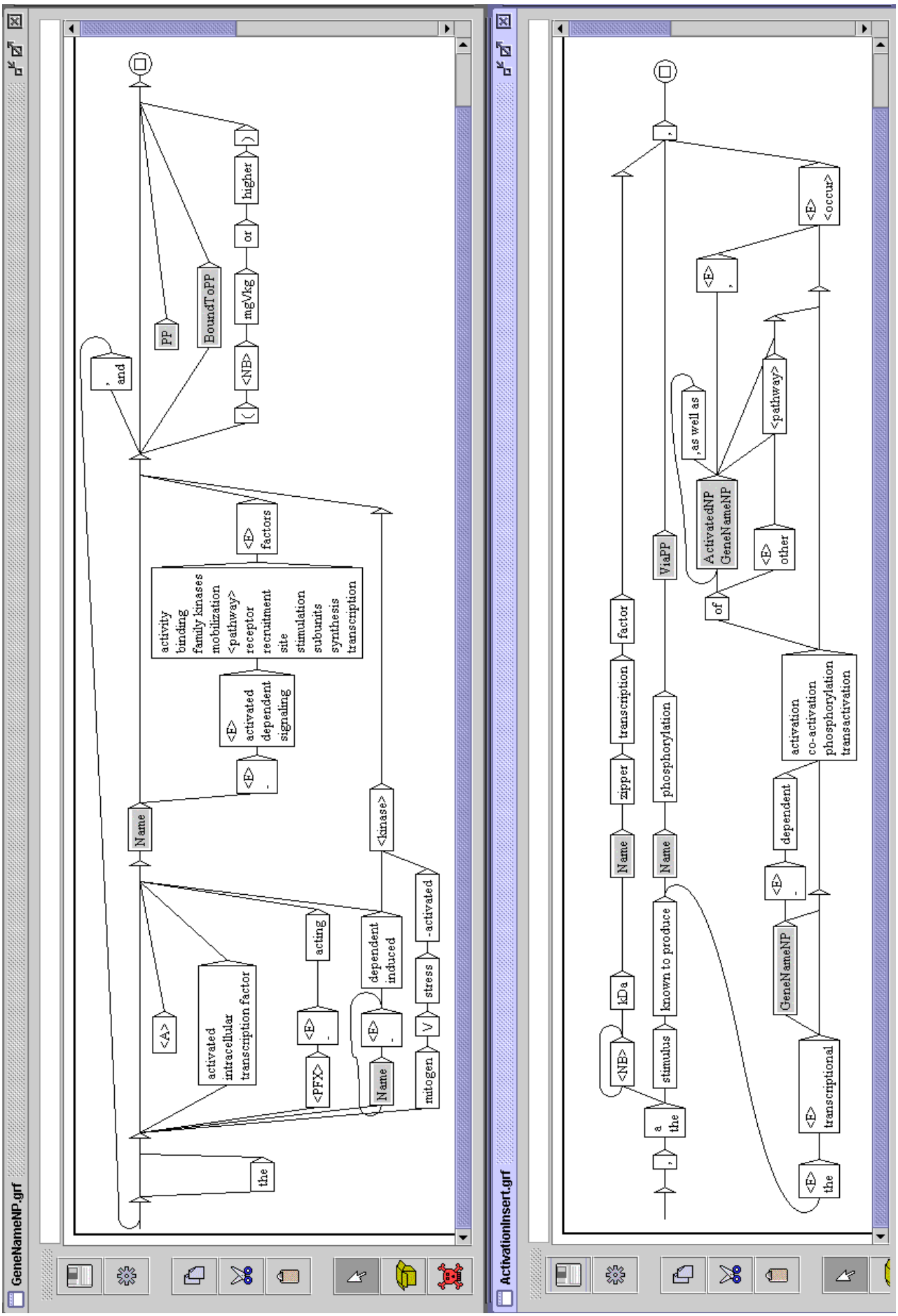


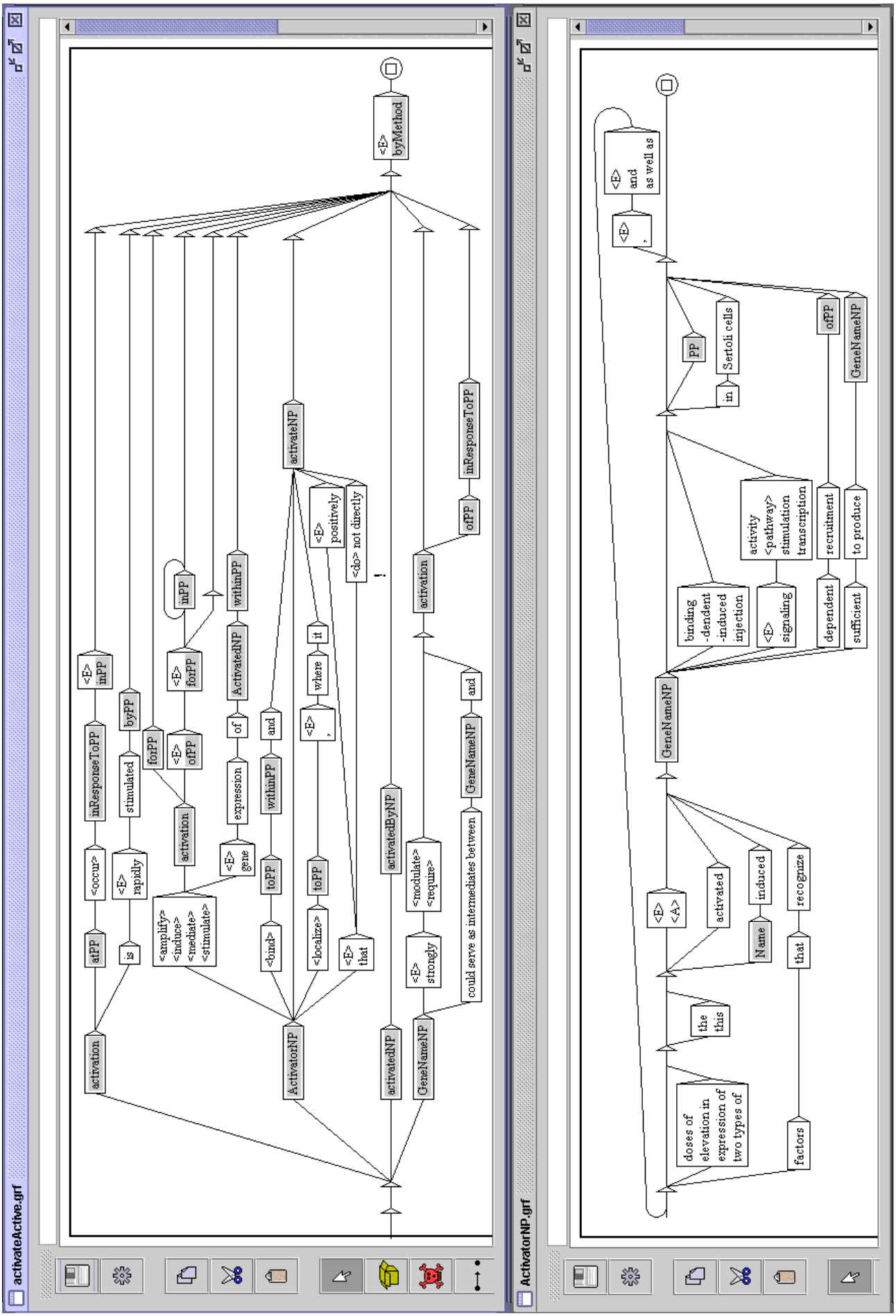


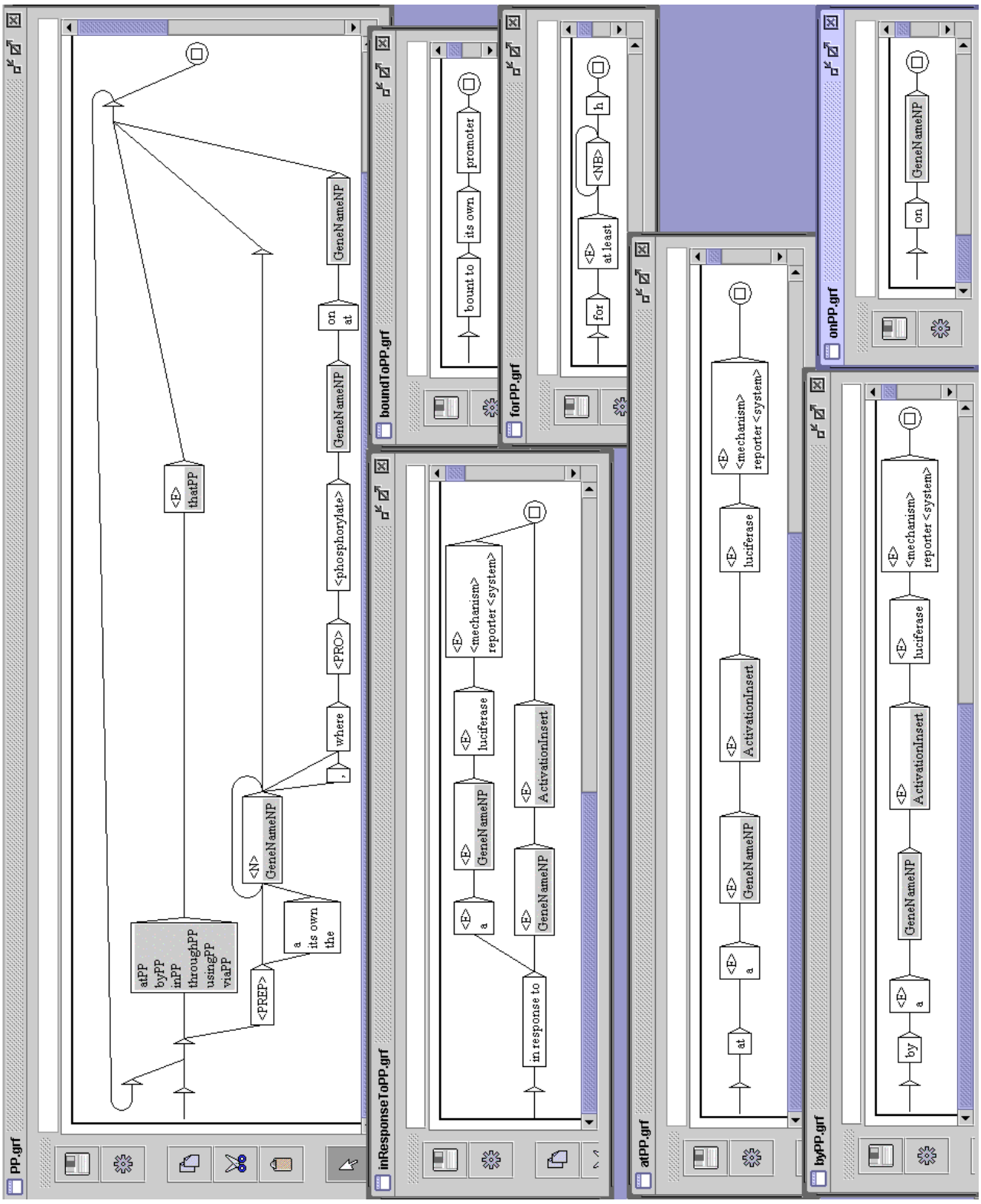

PreSentence.grf
<E> the possibility
<analyze> <conclude> <demonstrate> <hypothesize> <observe> <report> <show>
<E> also previously
<have>
<E> thus
<evidence>
<E> compelling
<present> <provide>
<report>
<study>
<E> for
<couple> <demonstrate> <exist> <show> <underlie>
<mechanism> <evidence> <result>
<E> prior
<E> our the these this
implies
documented
<E> well
<be> <have> <be>
that
ActivatorNP
for
duringPP
identified
<E> first
was
GeneNameNP
this
it
several
laboratories studies
I we Citation
, (including ours ,)
ByMethod
,
PreCondition
<E> that
byMethod
<E> ,
inPP
<E> especially
<E> ,
(<NB>)

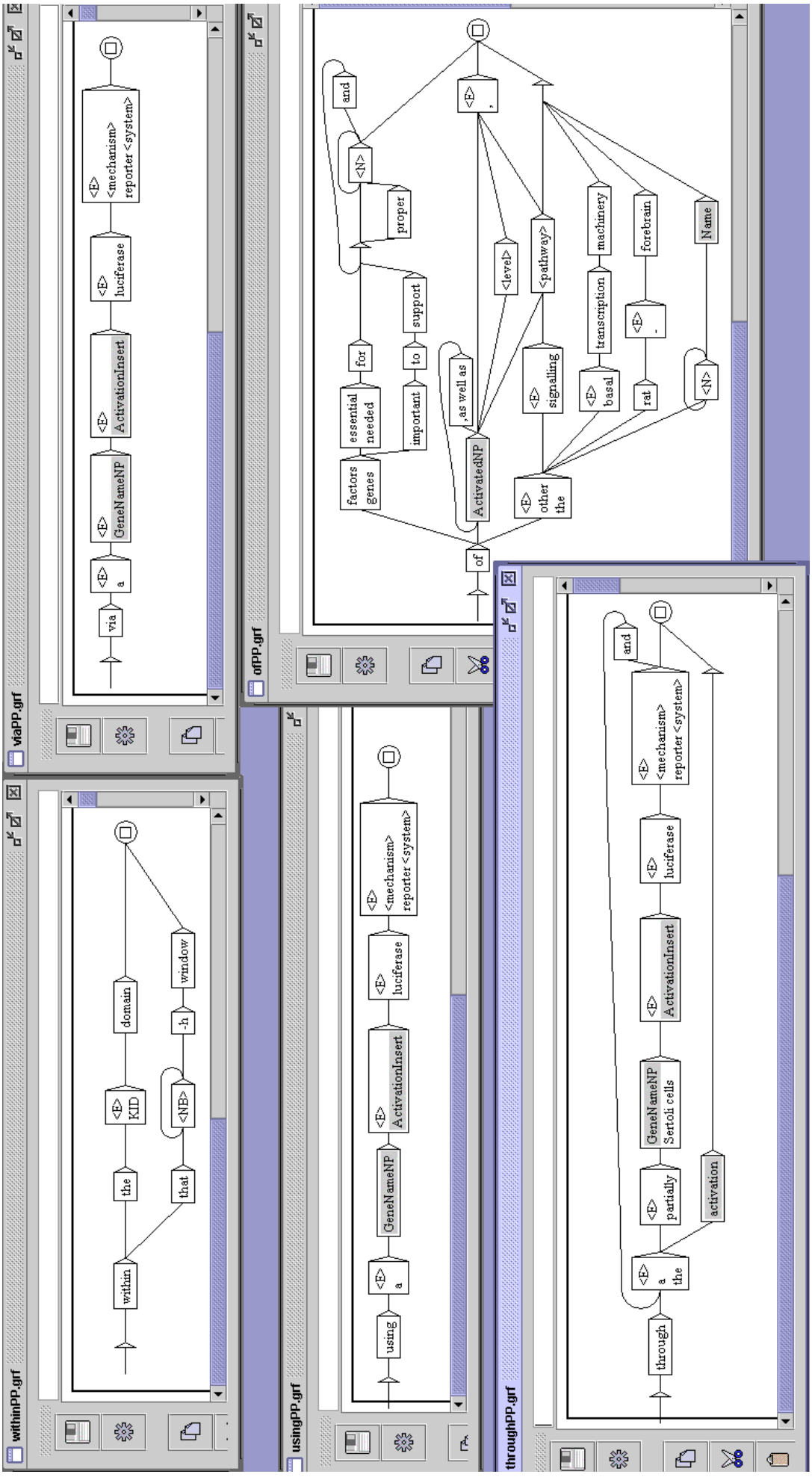

ByMethod.grf
<E> ,
or
by
<E> either
<E> ,
through using
<E> a the
<E> partially
GeneNameNP Sertoli cells
<E> ActivationInsert
<E> luciferase
<E> <mechanism> reporter <system>
and
activation
specific
<E> a the
against
antibody
phosphorylated
form
ofPP
atPP
inPP
<E> inPP toPP
<dimer> <element> <subunit>
<E> like
<E> Name regulatory
<E> <PRO> a its the
as to
binding releasing
immunohistochemistry in situ hybridization
duringPP.grf
during a search
toPP.grf
to
<E> a the
Name
<cell>
Name nucleus
<E> conserved
PreCondition.grf
,
although
because since
only
to determine the specificity of
as expected consequently furthermore here subsequently therefore
addition brief particular rat pancreatic acini summary
in
example
for
point of view
molecular
<a>
from

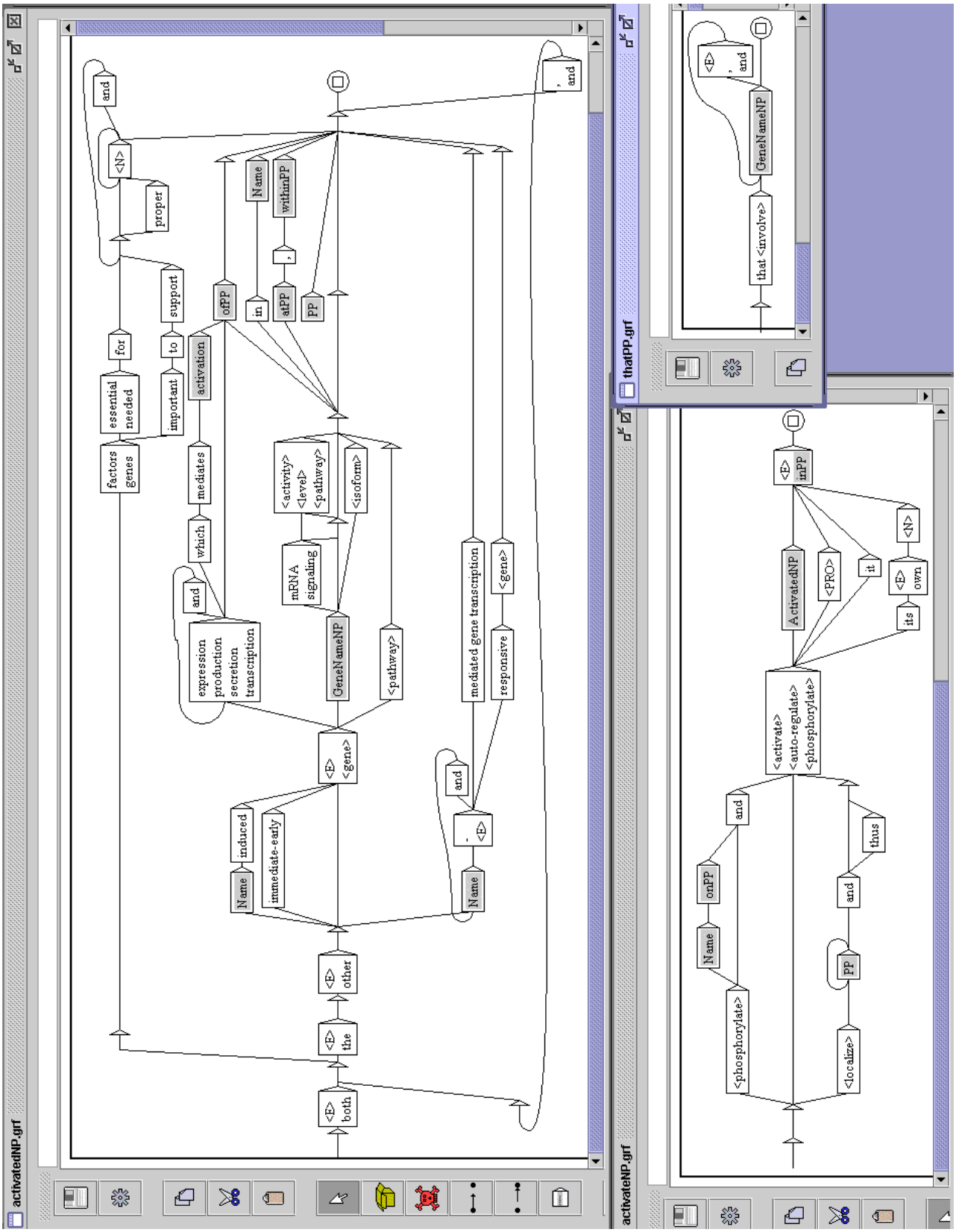

Name.grf
<N>
NOUN=
c-fos
cholesterol level
ERK
hGRP-R
leucine
lithium
rat CeA
<Antigen>
<Aversion>
<Element>
<Gene>
<Glycerol>
<Hormone>
<Kinase>
<Nucleus>
<ProtHum>
<Salt>
<Sequence>
<Substance>
<Technique>
<NB>
-
NWCCKBR
NWCRE
NWDNA
NWEGFR
NWFSH
NWGF
NWGPCR
NWICER
NWLiC1
NWLydigCells
NWMAPK
NWPC12
NWphosphat
NWPI3K
NWPKA
NWPLC
NWSer133
NWSRE
Name.grf
activation.grf
<E> inPP
and
activation
co-activation
expression
phosphorylation
transactivation
transcription
transcription
spermatogenesis
<E>
gene
mRNA
protein
<E> GeneNameNP
inducible
-
Name
<E> transcriptional
<E> sustained
<E> the
inPP.grf
, and
<E> ofPP
cells
regions
<E> SIIA
<E> cancer
<NB>
Name
duodenal
human gastric
HuTu
non-haematopoietic
Sertoli
<E> a the this
in
study
dose
<E> -
dependent
manner
mouse
rat
its
vivo
brain
CeA
insular cortex
promoter

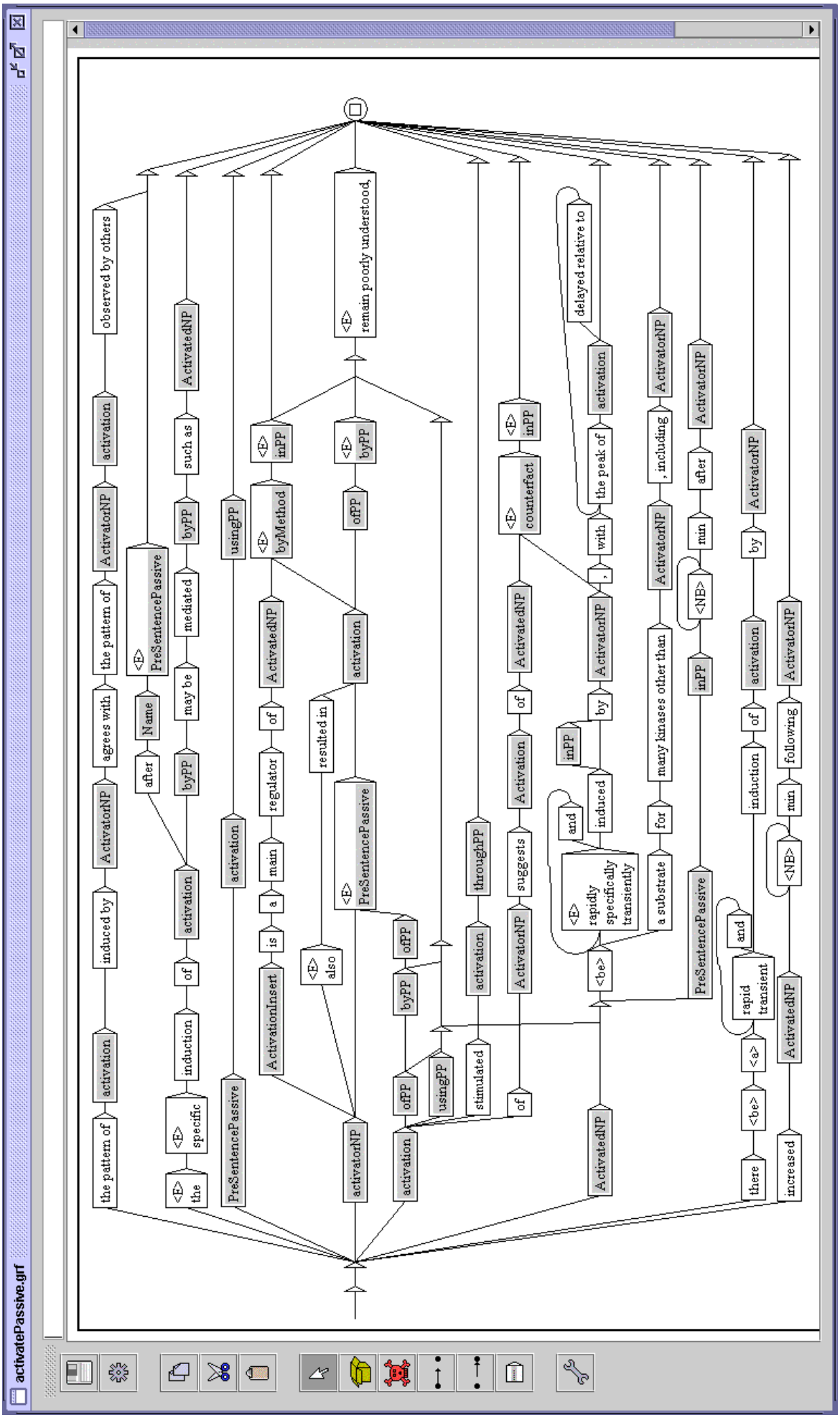

GeneNameNP.grf
the
<A>
activated
intracellular
transcription factor
<PFX>
<E>
acting
<E>
dependent
induced
Name
mitogen
V
stress
-activated
<kinase>
Name
<E>
activated
dependent
signaling
<E>
activity
binding
family kinases
mobilization
<pathway>
receptor
recruitment
site
stimulation
subunits
synthesis
transcription
<E>
factors
, and
PP
BoundToPP
<NB>
mgVkg
or
higher
)
(
ActivationInsert.grf
,
a
the
<NB>
kDa
stimulus
known to produce
Name
zipper
transcription
factor
Name
phosphorylation
ViaPP
<E>
dependent
<E>
GeneNameNP
<E>
transcriptional
<E>
the
activation
co-activation
phosphorylation
transactivation
of
ActivatedNP
GeneNameNP
, as well as
<E>
other
<pathway>
<E>
,
<E>
<occur>

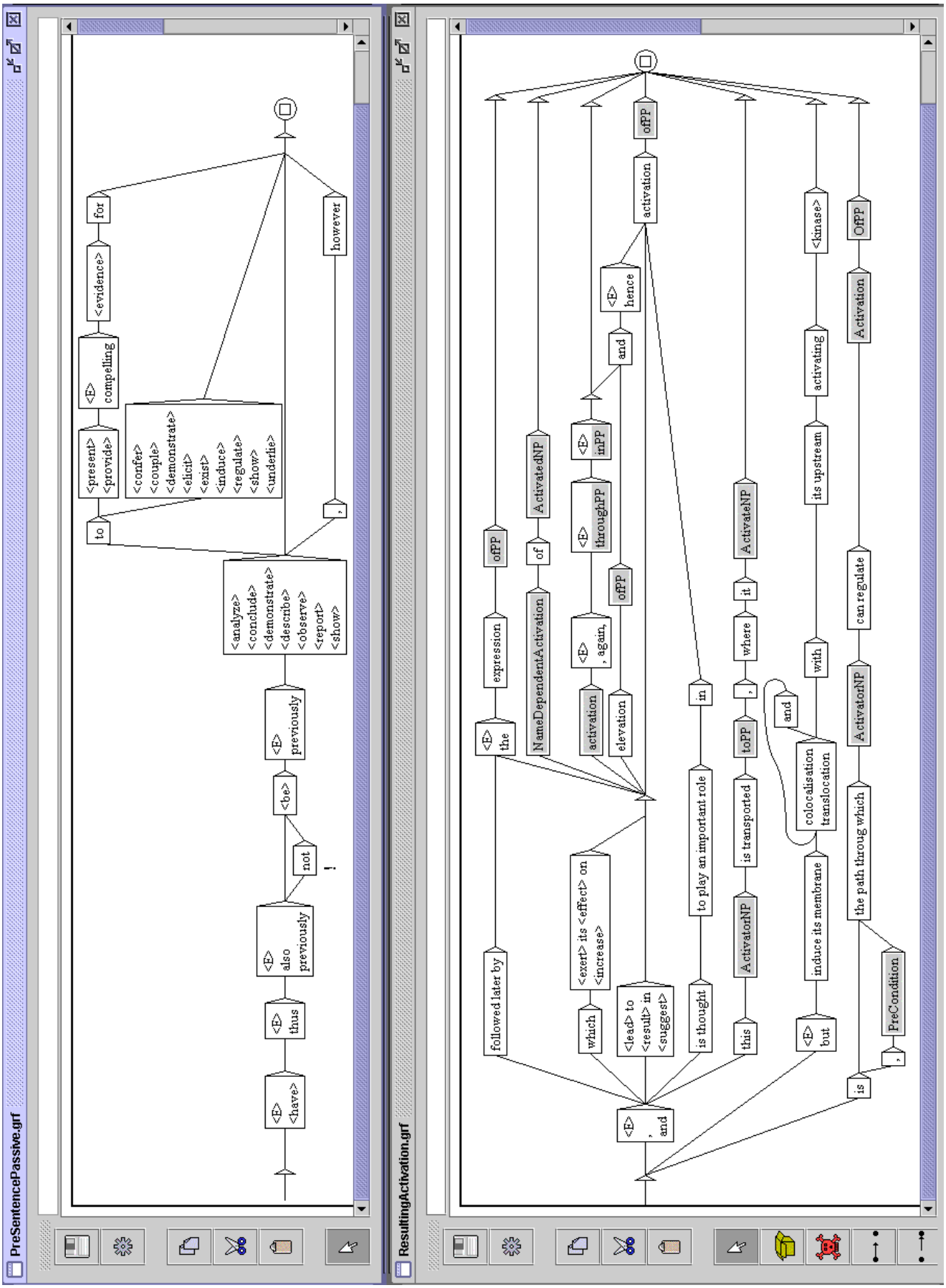

activateActive.grf
activation
atPP
<occur>
inResponseToPP
<E> inPP
is
<E> rapidly
stimulated
byPP
<amplify> <induce> <mediate> <stimulate>
activation
<E> ofPP
forPP
<E> forPP
inPP
withinPP
ActivatedNP
of
expression
<E> gene
toPP
<bind>
withinPP
and
ActivatorNP
toPP
<localize>
<E> ,
where
it
<E> that
activateNP
<E> positively
<do> not directly
!
activatedByNP
activatedNP
<modulate> <require>
<E> strongly
GeneNameNP
could serve as intermediates between
GeneNameNP
and
activation
inResponseToPP
ofPP
<E> byMethod
ActivatorNP.grf
<E> and as well as
<E> ,
PP
Sertoli cells
in
activity <pathway> stimulation transcription
binding -dendent -induced injection
<E> signaling
dependent
recruitment
ofPP
GeneNameNP
sufficient
to produce
GeneNameNP
<E> <A>
activated
induced
Name
recognize
that
the this
doses of elevation in expression of two types of
factors

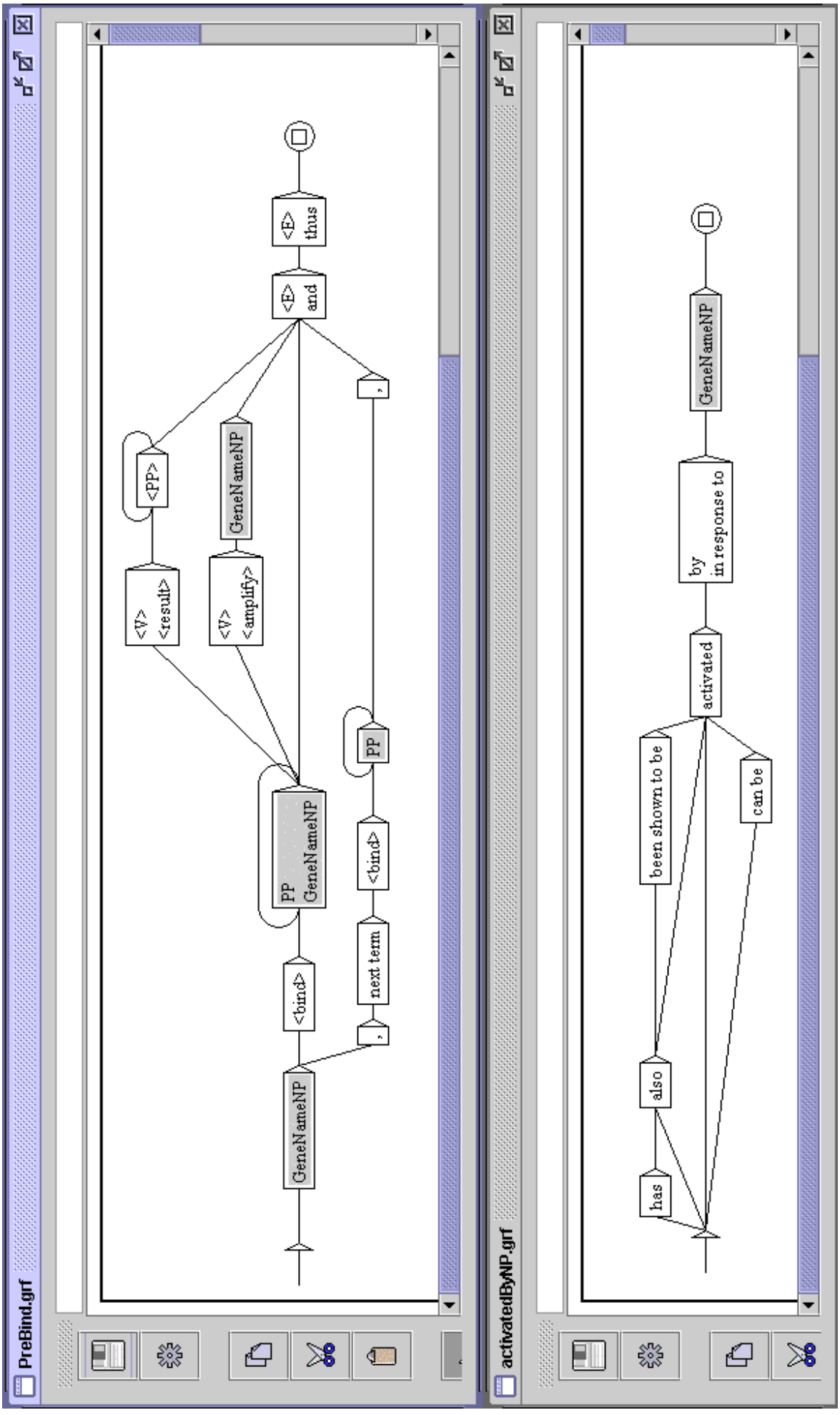

PP.grf
atPP
byPP
inPP
throughPP
usingPP
viaPP
<E>
thatPP
<PREP>
a
its own
the
<N>
GeneNameNP
,
where
<PRO>
<phosphorylate>
GeneNameNP
on
at
GeneNameNP
boundToPP.grf
bound to
its own
promoter
forPP.grf
for
<E>
at least
<NB>
n
onPP.grf
on
GeneNameNP
inResponseToPP.grf
in response to
<E>
a
<E>
GeneNameNP
<E>
luciferase
<E>
<mechanism>
reporter <system>
<E>
ActivationInsert
<E>
GeneNameNP
atPP.grf
at
<E>
a
<E>
GeneNameNP
<E>
ActivationInsert
<E>
luciferase
<E>
<mechanism>
reporter <system>
byPP.grf
by
<E>
a
<E>
GeneNameNP
<E>
ActivationInsert
<E>
luciferase
<E>
<mechanism>
reporter <system>

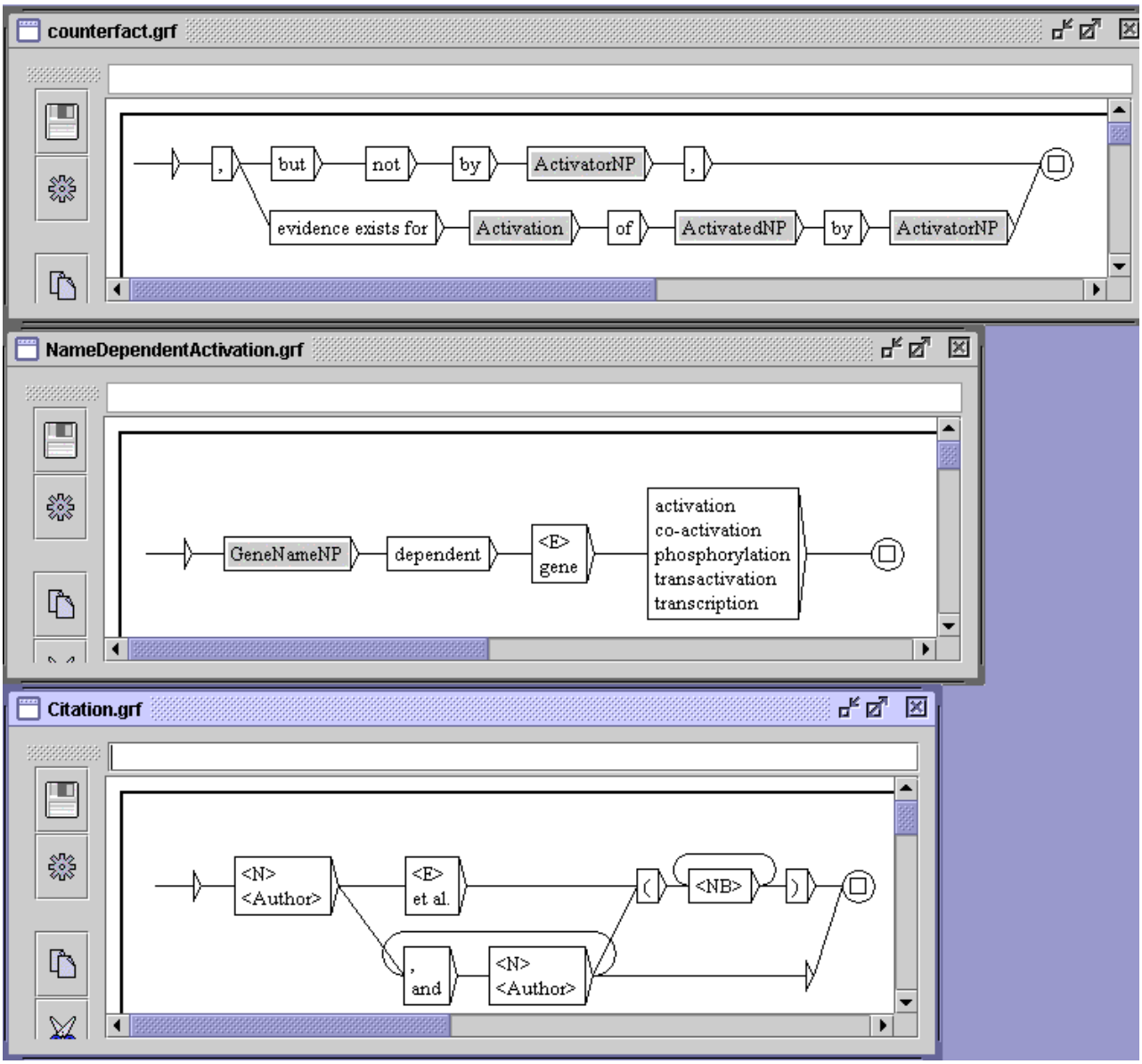

withinPP.grf
within
the
<E> KID
domain
that
<NE>
-h
window
viaPP.grf
via
a
<E> GeneNameNP
<E> ActivationInsert
<E> luciferase
<E> <mechanism> reporter <system>
usingPP.grf
using
a
GeneNameNP
<E> ActivationInsert
<E> luciferase
<E> <mechanism> reporter <system>
ofPP.grf
of
factors genes
essential needed
for
important
to
support
proper
and
<N>
<E> other the
ActivatedNP
,as well as
<E> signalling
<pathway>
<level>
<E> basal
transcription
machinery
rat
<E> .
forebrain
<N>
Name
throughPP.grf
through
<E> a the
<E> partially
GeneNameNP Sertoli cells
activation
<E> ActivationInsert
<E> luciferase
<E> <mechanism> reporter <system>
and


activatedNP.grf
thatPP.grf
activateNP.grf
and
<N>
proper
for
support
to
essential
needed
important
factors
genes
activation
mediates
which
and
expression
production
secretion
transcription
ofPP
Name
withinPP
,
in
atPP
PP
<activity>
<level>
<pathway>
<isoform>
mRNA
signaling
GeneNameNP
<pathway>
<E>
<gene>
induced
Name
immediate-early
<E>
other
<E>
the
<E>
both
, and
mediated gene transcription
<gene>
responsive
and
-
<E>
Name
<E>
, and
GeneNameNP
that <involve>
<E>
inPP
ActivatedNP
<PRO>
it
<N>
<E>
own
its
<activate>
<auto-regulate>
<phosphorylate>
and
onPP
Name
<phosphorylate>
thus
and
PP
<localize>


activatePassive.grf
observed by others
activation
ActivatorNP
the pattern of
agrees with
Name
after
<E>
PreSentencePassive
ActivatedNP
such as
byPP
mediated
may be
byPP
activation
of
induction
<E>
specific
<E>
the
induced by
activation
the pattern of
PreSentencePassive
activation
usingPP
<E>
inPP
<E>
byMethod
ActivatedNP
of
regulator
main
a
is
ActivationInsert
<E>
also
resulted in
activation
<E>
byPP
offPP
<E>
PreSentencePassive
activationNP
offPP
byPP
offPP
activation
usingPP
stimulated
throughPP
activation
ActivatorNP
suggests
Activation
of
ActivatedNP
of
<E>
counterfact
inPP
<E>
remain poorly understood,
delayed relative to
activation
the peak of
with
,
ActivatorNP
by
inPP
induced
and
<E>
rapidly
specifically
transiently
<be>
ActivatedNP
ActivatorNP
, including
ActivatorNP
many kinases other than
a substrate
for
<NB>
min
after
ActivatorNP
inPP
PreSentencePassive
and
rapid
transient
<a>
<be>
there
activation
induction
of
by
ActivatorNP
<NB>
min
following
ActivatorNP
ActivatedNP
increased


PreSentencePassive.grf
<E> <have>
<E> thus
<E> also previously
not
<be>
<E> previously
<analyze> <conclude> <demonstrate> <describe> <observe> <report> <show>
to
<present> <provide>
<E> compelling
<evidence>
for
<confer> <couple> <demonstrate> <elicit> <exist> <induce> <regulate> <show> <underlie>
however
,
ResultingActivation.grf
followed later by
<E> the
expression
ofPP
NameDependentActivation
of
ActivatedNP
<exert> its <effect> on <increase>
which
<lead> to <result> in <suggest>
activation
elevation
<E> , again,
<E> throughPP
inPP
ofPP
<E> , and
to play an important role
is thought
in
ofPP
activation
<E> hence
and
this
ActivatorNP
is transported
toPP
,
where
it
ActivateNP
induce its membrane
colocalisation translocation
and
with
<E> but
its upstream
activating
<kinase>
is
the path throug which
ActivatorNP
can regulate
Activation
OfPP
PreCondition
,


PreBind.grf
GeneNameNP
,
<bind>
next term
<bind>
PP
GeneNameNP
<V>
<result>
<PP>
<V>
<amplify>
GeneNameNP
<E>
and
<E>
thus
,
PP
activatedByNP.grf
has
also
been shown to be
can be
activated
by
in response to
GeneNameNP



In addition to the graphs presented in this appendix, there are around 20 "named entity graphs" (or lexicon grammars), such as the one presented in Figure 1.

The graphs presented in this Appendix are:

- Activate
- ActivateActive
- ActivatedByNP
- ActivatedNP
- ActivateNP
- ActivatePassive
- Activation
- ActiovationInsert
- ActivatorNP
- atPP
- bountToPP
- byMethod
- byPP
- Citation

- Counterfact
- duringPP
- forPP
- GeneNameNP
- inPP
- inResponseToPP
- Name
- NameDependentActivation
- ofPP
- onPP
- PP
- PreBind
- PreCondition
- PreHybridization
- PreRegulation
- PreSentence
- PreSentencePassive
- ResultingActivation
- thatPP
- throughPP
- top
- usingPP
- viaPP
- withinPP

The *lexicon grammars* not shown in this Appendix are:

- CCKBR
- CRE
- DNA
- EGFR
- FSH
- GF
- GPCR
- ICER
- LiCl
- LydigCells
- MAPK
- PC12
- Phosphate
- PI3K
- PKA
- PLC
- Ser133
- SRE