Universiteit Leiden
Opleiding Informatica

Constructing Petri net models from biological literature using structured annotation

Name: Anneloes Louwe

Date: 28/08/2016

1st supervisor: Jetty Kleijn
2nd supervisor: Fons Verbeek

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Constructing Petri net models from biological literature using structured annotation

Anneloes Louwe

Abstract

Construction of Petri net models based on biological literature has been a significant focus of recent efforts in modelling biology. However, this process has so far almost exclusively relied on expert knowledge about both biology and Petri nets. We aim to investigate possibilities for support provided by literature annotation when constructing Petri net models. On basis of an existing text annotation scheme for biological literature, we define a set of rules on how to translate each of these annotation types directly into Petri nets. These rules are then implemented in our newly developed tool, Woodstock, to automate the construction of Petri nets. The tool is evaluated using a set of annotated PubMed abstracts, demonstrating that generated Petri nets can be used to assist in the construction of Petri nets.

Keywords

bio-modelling, biological literature, text mining, annotation, BRAT, MLEE, Petri net, Snoopy, Woostock

Table of Contents

List of Tables

List of Figures

1 Introduction

Petri nets (PNs) [4] are named after Carl Adam Petri who proposed a graphical and mathematical method for the modelling and analysis of concurrent systems. Over the last few years, Petri nets are increasingly used to model biological systems such as molecular networks.

Biology is the study of life and all living organisms. Biological research ranges from behaviour of organisms (e.g. humans or bacteria), to anatomical systems (e.g. the central nervous system), organs (e.g. heart or skin), multi-tissue structures (e.g. blood vessels or lymph nodes), tissues, cells (e.g. skin cells or cancer cells), cellular components, genes, DNA, proteins and –last but not least– to drugs.

However, most biological behaviour can not be explained by looking at the behaviour of each individual component. Behaviour exhibited by tissues differs from the behaviour of its individual cells and the behaviour of cells differs from the behaviour of its cellular components. Those properties *emerge* from the functioning as a system, making it more difficult to study and to understand the details of biological behaviour.

Many biological systems have no real boundaries and often work together to accomplish tasks. The present understanding of many biological processes remains vague and incomplete. Modelling what we *do* know, however, can help us discover what we do not know. Modelling of biological systems using all sorts of approaches –mathematical, computational, algorithmic [5, 6, 21]– is becoming increasingly important.

The construction of 'biological' Petri net models can lead to a better understanding or to new insights, which makes it of vital importance especially in the bio-medical domain. However, modelling of biology is extremely complex and requires profound literature study and expert knowledge of both Petri net theory and the biological subject of interest. Finding or educating such experts can be extremely difficult since biology and computer science/mathematics often are two completely separated fields of research. Therefore we aim to investigate whether using text mining could minimize the requirement of expert knowledge and human effort in the construction process of biological Petri net models.

We started with a literature review of existing approaches for the construction of biological Petri net models based on text annotation, but were unable to find existing approaches. Thus we tried multiple annotation tools and techniques to develop and test three different approaches.

The first approach involved the online annotation tool Egas [2] with its build-in annotation features. However, –at that time– this online tool had problems connecting to the server, which resulted in various bugs, and the user-interface

made the manual annotation of text a time-consuming process. We also discovered that its automatic annotation features (entity recognition and basic relation extraction) did not extract sufficient information for the construction of Petri nets.

For our next approach we turned to BRAT [25], which initially offered no pre-defined annotation scheme nor had automatic annotation tools integrated. Various existing annotation schemes for biological events were studied. However, those schemes all seemed too complex to find a simple systematic approach for the translation to Petri nets. Then we created a new annotation scheme consisting of just two entity types (*subject* and *object*) and one event (*predicate*) for the extraction of triplets (Fig. 1).

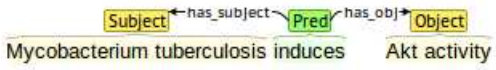


Fig. 1: Triplet extraction example

We tested the technique by manually annotating triplets followed by visualization of the triplets as simple graphs using the DOT language [18] and creating a Petri net in Snoopy similar to the DOT net. During this process it became apparent that only a few biological events can be described as triplets, and therefore this approach resulted in highly incomplete Petri nets. As this approach was mainly manual –manual annotation and manual Petri net construction– it was also very time-consuming and still relied on a significant amount of expert knowledge. Hence, a rather different approach was tried, which will be described in more detail in this thesis.

First, we will look into the extraction of information from literature using text mining techniques (Chapter 2.2) and tools (Chapter 2.4), in particular document annotation using a fixed annotation scheme. An example of annotated text is shown in Fig. 2. The annotation scheme that was used enabled the annotation of specific information, in this case “cats eat mice”. Furthermore, the annotations add new information to the text by indicating which animal is the prey and which is the predator in this relationship.

Next, we will use the Multi-Level Event Extraction (MLEE) annotation scheme [22] for the extraction of relevant biological entities, relations and events across multiple levels of biology (e.g. molecular, cellular). Annotation using this annotation scheme results in a set of highly structured information; on one hand we will have biological entities (e.g. cells, proteins) and on the other hand we will have *event triggers* (e.g. binding, growth). This is broadly comparable to

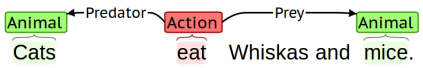


Fig. 2: Example of structured annotation

Petri nets, which are bipartite graphs consisting of nodes representing passive elements (*places*) and nodes representing active components (*transitions*) and thus it could potentially be possible to create Petri nets exclusively based on annotations. Figure 3 shows an example of a Petri net. Additional information about Petri nets can be found in Chapter 3.

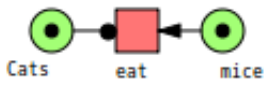


Fig. 3: Example of a Petri net

Next, we will use the MLEE *corpus* (a set of annotated literature) to analyse multiple occurrences of each annotation type in the MLEE annotation scheme. Then, based on this analysis and basic knowledge about biology and Petri nets, we will define a set of rules on how to translate all possible annotations to places, transitions and the edges between them (Chapter 4.1). These rules together with rules about naming, synonyms, handling of duplicates, and so on, will form the foundation for a new conversion tool, Woodstock (Chapter 4.2). The implementation details of this tool can be found in Chapter 4.2 and in Appendix 7.2.

Finally, we will evaluate Woodstock (Chapter 5) and reconsider our research question in Chapter 6: Can text annotation assist in the construction of Petri net models?

2 Biological text mining

An important step in text mining is annotation. Annotation of text adds information to a document collection that can later be exploited for text mining purposes [23]. Biological text annotation can be divided into three major subtasks; named entity recognition (NER), relation extraction and event extraction. These subtasks are often utilized as a first step in other biomedical text mining tasks such as summarization, question answering, literature based discovery or, in our case, the construction of Petri net models.

2.1 Annotation

Named Entity Recognition NER is the task of identifying *entities*, which are occurrences of biological or medical terms in text, such as drug and protein names. NER is typically a multi-step process that, in addition to the identification of the entity, involves assigning it to a predefined class or category.

As an example consider the annotated text fragment depicted in Fig.4. This fragment of text shows three biological entities (*tumors, mouse, lung*) annotated and classified as `Pathological formation` (PathF), `Organism` (Org) and `Organ` respectively. Commonly, abbreviated labels are used above smaller annotated text spans.

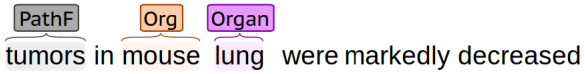


Fig. 4: Named Entity Recognition (NER)

Relation extraction The most simple associations among biomedical entities are binary, involving only a pair of entities. Therefore, the goal of *relation extraction* is to identify *relations*, which are occurrences of particular types of relationships between pairs of given entities. The relation between proteins, protein-protein interaction (PPI), is by far the most widely researched topic in text mining. [14].

Event extraction Since biological relationships often involve more than two entities, relation extraction alone is not sufficient. These types of biological relationships can be annotated using *event extraction*. An *event* consist of an *event trigger* which is typically a verb, and one or more *event arguments*, which are

other entities or other (nested) events participating in the event. As an example consider Fig. 5. In this example, *inhibited* and *formation* are annotated as *event triggers*, along with their *event arguments* indicated by black and green arrows stating the role of the argument. Biological events can be extremely complex and therefore automatic event extraction remains an ambitious task.

Fig. 5: Event extraction

2.2 Corpora

Although text is the primary resource for text mining, it can be useful to utilize already annotated collections of text, which are publicly available. Throughout the years, many of these specialized *corpora* have been created, containing collections of abstracts or full-text articles from a variety of resources and using a variety of annotation techniques.

The GENIA corpus [12] is currently the most thoroughly annotated collection of PubMed abstracts and its annotations include biomedical concepts, events, disease-gene associations, pathways, meta-knowledge [27] and more. Its annotation scheme was used in the BioNLP 2009 Shared Task [13] and applied in numerous corpora and methods introduced since. Its focus lies almost exclusively on molecular-level processes.

Additionally, a variety of smaller corpora have been developed focussing on other levels of biology (e.g. Anatomical Entity Mention (AnEM) corpus [19] focussing on the anatomical and organism level) or with specific intentions (e.g. CellFinder [24], which aims to establish a central repository for stem cells).

Recently, researchers at the National Centre for Text Mining (NaCTeM) combined the GENIA annotation scheme with the AnEM annotation scheme, creating the Multi-Level Event Extraction (MLEE) corpus [22] for the extraction of biological events across all levels of biological organisation. This corpus consists of 262 PubMed abstracts on angiogenesis (blood vessel development) containing a total of 6677 annotations.

2.3 MLEE annotation scheme

The annotation types of the MLEE annotation scheme are shown in Fig. 6: (a) 16 entity types, (b) 2 relation types, (c) 32 event types and (d) 2 binary modifiers. Its entity types range from molecular entities to organs and organisms. Its relation types are not used to extract typical biological relations, such as protein-protein interaction. Instead, they are used to indicate synonyms and abbreviations (*equivalence*-relations) and to combine fragments of entity names which are split up by other text (*fragment*-relations). However, fragment relations are redundant in some annotation tools as some provide the option to annotate entities consisting of multiple fragments directly as one entity in the annotation file.

The event types are used to annotate biological processes together with their cause and/or other information. Binary modifiers mark events as *negated* or stated in a *speculative* context. Details about the annotation scheme, including all events and their arguments, are listed in [22, Supplementary Data – Chapter 1].

2.4 Annotation tools and annotators

Although text can be annotated using pen and paper, annotations are often created using *annotation tools*, enabling the user to self-annotate the text and/or to use task-specific *annotators* for the automatic annotation of information.

BRAT [25] is a web-based solution for in-line annotation of documents. It provides concept normalization features, automatic services integration, search capabilities and document comparison. However, the configuration of annotation tasks (e.g. annotation scheme, short-cuts or color set-up) is controlled by text-based configuration files, which may be difficult for new users to work with. Therefore, Egas [2], a roughly similar tool for in-line annotation was created, which aims to provide highly usable interfaces for both manual and automatic annotation of biological literature, easy access to annotation task settings, and real-time collaboration and conversation functionalities.

2.5 Annotation files

Annotations created using annotation tools and annotators are often stored separately from the original text in a stand-off text file. The format of these *annotation files* was first defined in the BioNLP'09 Shared Task [13]. In this file all annotation data is stored separated by white space and line breaks. A small example can be found in Fig. 7. The first three lines each correspond to a text span, containing a unique identifier, its type, its location (e.g. T1 is located on first four characters of the text file) and its text. The fourth line stores the event, with an identifier, its type and its arguments.

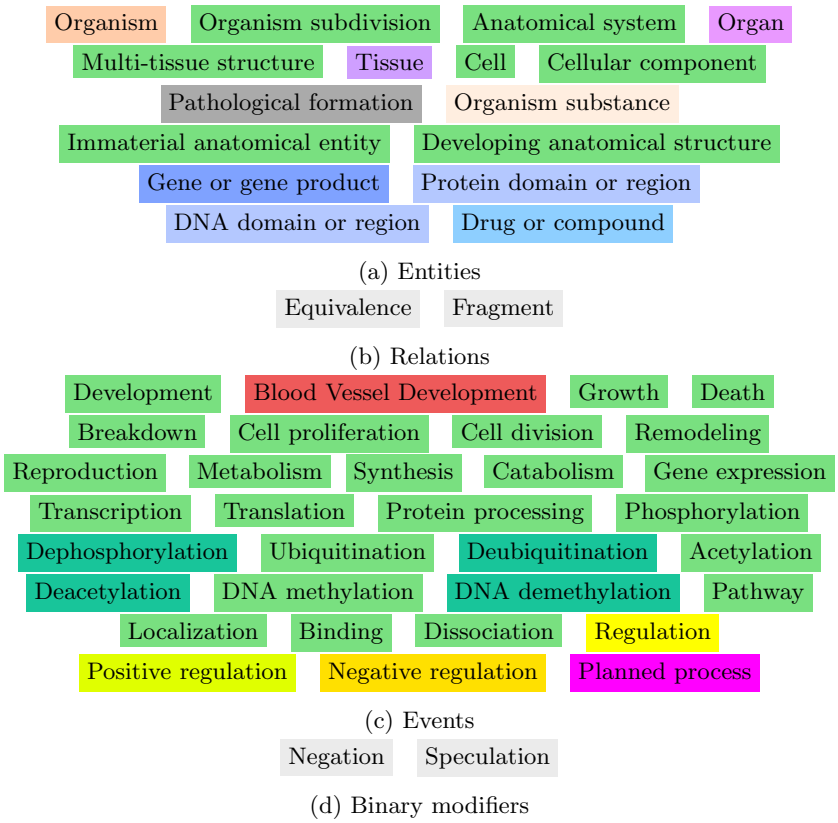


Fig. 6: MLEE annotation scheme

```
1 T1   Animal 0 4   Cats
2 T2   Animal 21 25   mice
3 T3   Action 5 8   eat
4 E1   Action:T3 Predator:T1 Prey:T2
```

Fig. 7: Annotation file corresponding to the annotation in Fig. 2.

3 Petri Nets

In this section we will start with explaining the basic graphical and mathematical definitions of standard Petri nets and Extended Petri nets. Next we will discuss tools for the design, animation and automatic generation of Petri nets.

3.1 Petri net basics

Petri nets (PNs) are directed bipartite graphs with two types of nodes; places and transitions. Places and transitions are connected by weighted edges. A place is an *input place* of a transition if it has an edge running to the transition and a place is an *output place* of a transition if an edge runs from the transition to the place. In addition, the definition of a PN includes the specification of an *initial marking*, which allocates zero or more *tokens* to each place. These token distributions ('markings') represent the current state of the system.

A transition *can fire* (or: *is enabled*) if all its input places contain at least the required amount of tokens, which is defined by the weight of the edges. When a transition fires, it consumes tokens from its input places and produces tokens in its output places. Firing is atomically and does not consume time. Multiple transitions can fire at the same time, making Petri nets non-deterministic and suitable for the modelling of concurrent systems, such as biological networks. See references [1, 4] for a more extensive introduction to Petri nets.

As an example consider Fig. 8. It shows a basic Petri net. The places are drawn as circles, the transitions as squares and the tokens as black dots. If a place contains more than 3 tokens, the black dots are replaced by a number indicating the number of tokens. Figure 8a shows the initial marking in which transition `t3` is *enabled*, meaning its input place `p3` contains enough tokens. Figure 8b shows the PN after firing `t3`. Now place `p1` contains one token, enabling `t1` to fire. Figure 8c shows the PN after firing `t1`: place p2 contains 4 tokens and `p3` contains 2 tokens. Transition `t2` is enabled.

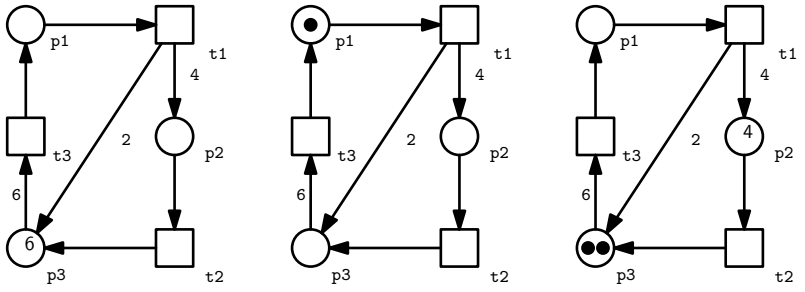


Fig. 8: Firing rule

The formal definition of Petri nets, input and output places and the firing rule can be found in Definition 1, 2 and 3 respectively.

Definition 1 (Standard Petri net). *A standard Petri net is a quadruple* $N = (P, T, f, m_0)$*, where:*

- *P, T are finite, non-empty, disjoint sets. P is the set of places. T is the set of transitions.*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_0$ *defines the set of directed edges, weighted by non-negative integer values.*
- $m_0 : P \to \mathbb{N}_0$ *gives the initial marking*

Definition 2 (Preset and Postset).

- *Preset, set of input places:* $\bullet x := \{y \in P \cup T | f(y, x) \neq 0\}$
- *Postset, set of output places:* $x\bullet := \{y \in P \cup T | f(x, y) \neq 0\}$

Definition 3 (Firing Rule). *Let* $N = (P, T, f, m_0)$ *be a Petri net::*

- *A transition is enabled in marking m if* $\forall p \in \bullet t : m(p) \geq f(f, t)$*, else it is disabled.*
- *A transition t, which is enabled in m, may fire.*
- *When t in m fires, a new marking m' is reached:* $m[t\rangle m'$*, with* $\forall p \in P : m'(p) = m(p) - f(p, t) + f(t, p)$*.*

3.2 Extensions

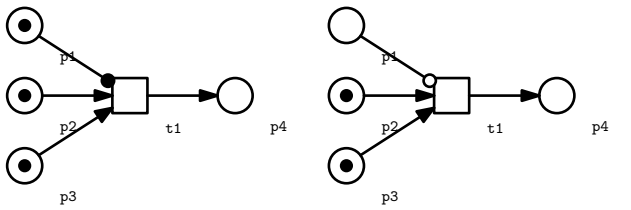
Many extensions to ordinary Petri nets exists, including additional types of edges. Two common types are read edges (also called read arcs) and inhibitor edges (also called inhibitor arcs).

If a read edge is used to connect a place with a transition, the transition is enabled if the place and all other places connected with transition via standard

edges are sufficiently marked. By firing the transition, tokens are not deleted from this place.

If an inhibitor edge is used to connect a place with a transition, the transition is enabled if the place is *not* sufficiently marked, meaning the amount of tokens is less than the edge weight, and all other places connected with the transition via the standard arc are sufficiently marked. The amount of tokens on this place is not changed if the transition fires. Inhibitor edges can not be reduced to the standard edges.

Petri nets containing read and inhibitor edges are called *Extended Petri Nets*. Figure 9a illustrates a Petri net containing a read edge. Read edges are drawn as an edge with a filled dot at the end. After firing the transition, tokens are removed from places `p2` and `p3`, but not from `p1`. Figure 9b illustrates a Petri net containing an inhibitor edge. Inhibitor edges are drawn as an edge with an empty dot at its end. After firing `t1`, place `p1` remains empty and tokens are removed from `p2` and `p3`.



(a) PN with read edge (b) PN with inhibitor edge

Fig. 9: Additional types of edges

3.3 Tools

Nowadays numerous computer tools exist for the design, simulation and analysis of Petri nets. The tool Snoopy [8] was developed by the University of Technology in Cottbus, Dep. of Computer Science, "Data Structures and Software Dependability". It supports the design and animation of Petri nets, in particular for modelling biological systems.

3.4 Automatic Petri net generation

Petri net construction for biological systems can be highly time-consuming and labour-intensive. Some assistance can be provided by tools for the semi-automatic

or automatic construction of Petri net models based on knowledge sources such as literature and databases. The Kyoto Encyclopedia of Genes and Genomes (KEGG) database [11] is a valuable resource for Petri nets, containing structured information about genes and biological pathways. A web-based tool, MoVisPP [3], was developed for the automatic generation of biochemical Petri net models for all pathways in the KEGG database, supplemented with information from other databases. To the best of our knowledge, similar tools for the generation of Petri nets based on literature have not yet been developed.

4 Methods and Results

We started with the installation of a standalone BRAT environment and configured it to be able to view and create MLEE structured annotations by adapting its annotation configuration file (Appendix 7.1). In addition, the visual configuration file from the BRAT online MLEE demonstration corpus [10] was adopted to ensure the annotations in our BRAT environment will look the same as in the original MLEE corpus.

4.1 Annotation to PN conversion

Next, rules were defined for the conversion of annotations to Petri net elements. Altogether, these rules will create Extended Petri Net models, which can be viewed and animated in Snoopy.

Entities Spans of text annotated as entity will be places in the Petri net. The text span will be its name, but all characters other than alphabetical characters and digits will be replaced by an underscore. The place will have the same color as the annotation in BRAT and the place width will be equal to the length of its name, to avoid overlapping names in the Petri net. None of the places will be marked with tokens. Entities with the same name will be merged as one place.

Relations The two possible types of relations are Equivalence and Fragment relations. Since long node names may overlap with other elements in the Petri net, the shortest name in the Equivalence relation is chosen and will be used as node name throughout the Petri net instead of the longer name. Fragment relations will be ignored in the Petri net construction.

Events The biggest challenge of creating rules lies in the conversion of events to nodes and edges in Petri nets. All event triggers will be transitions in the PN, with colors corresponding to their type, and their arguments will be pre- or post places connected by standard edges (STD), read edges (RD) or inhibitor edges (INH). BRAT has advanced search functionalities to search specific annotated events and entities. Using this feature, multiple instances of each event type in the MLEE corpus were randomly selected. Next, we constructed possible Petri net representations for each instance by hand and analysed these to determine which representation was best in the majority of the instances for each event type. This representation, determining the edge type and whether the argument is an pre-place or an post-place, was chosen and used to create a conversion rule to convert the annotation to a Petri net. Table 1 lists all events and the conversion rules for their arguments.

Event type \ Argument type	Theme	Cause	Site	From-Loc	At-Loc	To-Loc	Instrument	Participant
Development	$POST_{STD}$							
Blood vessel dev.	$POST_{STD}$				⊠			
Growth	PRE_{RD}							
Death	PRE_{STD}							
Breakdown	PRE_{STD}							
Cell proliferation	PRE_{RD}							
Cell division	PRE_{RD}							
Remodeling	PRE_{RD}							
Reproduction	PRE_{RD}							
Metabolism	PRE_{STD}							
Synthesis	$POST_{STD}$							
Catabolism	PRE_{STD}							
Gene expression	$POST_{STD}$							
Transcription	$POST_{STD}$							
Translation	$POST_{STD}$							
Protein processing	PRE_{STD}							
Phosphorylation*	PRE_{STD}		⊠					
Dephosphorylation*	PRE_{STD}		⊠					
DNA methylation	PRE_{STD}		⊠					
Pathway								⊠
Localization	PRE_{RD}			⊠	⊠	⊠		
Binding	PRE_{STD}		⊠					
Dissociation	$POST_{STD}$		⊠					
(+)Regulation (entity participant)	PRE_{STD}	PRE_{RD}	⊠					
(+)Regulation (event participant)	$POST_{STD}$	PRE_{STD}	⊠					
–Regulation (entity participant)	PRE_{STD}	PRE_{RD}	⊠					
–Regulation (event participant)	$POST_{INH}$	PRE_{STD}	⊠					
Planned process	PRE_{RD}						PRE_{RD}	

Table 1: **Conversion rules for events**. Each cell in the table shows whether the argument will be a pre- or post place and which edge type will be used to connect. The conversion of a regulatory event will be based on the type of its participant(s). ⊠: argument will not be converted to PN. *Acetylation/Deacetylation and Ubiquitination/Deubiquitination will be converted similarly to Phosphorylation/Dephosphorylation.

Binary modifiers The binary modifiers (Negation and Speculation) will not cause a functional difference in the Petri net, but they will be shown visually in the PN enabling users to change the Petri net manually according to the effect of the modifier. Negated events will be transitions with a red border and events stated in a speculative context will have a grey-coloured node name.

4.2 Woodstock: automatic conversion tool

Manual Petri net construction based on the conversion rules stated in the previous section would be highly time-consuming and error-prone. It would be easier to use a tool to automatically translate an annotation file to an Extended Petri net file. As such tool did not yet exist we took up the development of a new tool, which was named Woodstock after the bird Woodstock, Snoopy's best friend in the comic Peanuts.

Woodstock is an online tool implemented in PHP [15] and therefore platform-independent and supported by every (modern) webbrowser. The PHP files can be found in Appendix 7.2. Its input files have to be *annotation files* (.ann) belonging to a single document annotated using the MLEE annotation scheme. The annotations have to be stored in the format described in subsection 2.5.

Woodstock reads the input file line by line, meanwhile storing all entities, events and other information in arrays. After reaching the end of the input file, all information is post-processed (e.g. to merge duplicate entities/places) and then a Snoopy Extended Petri Net (.spept) file is constructed. These files are written in an xml-based language. An example is shown in Fig. 11.

To view the Petri net, the spept-file needs to be downloaded and then opened in Snoopy. However, Woodstock does not contain an algorithm for the positioning of nodes. Instead, all nodes will appear stacked in the upper-left corner. In Snoopy, the desired layout can be chosen by pressing `Ctrl + L`.

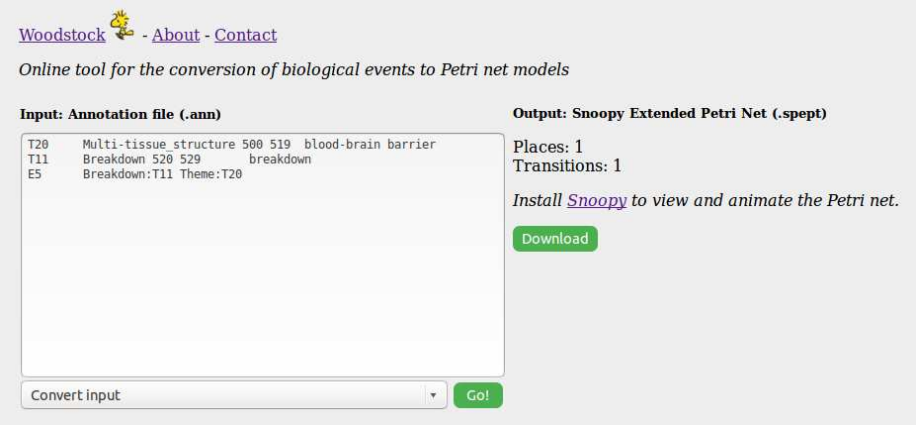


Fig. 10: Woodstock

```
1       <nodeclass count="0" name="Place">
2         <node id="2" net="1">
3           <attribute name="Name" net="1">
4             <![CDATA[Cats]]>
5             <graphics count="1">
6               <graphic grparent="-2" xoff="0" yoff="15"  net="
                    1" show="1" state="1" />
7             </graphics>
8           </attribute>
9           <attribute name="ID" net="1">
10            <![CDATA[2]]>
11            <graphics count="1">
12              <graphic grparent="-2" net="1" show="0" state="1
                    " pen="0,0,0" brush="255,255,255"/>
13            </graphics>
14          </attribute>
15          <graphics count="1">
16            <graphic id="-2" x="20" y="20" net="1" show="1" w=
                  "20" h="20" state="1" pen="0,0,0" brush="255,
                  204, 170"/>
17          </graphics>
18        </node>
19      </nodeclass>
```

Fig. 11: **Snoopy Extended Petri Net file**. This is a fragment of a spept-file created by Woodstock and creates a place named 'Cats' in Fig. 3. The full spept-file creating to the Petri net in Fig. 3 can be found in Appendix 7.3.

5 Evaluation

Now a conversion tool has been developed, the generation of Petri nets can be analysed. Petri nets of various sizes, originating from different articles, will be investigated and animated.

5.1 Annotation selection

Abstracts in MLEE corpus containing the most *event* annotations were selected and from those only the titles were used for the generation of Petri nets. The first four titles are depicted in Fig 12. All these titles contain regulatory events, which is not a coincidence since regulatory events account for approximately 45% of all event annotations in the MLEE corpus.

Figure 12 also demonstrates not all information is annotated, which may result in Petri nets lacking essential information. However, the aim of both text mining and modelling is the extraction/modelling of relevant information from text rather than all information.

5.2 Petri net generation

To generate a Petri net for one single sentence, the full annotation file was loaded into Woodstock together with a line indicating a start- and end offset value to select the title (e.g. O 0 90).

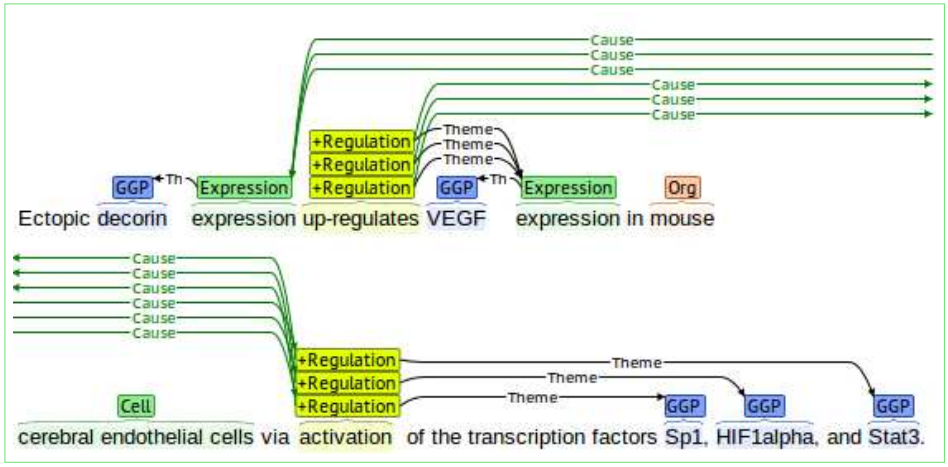
After the Petri nets had been successfully generated using Woodstock, the Sugiyama [26] layout was applied using Snoopy. The nets are depicted in Fig. 13, demonstrating that the Petri nets look as defined by the rules in section 4.1. Generated Petri nets often contain auxiliary places (drawn without color nor name) between a pair of transitions, which is the result of nested events directly connected to other event triggers.

In addition, some full abstracts were randomly selected from the MLEE corpus to analyse the generation of bigger Petri nets. The connectivity in these nets is caused by both basic argument relations (e.g. `Theme`, `Cause`) and by merging places with the same name. The example in Fig. 14 consists of three unconnected/isolated Petri nets. The cause of this isolation can be missing information in text, unannotated information or unconverted annotations (e.g. `Site` arguments).

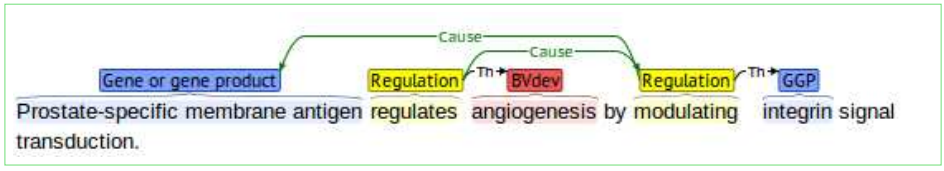
5.3 Petri net analysis

Next, the behaviour of several generated Petri nets was analysed. Snoopy was used to add tokens and to animate the nets. The behaviour of the Petri nets corresponded in most cases with the behaviour described in the text.

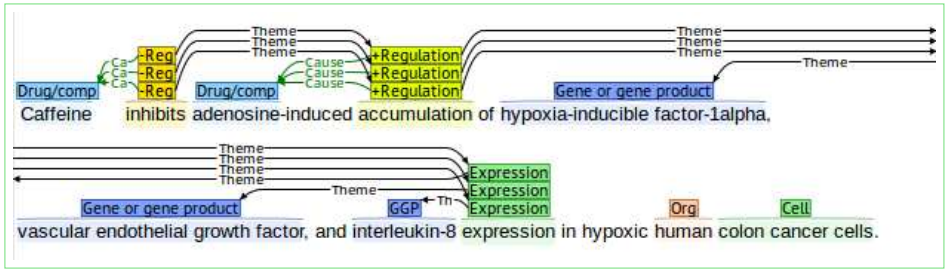
As an example consider Fig. 13c. It shows the generated Petri net for following description: *"Caffeine inhibits adenosine-induced accumulation of hypoxia-inducible factor-1alpha (HIF-1alpha), vascular endothelial growth factor (VEGF), and interleukin-8 (IL-8) expression in hypoxic human colon cancer cells."* In other words, adenosine will cause an increase in HIF1-alpha, VEGF and IL-8, but this will be inhibited (prevented) if caffeine is present. In Snoopy the net was once animated without caffeine in its initial marking and once with caffeine. After ten steps the amount of tokens in places `hypoxia_inducible_factor_1alpha`, `VEGF` and `IL_8` was increased in the net lacking caffeine (Fig. 15a), while this did not increase in the net with caffeine (Fig. 15b). The animated Petri net can be found in Appendix 7.4.



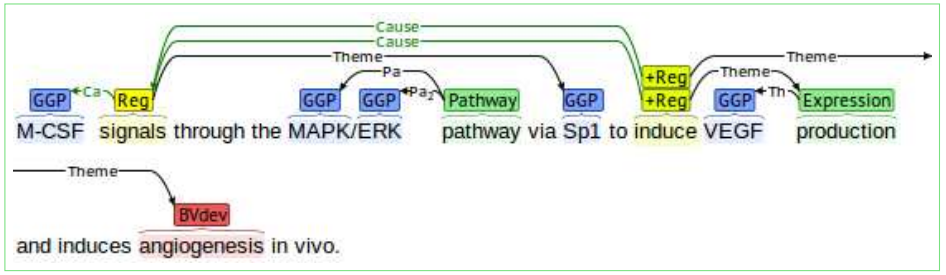
(a) PMID-18021292



(b) PMID-16809768



(c) PMID-17488804



(d) PMID-18852899

Fig. 12: Annotations from MLEE-corpus

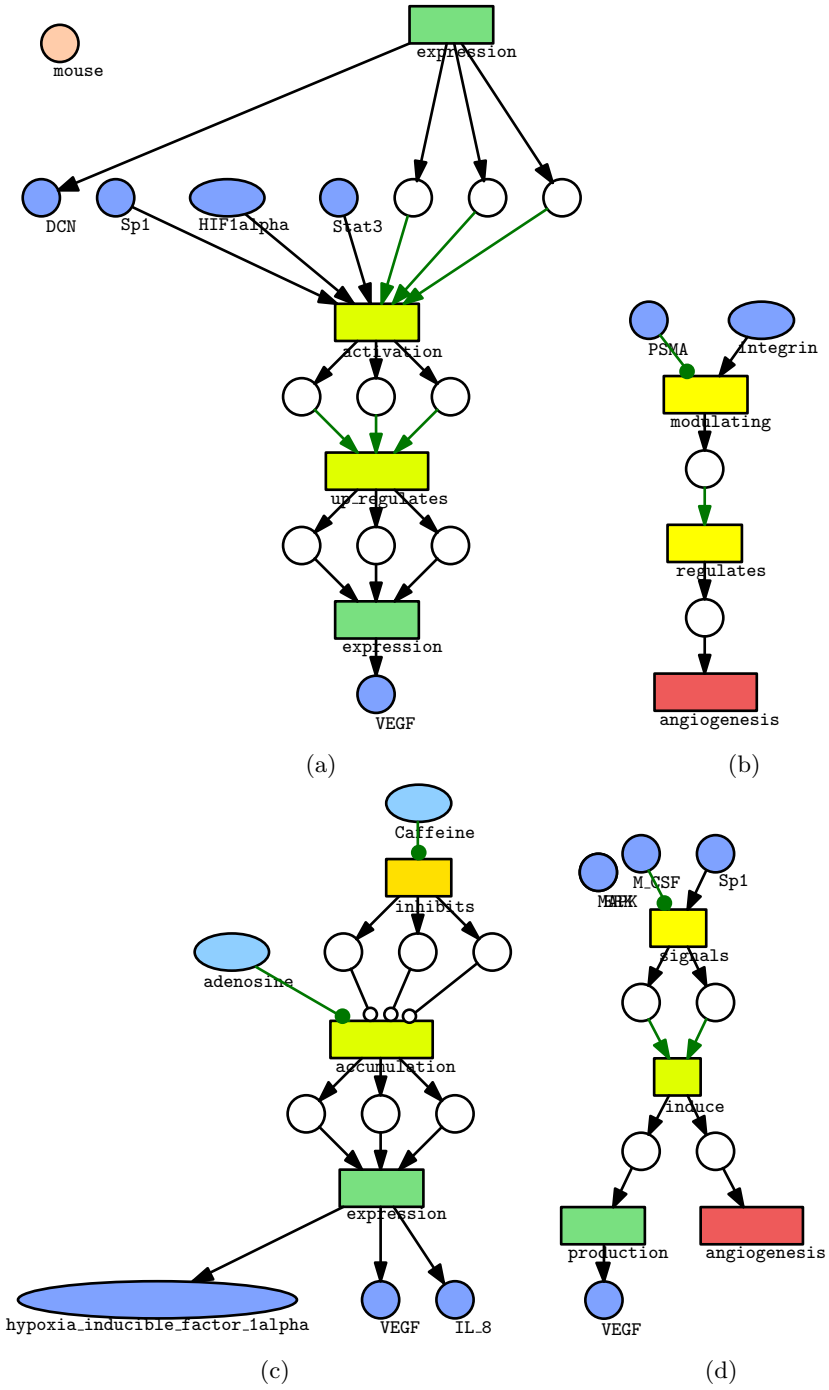


Fig. 13: Generated Petri nets for the annotations shown in Fig. 12.

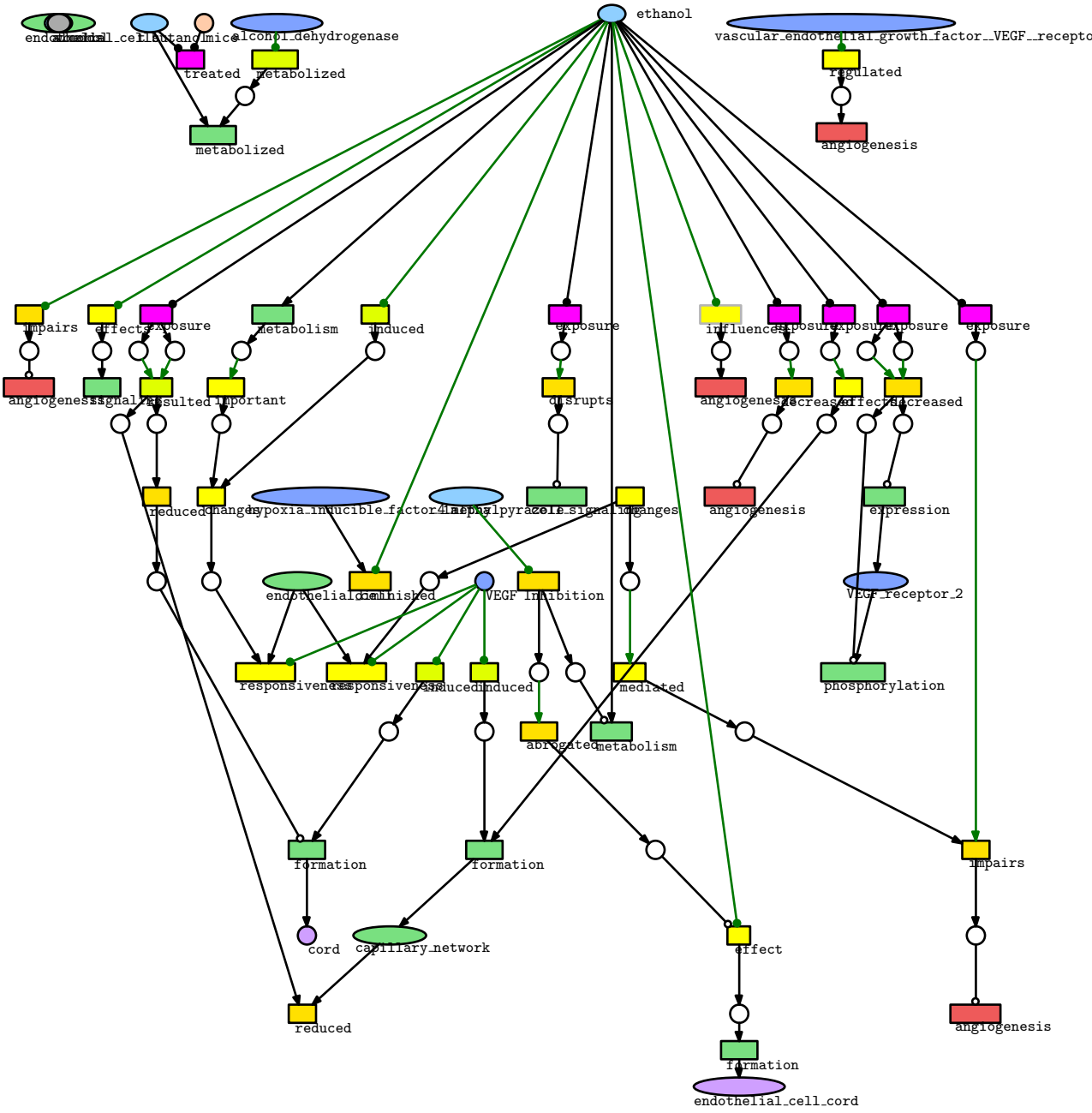


Fig. 14: **Generated Petri net.** *Acute ethanol exposure disrupts VEGF receptor cell signaling in endothelial cells.* [PMID-18469146]

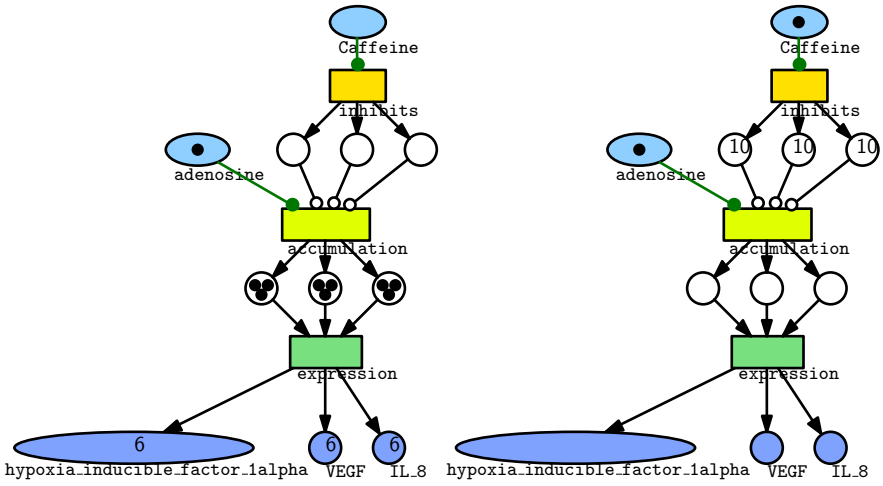


Fig. 15: **Animated Petri net** (a) Initial marking: 1 token in `adenosine`. Step count: 10 (b) Initial marking: 1 token in `Caffeine` and 1 token in `adenosine`. Step count: 10

5.4 Time and effort

The generation of Petri nets can be divided into 3 stages; annotation, conversion and Petri net analysis. When using unannotated documents, the first stage will require some time and effort. However, the BRAT interface is very intuitive. Annotations are added by selecting text with the cursor and arguments are added to events by dragging an arrow from the event to the entity. Entity and event types can be quickly selected by using keyboard shortcuts. The annotation of an abstract costs on average half an hour to an hour maximum. This event-level approach of studying text also requires minimal understanding of biology, although assigning the right entity types and events may be a small obstacle for non-biologists.

As for the next steps, minimal time and effort is required. Obtaining the annotation file, converting it in Woodstock and changing the layout in Snoopy only takes seconds. Analysing and adapting Petri nets in Snoopy is also very intuitive and should not require much effort.

6 Conclusions

In this thesis possibilities for assistance in the construction of literature-based biological Petri net models were investigated. Different approaches using literature annotation were proposed, a scheme for the annotation of biological events across multiple levels of biological organization was presented and Woodstock, an online tool for the conversion of annotated biological events to extended Petri net models, was developed.

Manual annotation using the MLEE annotation scheme proved to be a relatively easy and fast approach to extract information from literature. Experiments using Woodstock with pre-annotated documents from the MLEE corpus demonstrated that Petri net models can be obtained, looking similar to manually constructed Petri nets (Fig. 13, 14) and, when animated in Snoopy, can exhibit behaviour corresponding to the description in the literature (Fig. 15). However, since relevant information in the MLEE corpus was not annotated in some cases, it is expected that manual document annotation will result in better Petri net models. To conclude, the work in this thesis can be viewed as a first step in the construction of Petri net models based on annotated literature.

Earlier, an approach was developed to generate a Petri net model for the plant defence response to pathogens on the molecular level based on literature [17]. An annotation scheme was used consisting of 2 entity types, *Subject* and *Object*, and one event type, consisting of an event trigger, *Predicate*, with exactly one Subject and one Object as its arguments. This technique was called a *triplet extraction*. Afterwards, they compared the generated Petri net with a manually developed Petri net and found that it could not compete –in terms of detail and correctness– with the manually constructed Petri net, but that it could be used to assist experts in building a model or provide novel information. Although we did not compare generated Petri nets with manually constructed Petri nets, we can also state that Petri nets generated by Woodstock are unable to compete with manual constructed Petri nets. However, using *event extraction*, we were able to extract more complex information and thus generate Petri nets with more detail.

In this thesis we focussed on the correct conversion of individual types of actions. However, a single change in a cell, for example the synthesis of certain cellular product, is often the result of a long series of actions among various molecules within or outside the cell. Those series of actions are called *biological pathways* [9] and a logical next step would be to focus on complete pathways in future work. The most common biological pathways can be divided into three groups. The first group are the metabolic pathways, which make the chemical reactions possible, such as the building of molecules or the breakdown of food into energy molecules that can be stored for later use. The second group, gene-regulatory pathways, turn genes on and off. By doing so they influence which proteins are produced in the cell, thus determining the exact appearance, function and behaviour of the cell. Last, signal transduction pathways, move a signal

from a cell's exterior to its interior. Cells are able to receive specific signals through protein structures on their surface (*receptors*). After interacting with these receptors, the signal travels into the cell, where its message is transmitted by specialized proteins that trigger a specific reaction in the cell (e.g. production of a protein or spur the cell to move).

Figure 16 illustrates a typical example of a pathway description, explaining a part of the heat shock response pathway [7]. It describes the regulation of heat shock proteins (HSPs) by heat shock factors (HSFs) and vice versa [16, 20]. To provide a starting point for future work, we annotated this description and obtained the Petri net depicted in Fig. 17. It was generated by Woodstock and visualized by Snoopy using a Planarization layout. See Appendix 7.4 for the animated Petri net.

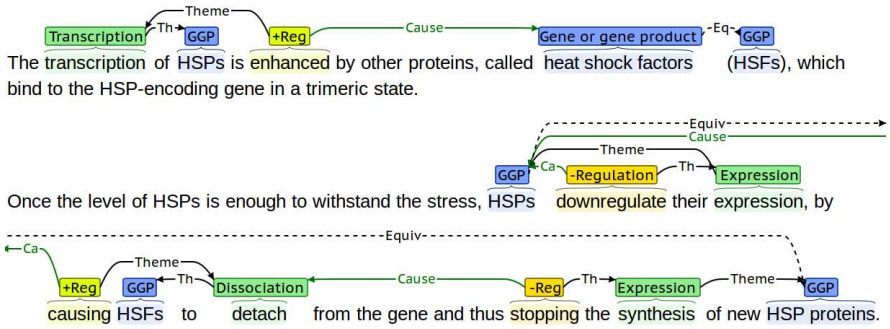


Fig. 16: Annotated description of heat shock response

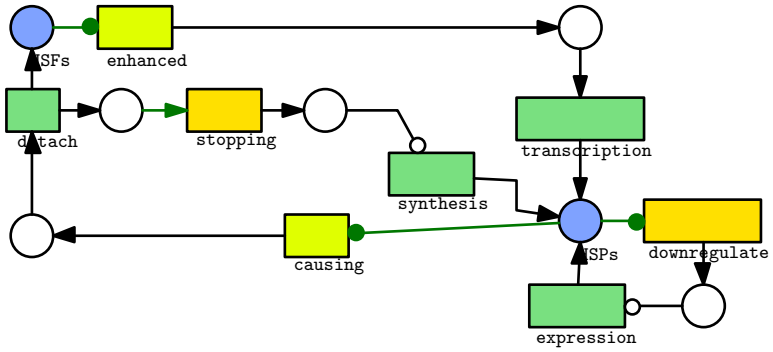


Fig. 17: Petri net for heat-shock response

In the future it could also be interesting to experiment with different rules for the conversion of annotations to Petri nets or to add rules to improve the Petri net. For example, by adding one extra rule it is possible to merge multiple auxiliary places between a pair of transitions to one auxiliary place.

Furthermore, it may be useful to add conversion rules for the *Fragment* relation type and for those arguments for which currently no conversion rules exist. (Table 1). Another option would be to choose a completely different annotation scheme for the extraction of information. For example, the GENIA annotation scheme enriched with meta-knowledge will enable the creation of more specific conversion rules –in particular for the conversion of regulatory events–, although annotation will be more time consuming and the development of a conversion tool may be more challenging. Finally, in addition to the generation of extended Petri nets it would also be interesting to generate different types of Petri nets (e.g. hierarchical Petri nets or quantitative Petri nets).

Acknowledgements

First, and most of all, I would like to thank my supervisor Jetty Kleijn for providing me with useful new insights and for her continuous support during the project. I would also like to thank Fons Verbeek for being the second examiner. Last of all I would like to thank my sister and my friends who supported me throughout the process of creating this thesis.

References

1. Mary Ann Blätke, Monika Heiner, and Wolfgang Marwan. Tutorial: Petri nets in systems biology. 2011.
2. David Campos, Jóni Lourenço, Sérgio Matos, and José Luís Oliveira. Egas: a collaborative and interactive document curation platform. *Database*, 2014:bau048, 2014.
3. Ming Chen, Sridhar Hariharaputran, Ralf Hofestädt, Benjamin Kormeier, and Sarah Spangardt. Petri net models for the semi-automatic construction of large scale biological networks. *Natural Computing*, 10(3):1077–1097, 2011.
4. Jörg Desel and Gabriel Juhás. What is a Petri net: Informal answers for the informed reader. In *Unifying Petri Nets*, pages 1–25. Springer, 2001.
5. Jasmin Fisher, David Harel, and Thomas A Henzinger. Biology as reactivity. *Communications of the ACM*, 54(10):72–82, 2011.
6. Jasmin Fisher and Thomas A Henzinger. Executable cell biology. *Nature biotechnology*, 25(11):1239–1249, 2007.
7. Diana-Elena Gratie. PhD thesis in preperation. 2016.
8. Monika Heiner, Mostafa Herajy, Fei Liu, Christian Rohr, and Martin Schwarick. Snoopy–a unifying petri net tool. In *International Conference on Application and Theory of Petri Nets and Concurrency*, pages 398–407. Springer, 2012.
9. National Human Genome Research Institute. Biological pathways. `https://www.genome.gov/27530687`.
10. National Human Genome Research Institute. BRAT - MLEE. `http://www.nactem.ac.uk/eccb2012/index.xhtml#/`.
11. Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.
12. Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. Association for Computational Linguistics, 2009.
13. Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. Genia corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182, 2003.
14. Martin Krallinger, Florian Leitner, Carlos Rodriguez-Penagos, and Alfonso Valencia. Overview of the protein-protein interaction annotation extraction task of biocreative ii. *Genome biology*, 9(2):1, 2008.
15. Rasmus Lerdorf. Php: Hypertext preprocessor. *URL http://php. net*, 1995.
16. Susan Lindquist. The heat-shock response. *Annual review of biochemistry*, 55(1):1151–1191, 1986.
17. Dragana Miljkovic, Vid Podpečan, Miha Grčar, Kristina Gruden, Tjaša Stare, Marko Petek, Igor Mozetič, and Nada Lavrač. Modelling a biological system: network creation by triplet extraction from biological literature. In *Bisociative Knowledge Discovery*, pages 427–437. Springer, 2012.
18. Stephen C North and E Koutsofios. Drawing graphs with dot. *TechnicalReport, AT&T Bell Lab*, 1993.
19. Tomoko Ohta, Sampo Pyysalo, Jun'ichi Tsujii, and Sophia Ananiadou. Open-domain anatomical entity mention detection. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, pages 27–36. Association for Computational Linguistics, 2012.

20. Ion Petre, Andrzej Mizera, Claire L Hyder, Annika Meinander, Andrey Mikhailov, Richard I Morimoto, Lea Sistonen, John E Eriksson, and Ralph-Johan Back. A simple mass-action model for the eukaryotic heat shock response and its mathematical validation. *Natural Computing*, 10(1):595–612, 2011.
21. Corrado Priami. Algorithmic systems biology. *Communications of the ACM*, 52(5):80–88, 2009.
22. Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, Han-Cheol Cho, Jun'ichi Tsujii, and Sophia Ananiadou. Event extraction across multiple levels of biological organization. *Bioinformatics*, 28(18):i575–i581, 2012.
23. Matthew S Simpson and Dina Demner-Fushman. Biomedical text mining: a survey of recent progress. In *Mining text data*, pages 465–517. Springer, 2012.
24. Harald Stachelscheid, Stefanie Seltmann, Fritz Lekschas, Jean-Fred Fontaine, Nancy Mah, Mariana Neves, Miguel A Andrade-Navarro, Ulf Leser, and Andreas Kurtz. Cellfinder: a cell data repository. *Nucleic acids research*, 42(D1):D950–D958, 2014.
25. Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.
26. Kozo Sugiyama. A cognitive approach for graph drawing. *Cybernetics and Systems: An International Journal*, 18(6):447–488, 1987.
27. Paul Thompson, Raheel Nawaz, John McNaught, and Sophia Ananiadou. Enriching a biomedical event corpus with meta-knowledge annotation. *BMC bioinformatics*, 12(1):1, 2011.

7 Appendices

7.1 A BRAT annotation configuration file

BRAT file for the configuration of annotation schemes

annotation.conf

```
[entities]

!Organisms
    Organism
----------
!Anatomy
    Organism_subdivision
    Anatomical_system
    Organ
    Multi-tissue_structure
    Tissue
    Cell
    Cellular_component
    Pathological_formation
    Organism_substance
    Immaterial_anatomical_entity
    Developing_anatomical_structure
----------
!Molecular
    Gene_or_gene_product
    Protein_domain_or_region
    DNA_domain_or_region
    Drug_or_compound
----------

[relations]

Equiv Arg1:Descriptive_reference, Arg2:Descriptive_reference
Equiv Arg1:Protein_domain_or_region, Arg2:
    Protein_domain_or_region
Equiv Arg1:DNA_domain_or_region, Arg2:DNA_domain_or_region
Equiv Arg1:Gene_or_gene_product, Arg2:Gene_or_gene_product
Equiv Arg1:Protein_family_or_group, Arg2:
    Protein_family_or_group
Equiv Arg1:Cell_type, Arg2:Cell_type
Equiv Arg1:Drug_or_compound, Arg2:Drug_or_compound
Equiv Arg1:Other_pharmaceutical_agent, Arg2:
    Other_pharmaceutical_agent
Equiv Arg1:Tissue, Arg2:Tissue
Equiv Arg1:Organism, Arg2:Organism
Equiv Arg1:Other_entity, Arg2:Other_entity
Equiv Arg1:Cell, Arg2:Cell
Equiv Arg1:Cellular_component, Arg2:Cellular_component
```

```
41 Equiv Arg1:Pathological_formation, Arg2:Pathological_formation
42 Equiv Arg1:Organism_substance, Arg2:Organism_substance
43 Equiv Arg1:Anatomical_system, Arg2:Anatomical_system
44 Equiv Arg1:Multi-tissue_structure, Arg2:Multi-tissue_structure
45
46 frag Arg1:Descriptive_reference, Arg2:Descriptive_reference
47 frag Arg1:Gene_or_gene_product, Arg2:Gene_or_gene_product
48 frag Arg1:Protein_domain_or_region, Arg2:
       Protein_domain_or_region
49 frag Arg1:Tissue, Arg2:Tissue
50 frag Arg1:Cell, Arg2:Cell
51 frag Arg1:Organism, Arg2:Organism
52 frag Arg1:Cell_type, Arg2:Cell_type
53 frag Arg1:Other_pharmaceutical_agent, Arg2:
       Other_pharmaceutical_agent
54 frag Arg1:Drug_or_compound, Arg2:Drug_or_compound
55 frag Arg1:Pathological_formation, Arg2:Pathological_formation
56 frag Arg1:Multi-tissue_structure, Arg2:Multi-tissue_structure
57 frag Arg1:Organism_subdivision, Arg2:Organism_subdivision
58 frag Arg1:Cellular_component, Arg2:Cellular_component
59
60 ENTITY-NESTING Arg1:Gene_or_gene_product, Arg2:Organism
61 ENTITY-NESTING Arg1:Gene_or_gene_product, Arg2:Tissue
62 ENTITY-NESTING Arg1:Gene_or_gene_product, Arg2:Cell
63 ENTITY-NESTING Arg1:Gene_or_gene_product, Arg2:
       Pathological_formation
64 ENTITY-NESTING Arg1:Gene_or_gene_product, Arg2:Cell_type
65 ENTITY-NESTING Arg1:<ANY>, Arg2:Descriptive_reference
66 ENTITY-NESTING Arg1:Other_pharmaceutical_agent, Arg2:<ANY>
67 ENTITY-NESTING Arg1:Other_entity, Arg2:<ANY>
68 ENTITY-NESTING Arg1:<ANY>, Arg2:Other_entity
69
70 [events]
71
72 <CORE-ENTITY>=Gene_or_gene_product|Drug_or_compound
73 <CELLS>=Cell_type|Tissue
74 <ANATOMY>=Organism|Organism_subdivision|Anatomical_system|
       Organ|Multi-tissue_structure|Tissue|Cell|
       Cellular_component|Pathological_formation|
       Organism_substance|Immaterial_anatomical_entity|
       Developing_anatomical_structure
75
76 !Anatomical
77   Development Theme:Cell_type|Tissue|Organism|<ANATOMY>
78     Blood_vessel_development Theme?:Cell_type|Tissue|Organism
         |<ANATOMY>, AtLoc?:<CELLS>|<ANATOMY>, ToLoc?:<CELLS>|<
         ANATOMY>, FromLoc?:<CELLS>|<ANATOMY>
79   Growth Theme:Cell_type|Tissue|Organism|<ANATOMY>
80   Death Theme:Cell_type|Tissue|Organism|<ANATOMY>
81   Breakdown Theme:Tissue|<ANATOMY>
```

```
82    Cell_proliferation  Theme: Cell_type | Cell
83    Cell_division  Theme: Cell_type | Cell
84    Remodeling  Theme: Tissue|<ANATOMY>
85    Reproduction  Theme: Organism
86  ----------
87  ! Molecular
88    Metabolism  Theme: Gene_or_gene_product | Drug_or_compound
89      Synthesis  Theme: Drug_or_compound
90      Catabolism  Theme: Gene_or_gene_product | Drug_or_compound
91      Gene_expression  Theme+: Gene_or_gene_product |
            Descriptive_reference
92        Transcription  Theme: Gene_or_gene_product
93        Translation  Theme: Gene_or_gene_product
94        Protein_processing  Theme: Gene_or_gene_product
95    Phosphorylation  Theme: Gene_or_gene_product | Drug_or_compound ,
          Site?: Protein_domain_or_region
96    Dephosphorylation  Theme: Gene_or_gene_product |
        Drug_or_compound , Site?: Protein_domain_or_region
97    Ubiquitination  Theme: Gene_or_gene_product | Drug_or_compound ,
          Site?: Protein_domain_or_region
98    Deubiquitination  Theme: Gene_or_gene_product |
        Drug_or_compound , Site?: Protein_domain_or_region
99    Acetylation  Theme: Gene_or_gene_product | Drug_or_compound ,
        Site?: Protein_domain_or_region
100   Deacetylation  Theme: Gene_or_gene_product | Drug_or_compound ,
        Site?: Protein_domain_or_region
101   DNA_methylation  Theme: Gene_or_gene_product | Drug_or_compound ,
          Site?: DNA_domain_or_region
102   DNA_demethylation  Theme: Gene_or_gene_product |
        Drug_or_compound , Site?: DNA_domain_or_region
103   Pathway  Participant*:<ENTITY>|<EVENT>
104 ----------
105 ! General
106   Localization  Theme+:<CORE-ENTITY>|<CELLS>|<ANATOMY>|
        Descriptive_reference , ToLoc?:<CELLS>|<ANATOMY>, AtLoc
        ?:<CELLS>|<ANATOMY>, FromLoc?:<CELLS>|<ANATOMY>
107   Binding  Theme+:<CORE-ENTITY>|<CELLS>|<ANATOMY>|
        Descriptive_reference , Site?: Protein_domain_or_region |
        DNA_domain_or_region
108   Dissociation  Theme+:<CORE-ENTITY>|<CELLS>|<ANATOMY>
109   Regulation  Theme:<ENTITY>|<EVENT>, Cause?:<ENTITY>|<EVENT>,
          Site?: Protein_domain_or_region | DNA_domain_or_region ,
        CSite?: Protein_domain_or_region | DNA_domain_or_region
110     Positive_regulation  Theme:<ENTITY>|<EVENT>, Cause?:<ENTITY
          >|<EVENT>, Site?: Protein_domain_or_region |
          DNA_domain_or_region , CSite?: Protein_domain_or_region |
          DNA_domain_or_region
111     Negative_regulation  Theme:<ENTITY>|<EVENT>, Cause?:<ENTITY
          >|<EVENT>, Site?: Protein_domain_or_region |
```

```
        DNA_domain_or_region, CSite?:Protein_domain_or_region|
        DNA_domain_or_region
112 ----------
113 !Planned
114   Planned_process Theme*:<ENTITY>|Blood_vessel_development,
          Instrument*:<ENTITY>
115 ----------
116 !Other
117   Other_event Pre*:<ENTITY>|<EVENT>, Post*:<ENTITY>|<EVENT>
118
119 [attributes]
120
121 Negation  Arg:<EVENT>
122 Speculation Arg:<EVENT>
```

7.2 B Woodstock source code

The complete source code of Woodstock (version 1.0) can be downloaded from `http://www.liacs.nl/~alouwe`.

7.3 C Snoopy Extended Petri Net file (spept)

Example of a Snoopy Extended Petri Net (spept) file generated by Woodstock.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="/xsl/spped2svg.xsl"?>
<Snoopy version="2" revision="1.13">
  <netclass name="Extended Petri Net"/>
    <nodeclasses count="4">
      <nodeclass count="0" name="Place">
        <node id="2" net="1">
          <attribute name="Name" net="1">
            <![CDATA[Cats]]>
            <graphics count="1">
              <graphic grparent="-2" xoff="0" yoff="15"  net="
                  1" show="1" state="1" />
            </graphics>
          </attribute>
          <attribute name="ID" net="1">
            <![CDATA[2]]>
            <graphics count="1">
              <graphic grparent="-2" net="1" show="0" state="1
                  " pen="0,0,0" brush="255,255,255"/>
            </graphics>
          </attribute>
          <graphics count="1">
            <graphic id="-2" x="20" y="20" net="1" show="1" w=
                "20" h="20" state="1" pen="0,0,0" brush="255,
                204, 170"/>
          </graphics>
        </node>
        <node id="4" net="1">
          <attribute name="Name" net="1">
            <![CDATA[mice]]>
            <graphics count="1">
              <graphic grparent="-4" xoff="0" yoff="15"  net="
                  1" show="1" state="1" />
            </graphics>
          </attribute>
          <attribute name="ID" net="1">
            <![CDATA[4]]>
            <graphics count="1">
              <graphic grparent="-4" net="1" show="0" state="1
                  " pen="0,0,0" brush="255,255,255"/>
            </graphics>
```

```
36            </attribute>
37            <graphics count="1">
38              <graphic id="-4" x="20" y="20" net="1" show="1" w=
                    "20" h="20" state="1" pen="0,0,0" brush="255,
                    204, 170"/>
39            </graphics>
40          </node>
41        </nodeclass>
42        <nodeclass count="0" name="Transition">
43          <node id="6" net="1">
44            <attribute name="Name" net="1">
45              <![CDATA[eat]]>
46              <graphics count="1">
47                <graphic grparent="-6" xoff="0" yoff="15"  net="
                      1" show="1" state="1" />
48              </graphics>
49            </attribute>
50            <attribute name="ID" net="1">
51              <![CDATA[6]]>
52              <graphics count="1">
53                <graphic grparent="-6" net="1" show="0" state="1
                      " pen="0,0,0" brush="255,255,255"/>
54              </graphics>
55            </attribute>
56            <graphics count="1">
57              <graphic id="-6" x="20" y="20" net="1" show="1" w=
                    "20" h="20" state="1" pen="0,0,0" brush="
                    255,224,0"/>
58            </graphics>
59          </node>
60        </nodeclass>
61      </nodeclasses>
62      <edgeclasses count="5">
63      <edgeclass count="1" name="Read Edge">
64        <edge source="2" target="6" id="1" net="1">
65          <graphics count="1">
66            <graphic net="1" source="-2" target="-6" id="-1"
                  state="1" show="1" pen="0,119,0" brush="0,119,0"
                   edge_designtype="3">
67              <points count="2">
68                <point x="20" y="20"/>
69                <point x="20" y="20"/>
70              </points>
71            </graphic>
72          </graphics>
73        </edge>
74      </edgeclass>
75      <edgeclass count="1" name="Edge">
76        <edge source="4" target="6" id="2" net="1">
77          <graphics count="1">
```

```
78            <graphic net="1" source="-4" target="-6" id="-2"
                  state="1" show="1" pen="0,0,0" brush="0,0,0"
                  edge_designtype="3">
79              <points count="2">
80                <point x="20" y="20"/>
81                <point x="20" y="20"/>
82              </points>
83            </graphic>
84          </graphics>
85        </edge>
86      </edgeclass>
87    </edgeclasses>
88 </Snoopy>
```

7.4 D Animated Petri nets

Videos of animated Petri nets (in Snoopy) can be found at `http://www.liacs.nl/~alouwe`.