MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Robust Gene Ontology Enrichment Analysis of Partially-Specified Boolean Networks

Bachelor's Thesis

TIMOTEJ ŠVÁBY

Brno, Fall 2025

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Robust Gene Ontology Enrichment Analysis of Partially-Specified Boolean Networks

Bachelor's Thesis

TIMOTEJ ŠVÁBY

Advisor: doc. RNDr. David Šafránek, Ph.D.

Department of Machine Learning and Data Processing

Brno, Fall 2025


FACULTAS ARTIS INFORMATICÆ
UNIVERSITAS MASARYKIANA

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. During the preparation of this thesis, I used the AI tools, including Grammarly for grammar check, and ChatGPT to improve my writing style. I declare that I used these tools in accordance with the principles of academic integrity. I checked the content and took full responsibility for it. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Timotej Šváby

Advisor: doc. RNDr. David Šafránek, Ph.D.

Acknowledgements

I would like to sincerely thank my advisor, doc. RNDr. David Šafránek, Ph.D., for their guidance and valuable feedback throughout the writing of my thesis.

Abstract

Boolean networks are widely used for qualitative modelling of gene regulatory and signalling systems, but in many biological applications, the underlying regulatory logic is only partially known. This uncertainty can be represented using partially-specified Boolean networks (PSBNs), which encode sets of admissible network instances rather than a single fully defined model. This thesis presents a computational pipeline for attractor-based Gene Ontology enrichment analysis of PSBNs built on the AEON framework and its Python interface. The pipeline enumerates network instances, computes their attractors under asynchronous dynamics, performs GO enrichment on attractor-derived gene sets, and aggregates results across attractors and instances to identify robust functional patterns. The approach is evaluated on three established signalling pathways and produces results consistent with their known biological behaviour, demonstrating that functional interpretation of PSBN dynamics is feasible despite regulatory uncertainty.

Keywords

Gene ontology, GO enrichment analysis, GO term, Boolean network, partially-specified Boolean network, attractor, AEON, network instance

Contents

Introduction

Gene regulatory networks capture how genes affect each other's activity and, in turn, shape the behaviour of a cell. One way to model these interactions is through Boolean networks (BNs), where each gene is assigned an on/off state and regulatory influences are written as logical rules that update those states. Boolean models are beneficial in systems biology, when quantitative kinetic parameters are unavailable.

A central concept in BN dynamics is that of attractors — states reached over time. Attractors are commonly viewed as representing cellular phenotypes, such as particular differentiated cell types or pathological states. To connect these attractors with known biological functions, Gene Ontology (GO) enrichment analysis can be applied to determine which functional categories are overrepresented (more than expected by chance) among the genes that are active in a given attractor. This helps place the model's predicted behaviours into a biological context.

A fully specified BN requires complete knowledge of all regulatory logic. In practice, biological knowledge is incomplete, which leads to partially-specified Boolean networks (PSBNs), where some logical functions are ambiguous or have multiple plausible forms. A PSBN can be understood as encoding not just one model, but a whole collection of models that share the same network structure and set of possible states. To distinguish between models of network, we speak about instances. Studying attractors in this context means considering the results across multiple valid instantiations of the network.

The resulting pipeline is built upon and extends AEON, a software framework from the Faculty of Informatics of Masaryk University[1] for constructing, abstracting, and dynamically analyzing BNs. AEON provides efficient algorithms for attractor computation and model manipulation, and its aeon.py Python API enables seamless integration into automated workflows and scripting environments.

This thesis aims to develop a pipeline for GO enrichment analysis of attractors derived from PSBNs in AEON. The computational pipeline extends the AEON Python API to generate instances consistent with PSBN and to compute their attractors. Finally, postprocessing strate-

1. `https://sybila.fi.muni.cz/tools_html/aeon.html`

gies are introduced to summarize and interpret enrichment outcomes across the resulting ensemble of instances and in instances themselves. By allowing the network to be examined without requiring every regulatory detail to be fully defined, this work can discover biological signals that analysis restricted to fully specified models might overlook. In addition, it provides a practical approach to linking BN dynamics with large-scale functional annotation resources, even in situations where detailed mechanistic knowledge is incomplete.

1 Background

This chapter introduces the key theoretical foundations and tools relevant to this thesis. The first part focuses on enrichment analysis, including the general principles behind it and the use of GO terms to interpret gene sets. Then it describes BNs as a modelling framework for biological systems, also with a focus on PSBNs, and continues with the concept of attractors, which represent the long-term behaviour of such networks. Finally, the chapter introduces the AEON tool, which provides methods for analyzing long-term dynamics and is extended in the practical part of this thesis.

1.1 Enrichment analysis

Modern high-throughput experimental techniques, such as microarrays and RNA-seq, measure the expression levels of many genes or proteins simultaneously. Such experiments usually produce large sets of genes that show significant increases or decreases in expression under a given biological condition. Interpreting these results is challenging, however, because the biological relevance of single genes is often difficult to interpret when viewed on their own.
Enrichment analysis provides a way to interpret these gene lists by identifying whether specific biological functions, processes, or pathways are statistically overrepresented compared to what would be expected by chance. In other words, enrichment analysis highlights shared biological themes within a set of genes [1, 2]. There are several methodological approaches to enrichment analysis, each differing in how they use gene expression information and how they define gene sets:

- Over-Representation Analysis (ORA). ORA assumes starting with a set of genes judged to be relevant, for instance, those showing significant differential expression when applying a threshold such as adjusted p-value < 0.05. The main question is whether certain biological functions or pathways include these genes more frequently than would occur by chance. To evaluate enrichment, statistical tests such as the hypergeometric test or Fisher's exact test are typically used. Although this approach is

simple and fast to compute, its outcomes depend strongly on the significance threshold used to select genes, which means that more subtle biological signals can be missed [1, 2].

- Gene Set Enrichment Analysis (GSEA). In contrast to ORA, GSEA works with a ranked list of all measured genes, typically ranked by differential expression statistics. Instead of selecting only a subset of significant genes, GSEA tests whether genes from predefined pathways are more often found among the most up-regulated or down-regulated genes than would be expected by chance. This method can reveal small but coordinated changes in groups of genes and is often easier to interpret biologically. However, GSEA requires careful control of ranking metrics and multiple testing correction [2, 3].
- Pathway Topology-Based Enrichment. More recent enrichment strategies consider the structure of biological pathways, including interactions and regulatory relationships between molecules. In contrast to set-based approaches, these techniques make use of the underlying network structure of the pathway. By incorporating interaction direction and regulatory impact, they can highlight how signals travel through the system and reveal underlying control mechanisms. However, such methods depend heavily on the completeness and correctness of pathway databases, which remain uneven across biological systems [3].

In practice, most enrichment approaches follow a similar analysis workflow consisting of the following steps:

1. Preparing a gene list The input to enrichment analysis is typically a set of genes of interest, such as significantly up- or down-regulated genes from differential expression analysis, or a ranked gene list as used in GSEA. This gene list captures the biological response to the studied condition and serves as the basis for all downstream interpretation.
2. Selecting a reference background Enrichment results are obtained by comparing the selected genes with a reference set, most often all genes measurable in the experiment. An improperly defined background can bias the enrichment calculation, producing results that do not accurately reflect the underlying biology.
3. Testing for overrepresentation and correcting for multiple comparisons The statistical task is to evaluate whether certain biological

functions, pathways, or GO terms occur more frequently in the input gene list than would be expected by chance. This is commonly done using the hypergeometric test or Fisher's exact test in ORA-based workflows, or by calculating enrichment scores in GSEA-based approaches. Since enrichment analysis usually tests many functional categories, the resulting p-values need to be adjusted to control the false discovery rate (FDR), most often using the Benjamini–Hochberg procedure. After correction, terms may be identified as:
Over-represented: occurring more frequently than expected, suggesting activation of associated biological processes.
Under-represented: occurring less frequently than expected, which may indicate suppression or loss of function and is also biologically meaningful.

4. Results once significant functional categories have been identified, results are typically summarized and visualized for interpretation. Enrichment results are presented as term lists and visual summaries, for example, bar or dot plots. Network-style views may also be used to group functionally related terms. Tools like GOrilla allow enriched terms to be shown in the GO hierarchy, making functional relationships visible. Network-based visualization approaches, such as those provided by EnrichmentMap [4], further aid interpretation by grouping related enriched categories into biological themes.

1.2 Gene ontology

The GO was introduced to address the lack of consistency in gene function descriptions across biological databases. It provides a structured vocabulary that allows functional annotations of genes and proteins to be compared between species [5]. In the past, gene functions were annotated using many different conventions, depending on the database or organism. As a result, comparing functional information across species was often difficult. Gene Ontology was created to unify these descriptions through a common hierarchical system that does not depend on species-specific terminology. GO consists of three main branches, each capturing a different aspect of gene function:

- Biological Process (BP) - representing pathways and larger processes performed by multiple molecular activities, for instance, cell division or signal transduction
- Molecular Function (MF) - describing specific activities carried out by individual gene products, for example, ATP binding or kinase activity
- Cellular Component (CC) - symbolizing where in the cell a gene product carries out its function, for instance, within the nucleus, the cytoplasm, or along the mitochondrial membrane

Each entry is assigned a distinct identifier and linked to related terms through hierarchical relations such as is a or part of, creating a structured network known as a directed acyclic graph. This structure allows both general and specific annotations to coexist and support reasoning across different levels of biological abstraction [6].
Since its introduction, the GO has developed into an extensive and continuously curated knowledge base. It is maintained by the GO Consortium, which integrates newly published experimental results and updates annotation quality on an ongoing basis. The most recent description of the resource reports millions of annotations spanning thousands of species, produced through a combination of automated pipelines and expert review [6]. This active curation keeps GO a dependable foundation for large-scale functional studies and omics analyses. Many analytical resources have incorporated GO to support the interpretation of gene sets. The PANTHER system combines GO terms with phylogenetic and pathway information, allowing both enrichment testing and ontology-based visualization of molecular pathways [7, 8]. Other widely used environments, such as DAVID, Enrichr, and g:Profiler, provide user-friendly web interfaces for annotation and enrichment analysis, linking GO terms with a variety of complementary biological databases [9]. Despite its wide use, GO-based analyses have limitations. The accuracy of enrichment results depends on the completeness and quality of the underlying annotations. These can vary widely between organisms and experimental settings. In addition, the hierarchical structure of GO causes related terms to overlap or describe similar biological processes. This redundancy can make results harder to interpret [1, 3]. Meaningful conclusions therefore require careful consideration of both the ontology structure and the evidence supporting individual annotations. Even so, GO remains

a central resource in bioinformatics. Its continued development and integration with tools such as PANTHER and DAVID ensure its importance for the analysis of large-scale biological and genomic data [6, 7, 9].

1.3 Boolean networks

BNs provide a modeling framework for studying the behaviour of complex biological systems, especially gene regulatory networks. In a BN, genes or proteins are modeled in a simplified way, where each component can only be active or inactive. These states are updated using logical rules that depend on the activity of other components in the network. Such models are useful when detailed quantitative information is missing, for example precise reaction rates or binding affinities.
In this thesis, Boolean networks are used according to the definition given in [10].
A BN $F^{(n)}$ assumes n Boolean variables $\mathrm{Var}_n = \{v_1, \ldots, v_n\}$ and consists of n Boolean update functions $\{F_1, \ldots, F_n\}$ (one for each variable), such that $F_i : \mathbb{B}^n \to \mathbb{B}$. Each Boolean-valued vector of $\mathbb{B}^n$ represents an assignment of Boolean values to the variables of Var_n. Let us call the set $\mathbb{B}^n$ the state space of $F^{(n)}$, and the members of $\mathbb{B}^n$ its states.

For each of the functions F_i,

$$v_j \in \mathrm{dep}(F_i) \iff \exists x \in \mathbb{B}^n.\ F_i\big(x[j \mapsto 1]\big) \neq F_i\big(x[j \mapsto 0]\big).$$

In this thesis, the asynchronous updating scheme is used. Under this scheme, at each update step, a single variable is updated independently of the others, as opposed to synchronous updating, where all variables update at once. This leads to a directed state-transition graph $\mathrm{STG}(F^{(n)}) = (S, T)$, where $S = \mathbb{B}^n$ is the network's state space, and the transition relation $T \subseteq S \times S$ is defined as follows:

$$(s, t) \in T \iff s \neq t \ \wedge\ \exists i \in \{1, \ldots, n\}.\ t = s[i \mapsto F_i(s)].$$

Notice that in the resulting graph, every transition "updates" exactly one network variable. For technical reasons, without loss of generality, a self-loop is included $(s, s) \in T$ whenever $\forall i.\ s_i = F_i(s)$. The standard abbreviations $s \to t$, $s \to^+ t$ and $s \to^* t$ are used to denote that

$(s,t) \in T$, $(s,t) \in T^+$ (transitive closure) and $(s,t) \in T^*$ (reflexive and transitive closure), respectively. A run is an infinite sequence of states $s_1, s_2, \ldots$, such that $\forall i.\ (s_i, s_{i+1}) \in T$ [10].
BNs are commonly applied to problems such as cell differentiation, cell-cycle control, disease-related regulation, and signal transduction. They are also used to study feedback mechanisms and decision-making processes in regulatory systems. Because they rely on qualitative rules rather than precise measurements, BNs offer a practical compromise between biological intuition and formal computational modelling.

Partially-specified Boolean networks

While classical BNs assume that the logical update function for each node is fully known, biological knowledge is often incomplete or uncertain. Many regulatory interactions remain undiscovered, and experimental data may be ambiguous or conflicting. PSBNs address this limitation by allowing incomplete or partially defined update rules, thereby representing all of the possible models rather than a single fully determined network [10]. A PSBN does not assign a complete Boolean function to each variable. Instead, only constraints on the possible function are specified, such as known activators, inhibitors, or logical relationships that must or must not appear. Each node (variable) is associated with a partial Boolean function defined only for some regulator input combinations. The set of instances of a PSBN consists of all fully specified BNs that satisfy the given constraints. This allows researchers to reason about guaranteed behaviours that occur in all instances and possible behaviours that occur in at least one instance. This framework represents biological uncertainty more faithfully than classical BNs and allows for incremental refinement as additional knowledge becomes available.
I use the definition presented in [10].
Prior knowledge about the model's behaviour can often be formulated in the form of Partially Specified update functions. When defining such functions, we may use uninterpreted (fixed but otherwise unknown) function symbols alongside standard Boolean expressions. Formally, let us assume a set $\mathcal{S}$ of function symbols, each with a specified arity. In the following, we use boldface symbols such as $\mathbf{f}^{(a)}$ to

denote the members of this set (with the corresponding arity a).
A PSBN $E^{(n)}$ again assumes n variables Var_n and consists of n Partially Specified Boolean expressions $\{E_1, \ldots, E_n\}$, where each E_i is defined by the following grammar:

$$E ::= 0 \mid 1 \mid v \mid \neg E \mid E \wedge E \mid E \vee E \mid \mathbf{f}^{(a)}(E, \ldots, E).$$

Here, v ranges over Var_n. In practical applications, we also allow standard syntactic abbreviations for other operators like $\vee$ (disjunction), $\Rightarrow$ (implication), $\Leftrightarrow$ (equivalence), and $\oplus$ (exclusive or, non-equivalence). Finally, when the arity of f is 0, this symbol is effectively an unknown constant. As such, we can simply write f instead of $\mathbf{f}^{(0)}()$.
An interpretation of $\mathcal{S}$ is a function $\mathcal{I}$ that assigns to each symbol $\mathbf{f}^{(a)} \in \mathcal{S}$ a Boolean function $f^{(a)}$ with the same arity. By fixing a particular interpretation $\mathcal{I}$ for a given PSBN $E^{(n)}$, we substitute a concrete Boolean function for each function symbol in its Partially Specified Boolean expressions, and thus obtain a standard BN, which we denote by $E(\mathcal{I})$. Finally, we call a BN $F^{(n)}$ consistent with a PSBN $E^{(n)}$ if there exists an interpretation $\mathcal{I}$ of $\mathcal{S}$ such that $F = E(\mathcal{I})$ [10].
Note that two different interpretations can lead to the same STG. Hence, the number of BN instances that produce a distinct behaviour can be smaller than the number of interpretations. For the sake of simplicity, we use the notion "instance" interchangeably with the notion "interpretation". For enrichment analysis, the formal difference between the notions does not critically affect the results.
The framework of PSBNs represents biological uncertainty more faithfully than classical BNs and allows for incremental refinement as additional knowledge becomes available, while enabling rigorous analysis of long-term behaviours and regulatory mechanisms.

1.4 Attractors

The dynamics of a BN can be understood by studying how its states evolve over time on the state-transition graph. When transitions are applied repeatedly, the system does not wander arbitrarily through the state space forever. Instead, trajectories eventually enter regions from which they cannot escape. These regions are called attractors

and represent the possible long-term behaviours of the network.
The following definition is taken from the concrete article referenced in source [10].
Formally, an attractor A is a bottom (or terminal) strongly connected component of STG(F). In other words, for any two $s, t \in A$, $s \to^* t$ (that is, A is an SCC), and for all $s \in A$, $s \to t$ implies that $t \in A$ (i.e., A cannot be escaped). In the asynchronous STG(F), we generally talk about fixed-point attractors (where A is a singleton set), cyclic or oscillating attractors (where A is a cycle within STG(F)), and complex or disordered attractors (where A is any other set) [10].

- Stable (fixed point) attractor is an attractor made up of a single state that does not change when the update rules are applied. After the system reaches this state, it stays there permanently. Such attractors are commonly interpreted as stable cellular conditions, for example, a fully differentiated cell type or a robust regulatory configuration.
- Cyclic attractor contains multiple states arranged in one or more directed cycles. The system cycles continuously among these states, resulting in oscillatory or periodic behaviour, for example gene expression rhythms or cell cycle phases. With asynchronous updating, several internal transitions within the cycle are possible, but all states in the cycle remain mutually reachable.
- A Disordered or complex attractor may consist of numerous states with a rich web of possible transitions between them. The model can then move through these states in many different ways, producing dynamics that are neither fixed nor clearly periodic. In biological terms, this often corresponds to variable or unstable conditions, for instance, fluctuating regulatory programs or signal noise.

Tools for analysing BNs generally focus heavily on these attractors. AEON, as one example, includes dedicated methods to track how attractors form, disappear, or reorganise in response to parameter changes, enabling structured attractor-bifurcation analyses [11]. For models that are only Partially Specified, analysing attractors makes it possible to determine which long-term behaviours occur across every valid instance and which ones appear only in certain instances of the model [10]. A fully specified BN can have only one instance, while a PSBN can have many.

1.5 AEON

AEON is capable of identifying all attractors in every instance (in AEON called behavioural class). Aeon can also distinguish between these attractors, and examine how they change under variation in instances. This is essential when studying regulatory networks with various attractors. AEON performs attractor bifurcation analysis, determining how the attractor structure changes with parameter assignments [11]. AEON avoids explicit enumeration of the entire state space, which for a network of n nodes has size 2^n, by using symbolic data structures such as decision diagrams. This symbolic representation enables AEON to analyze networks that are too large to be handled by classical state enumeration methods.
The framework is accompanied by AEON.py. AEON.py supports the construction and loading of Boolean networks, symbolic attractor computation, exploration of parametrised model instances, and the integration of attractor analysis into larger computational workflows [12].
In this thesis, AEON is used as the primary tool for analysing attractors in Boolean networks, including models with partially specified logic. Its support for asynchronous dynamics, parameterised network models, and symbolic attractor computation enables the study of stability, oscillations, and complex dynamical behaviour in the presence of regulatory uncertainty. The contributions of this work build upon AEON's existing capabilities and extend its applicability to new forms of analysis.

2 Problem formulation

BN models are widely used to describe regulatory and signaling processes in biological systems. However, for many systems of interest, such as metabolic or signaling pathways, the precise regulatory logic is only partially known. In many cases, biological knowledge does not define a single BN. Instead, it specifies a set of possible network instances that share the same interaction topology but differ in their logical update rules. AEON captures this situation using PSBNs. Although this representation reflects biological uncertainty more realistically, it creates a major computational burden. A single PSBN can encode a very large number of fully specified networks, and this number can grow double-exponentially with the amount of missing logical information. This combinatorial growth has a direct impact on any subsequent dynamical analysis. Attractors, which correspond to long-term stable behaviours of the system, must be examined not just for a single instance, but across many possible network instances. This leads to an explosion in the number of attractors that could represent biologically meaningful states. Without further organization or interpretation, the results rapidly become unmanageable.
Understanding what the attractors represent is not immediate. Although each attractor reflects a stable activation pattern, such patterns gain meaning when they are interpreted through GO annotations. Existing methods for processing enrichment analysis typically consider fully specified BNs, and they do not summarize functional patterns that persist across structurally uncertain networks. Thus, the core problem addressed in this thesis is twofold:

- To extend the methodology of attractor enrichment analysis from plain, fully BNs to PSBNs
- To design the implementation of the extended methodology

The aim is to develop a pipeline capable of navigating the large solution space induced by partial specification, extracting recurrent functional themes, and highlighting emergent biological behaviours that would remain hidden if only fully specified, and thus artificially constrained, models were considered.

3 Pipeline design

This chapter describes the conceptual design of the computational pipeline developed in this thesis.

3.1 Pipeline overview

The pipeline connects the modelling capabilities of PSBNs in AEON with the functional interpretation provided by GO enrichment analysis. Unlike enrichment workflows applied to a single instance, the pipeline must operate on a family of network instances that arise from Partially Specified regulatory logic. As a result, the design must account for model enumeration, attractor computation, aggregation of attractor states, and postprocessing of enrichment results so that meaningful biological conclusions can be drawn. At a conceptual level, the pipeline follows three main stages:

- Model interpretation stage, in which the PSBN is interpreted as a space of fully specified BN instances.
- Dynamical analysis stage, in which long-term behaviours of each instance are computed in the form of attractors.
- Functional interpretation stage, in which attractor activity patterns are mapped to biological functions using GO enrichment and subsequently aggregated across instances and within each instance.

The overall objective of this design is not only to compute attractors for individual instances, but to identify stable functional patterns that persist across many valid network instances, as well as to highlight behaviours that potentially depend or are hidden in (compared to what is visible within only particular instances) unresolved regulatory structure.

3.2 Requirements and constraints

Functional Requirements

The pipeline must satisfy the following functional requirements:

- PSBN Input Parsing - load a PSBN defined using uninterpreted function symbols and Boolean expressions.
- Model Enumeration - generate all instances of a PSBN by interpreting function symbols as concrete Boolean functions.
- Attractor Computation - for each network instance, use AEON.py to compute all attractors in the asynchronous STG.
- Attractor-to-Gene-Set translation - convert attractors into biologically interpretable sets of active genes.
- GO Enrichment Analysis - run GO enrichment on each attractor-derived gene set using an appropriate background gene set.
- Aggregation of Enrichment Results - combine enrichment outputs across all network instances and also within each instance to identify robust, recurrent functional patterns.
- Output and visualization - produce structured outputs summarizing both instance-specific and global enrichment results.

3.2.1 Non-functional requirements

- Integration with AEON.py and external enrichment libraries must be seamless.
- Given identical input data, configuration, and random seeds, the pipeline must produce identical outputs

3.2.2 Constraints

- The PSBN adheres to the grammar defined in [10].
- AEON and AEON.py are available in the execution environment.
- PSBN must be fully and correctly annotated
- The pipeline relies on AEON.py for attractor computation under asynchronous dynamics
- GO enrichment analysis depends on external libraries or web services. This introduces constraints related to API availability, runtime limits, background gene definitions, and supported organisms.

3.3 Pipeline architecture

The pipeline is designed as a linear yet modular workflow, where each component transforms its input into a form consumed by the next stage.

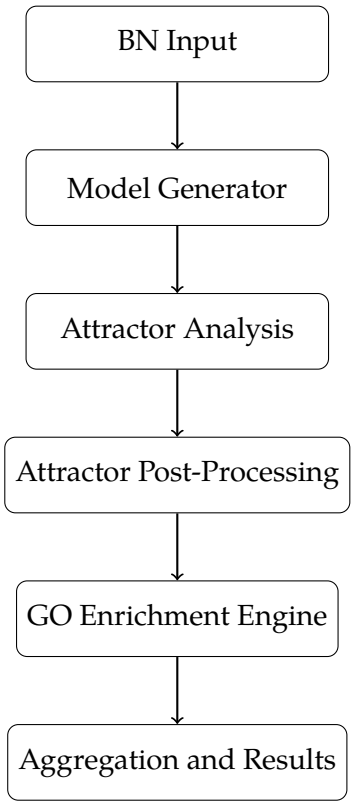

Figure 3.1: High-level architecture of the computational pipeline.

3.4 Component design

3.4.1 Input module

The input module serves as the entry point of the pipeline. It gathers the required external inputs and initializes the analysis setup prior to any network instantiation. This module handles:

- loading of BN models

- validation of network structure and syntax
- selection of organism-specific identifiers for GO enrichment
- initialization of global pipeline parameters, such as statistical thresholds

The input module strictly ensures that the pipeline starts from a consistent and well-defined configuration.

3.4.2 Model genration module

The model generation module is responsible for resolving the uncertainty encoded in the PSBN by constructing a set of valid, fully specified BN instances. The purpose of the model generation module is to resolve the uncertainty captured by a PSBN. It does so by translating partially specified update rules into a set of fully defined BN instances that remain consistent with the given logical restrictions. This module, therefore, defines the search space of models on which all subsequent dynamical and functional analyses operate.

3.4.3 Attractor analysis module

The attractor analysis module computes the long-term dynamical behaviour of each generated network instance. For every fully specified BN produced by the model generation module, this component:

- constructs the corresponding STG representation under asynchronous update semantics
- identifies all attractors of the network
- classifies the attractors according to their type

The output of this module is a structured set of attractors for each network instance. At this stage, attractors are still treated as abstract dynamical objects without direct biological interpretation.

3.4.4 Attractor post-processing

The attractor postprocessing module translates formal attractors into biologically interpretable objects. Its responsibilities include:

- extracting node activity patterns from each attractor
- distinguishing between activated and inhibited components
- grouping attractors into PSBN instances

- preparing attractor-derived gene sets for downstream enrichment analysis

This module establishes the crucial connection between dynamical stability and biological activity patterns, ensuring that attractors can be treated as functional gene sets in the subsequent analytical stages.

3.4.5 GO enrichment module

The GO enrichment module performs functional interpretation of attractor-derived gene sets using Gene Ontology. For each gene set associated with an attractor, the module:

- selects the appropriate GO category (BP, MF, or CC)
- defines an appropriate background gene universe
- applies a statistical enrichment test, multiple-testing correction

The output consists of enriched GO terms associated with each attractor, together with statistical significance measures and effect sizes. This module provides a biological meaning to dynamical states.

3.4.6 GO postprocessing/results module

The final module integrates and summarizes enrichment results across all levels of analysis. Its tasks include:

- aggregation of enrichment results across attractors within one network instance
- aggregation across all network instances generated from the PSBN
- identification of recurring functional patterns that persist across many valid instances
- detection of instance-specific enrichment effects

This module also prepares the final results for export and visualization. Its primary goal is to produce robust functional summaries that reflect both the uncertainty in regulatory structure and the stability of underlying biological interpretations.

3.5 Integration with AEON and AEON.py

From the design point of view, AEON serves as a standalone computation engine that handles all tasks related to state-space construction,

asynchronous dynamics, and attractor detection. The pipeline itself does not attempt to reproduce these low-level algorithms. Instead, it relies on AEON for the dynamical analysis and builds its own processing steps on top of the results produced by this backend. The AEON.py interface enables seamless integration of AEON's formal methods into the automated workflow of the pipeline. Through this interface, the pipeline can:

- construct network instances derived from PSBN
- trigger attractor computation procedures
- retrieve classified attractors and their structural properties

Pipeline controls the execution flow, while AEON provides verified results of dynamic analysis. The tight conceptual integration between the pipeline and AEON ensures that all reported biological interpretations are grounded in formally correct dynamical behaviour.

3.6 Complexity issues

The runtime behaviour of the pipeline is mainly driven by two separate sources of combinatorial growth. One comes from the number of possible instances of a PSBN, while the other arises from the size of the state space that must be explored for each instance. First, a PSBN may contain multiple uninterpreted function symbols or partially defined update rules, each of which admits several concrete Boolean interpretations. As a result, the number of fully specified network instances increases exponentially with the number of components whose logic is left unspecified. In addition, for any fully specified Boolean network with n variables, the corresponding asynchronous state-transition graph may contain up to 2^n states. Even though AEON relies on symbolic techniques, the computation and classification of attractors still exhibit exponential worst-case behaviour. When this analysis must be performed across all generated instances, the exponential growth in the number of models combines with the exponential growth of the state space, yielding a double-exponential worst-case complexity. For this reason, the experimental evaluation in this thesis is limited to PSBNs of modest size with only a small degree of underspecification. This choice keeps model enumeration, attractor computation, and

enrichment analysis computationally manageable, while still allowing the proposed methodology to be demonstrated.

3.7 Summary

This chapter presented the conceptual design of the pipeline developed in this thesis. It described the requirements, architecture, and design of each module, with emphasis on integration with the AEON framework and the need to process ensembles of instances arising from partial specifications. Chapter 4 is focused on the implementation details of this design, including code structure, algorithms, and technologies used.

4 Pipeline implementation

This chapter describes the concrete implementation of the computational pipeline introduced in chapter 3. While chapter 3 is focused on the conceptual architecture and design principles, the present chapter explains how the individual components were realised in practice, how data flows between them, and how the PSBN-based attractor enrichment workflow is executed end-to-end. All implementation work was carried out in a Google Colab environment, which provides a reproducible Python-based computational platform with support for interactive development, external package installation, and immediate visualisation of intermediate results. The pipeline is implemented in Python using the AEON.py interface together with standard scientific, data-processing, and graph-visualisation libraries. The structure of the code follows a modular design and is organised in 5 python files and 3 Jupyter notebooks to facilitate exploratory analysis for every case study. A project is available here `https://github.com/xsvaby/aeon_extension_bc_project`.
The implementation was first tested on a small BN, before applying the pipeline to the biological case studies . This step was used to verify that attractor computation, enrichment queries, and result aggregation worked correctly.
The notion "color" in the implementation reflects the notion "interpretation" in chapter 1.

4.1 Software environment and dependencies

The implementation was developed and executed in a Google Colab environment, which provides a Python-based interactive computing platform suitable for running AEON.py analyses, enrichment queries, and GO graph visualizations. All required packages were installed directly within the notebook using pip and apt. The following software components form the foundation of the pipeline:

- biodivine_aeon (version 1.3.0a3), which provides the symbolic analysis framework used for BN interpretation, parameter enumeration, and attractor computation under asynchronous update semantics.

- networkx, and matplotlib, which support the retrieval, processing, and visualisation of GO term relationships. These libraries are used to construct directed ontology graphs and render hierarchical subgraphs.
- graphviz, graphviz-dev, and pygraphviz, installed through apt and pip, enabling Graphviz-based layout algorithms for GO graph visualisation. Their availability allows the pipeline to render structured GO hierarchies rooted at enriched functional categories.
- The requests library, used for HTTP communication with external web services, including the PANTHER overrepresentation API (for GO enrichment) and the QuickGO API (for ontology structure queries).
- openpyxl, together with Python's os module, used to construct and update Excel-compatible output files containing attractor, class, and network level enrichment summaries.
- A set of custom Python classes and helper functions developed specifically for this thesis project. These comprise the enrichment data structures, aggregation utilities, and orchestration routines that implement the full PSBN analysis workflow.

BN and PSBN models are provided as .aeon files, which are directly supported by AEON.py. All attractor computations are performed under asynchronous update semantics, consistent with the formal framework described in chapter 1.

4.2 Enrichment data structures

To structure the enrichment-related information, the implementation defines a hierarchy of classes that represent individual GO terms, attractor-level enrichment profiles, PSBN-instance summaries, and PSBN-level aggregates. This design enables the pipeline to analyse enrichment at multiple abstraction levels while maintaining a consistent representation of GO terms and their attributes.

4.2.1 EnrichmentGOterm

Each enriched GO term returned by the PANTHER service is represented by an object that stores the essential information extracted from the enrichment output. This includes:

- the GO identifier,
- the textual label of the biological process,
- fold enrichment,
- false discovery rate (FDR),
- expected number of hits,
- number of genes in the reference set,
- p-value,
- the direction of enrichment.

In addition, each GO term object keeps track of its parents and children in the Gene Ontology graph. These relationships, obtained later via the QuickGO service, allow the implementation to reconstruct ontology subgraphs containing only the enriched terms and to visualise hierarchical relations among them.

4.2.2 EnrichmentAttractor

An enrichment attractor object links a single dynamical attractor to its associated functional annotation. For every attractor, the object stores:

- the attractor type,
- the set of enriched nodes (gene symbols or model variables),
- the GO term identifiers that pass the selected FDR threshold,
- a mapping from GO identifiers to GO term objects,
- sets of mapped and unmapped gene identifiers reported by the enrichment service.

Both successfully mapped and unmapped identifiers are collected and stored as sets, which allows gene coverage to be examined across different attractors and PSBN instances. Only GO terms that pass the FDR threshold and are not flagged as invalid by the enrichment tool are retained. The result is a concise enrichment summary for each attractor.

4.2.3 EnrichmentPSBNInstance

An enrichment PSBN instance represents all enrichment information derived from one fully specified BN. For each instance, the structure contains:

- all attractors produced by the instance,
- the attractor types,
- a dictionary of all GO terms appearing in any attractor in the instance,
- the instance identifier (referred to as its color in the implementation).

The instance object supports several operations that summarise enrichment within that instance:

- computing the intersection of GO term identifiers across all attractors,
- computing the intersection of GO term names across attractors,
- retrieving the GO term objects shared across attractors,
- counting the frequency of each GO term across attractors,
- analysing mapped and unmapped identifiers, including their frequencies and intersections.

These capabilities allow the pipeline to characterise functional signatures that consistently appear across the attractors of a particular PSBN instance and to identify attractor-specific differences.

4.2.4 EnrichmentPSBN

At the highest aggregation level, an enrichment PSBN object represents all PSBN instances obtained from a PSBN. It contains:

- a list of all PSBN instances,
- a global dictionary of all GO terms appearing in any instance.

This structure enables several cross-instance operations, including:

- computing intersections of GO term identifiers across all instances,
- computing intersections of GO term names across instances,
- retrieving GO term objects common to all instances,
- counting GO term frequencies aggregated across all instances and all their attractors,

- analysing mapped and unmapped gene identifiers globally, including their intersections and frequency statistics,
- counting the total number of attractors across all instances.

At the PSBN level, this representation allows the analysis of functional behaviour that is shared by all admissible instances. It also makes it possible to observe differences between instances, which arise from uncertainty in the original network specification.

4.3 GO enrichment module

The GO enrichment module is implemented as a set of functions that communicate with the PANTHER overrepresentation service. It takes attractor-derived gene sets as input and returns structured enrichment objects suitable for further analysis.

4.3.1 Gene set preparation

Before enrichment can be performed, the list of genes associated with an attractor must be converted into the format expected by the enrichment service. The implementation provides a helper function that:

- takes a set or list of gene identifiers,
- converts it into a comma-separated string,
- strips quotation marks and whitespace to ensure a valid encoding for use in URLs.

This function is used uniformly across the pipeline to avoid inconsistencies in formatting.

4.3.2 Enrichment query execution

Enrichment queries are executed using the HTTP-based API of the PANTHER overrepresentation service. For each attractor-derived gene list, the enrichment module:

- selects the GO aspect to be analysed - in this implementation the aspect is fixed to Biological Process,
- uses Fisher's exact test as the enrichment statistic,
- applies FDR as the multiple-testing correction method,
- sends a request specifying the gene list, organism identifier - fixed set to human id, and analysis parameters.

The service returns a JSON document containing the enrichment statistics. The implementation checks for the presence of error messages in the response and, if none are found, wraps the data into an EnrichmentResult object.

4.3.3 Enrichment result parsing

The JSON response from the enrichment service is converted into internal data structures as follows:

- The EnrichmentResult object stores information about the input gene list (organism, mapped and unmapped identifiers) and the list of enriched terms.
- For each enriched term, an EnrichmentGOterm instance is created, storing both statistical values and metadata.
- An EnrichmentAttractor object is created from the enrichment output. During this step, GO terms are filtered using the fixed set FDR threshold (0.05) and categories that are not of interest (for example, explicitly negative enrichments, if excluded) are removed. The remaining terms are stored as a concise representation of enrichment associated with a single attractor.

The resulting EnrichmentAttractor instances provide a compact and convenient representation of attractor-level enrichment.

4.4 Model interpretation and attractor analysis

This section describes how the pipeline interprets the PSBN model and computes attractors for each PSBN instance using the AEON.py interface.

4.4.1 Loading of network models

BN and PSBN models are loaded from .aeon files. In this work, three partially specified networks are analysed as case studies: a death receptor signalling network, a MAPK signalling network, and an interferon signalling network. The individual files describe the networks in a format accepted by AEON, including the list of variables, logical update rules, and partially defined expressions where the regulatory logic is incomplete.

4.4.2 Symbolic representation and parameter enumeration

For each PSBN, a symbolic representation of its state and parameter space is constructed using AEON's symbolic context. From this representation, the implementation extracts all admissible parameter assignments (colors). Each color encodes a particular interpretation of the unspecified update functions and therefore corresponds to one fully specified BN instance consistent with the original partial model.

4.4.3 Network instantiation and attractor computation

For each color obtained in the previous step, the implementation performs the following steps:

- instantiates the corresponding fully specified BN,
- constructs the asynchronous state-transition graph of the network,
- computes all attractors of the network symbolically,
- classifies attractors according to their type using AEON's classification routines,
- extracts stable phenotypes or node configurations associated with these attractors.

The classification output is then used to build attractor-level gene lists, which are passed to the enrichment module. For each parameter assignment, a new EnrichmentPSBNInstance is constructed to store the attractors and enrichment results associated with that specific color.

4.5 Attractor post-processing

Once attractors have been computed, they must be translated into biologically interpretable forms before enrichment can be applied. The implementation performs the following steps for each attractor:

- extracts the list of evaluated nodes using the sign convention introduced by the stable-phenotype classification,
- filters this list to obtain the subset of nodes that should be considered active for enrichment,
- converts the resulting node set into the input format required by the PANTHER service,

- constructs an EnrichmentAttractor object containing the attractor type, enriched gene set, and associated GO terms.

Each computed attractor is stored within its corresponding EnrichmentPSBNInstance. Along with the attractors, the instance keeps track of the parameter assignment (color) under which they were obtained. In this way, the uncertainty in parameter choice is explicitly linked to the observed attractor structure.

4.6 Implementation of enrichment aggregation

The aggregation logic is implemented primarily through the EnrichmentPSBN and EnrichmentPSBNInstance structures, each providing a set of operations for combining and analysing GO term sets.

4.6.1 Within-instance aggregation

A single PSBN instance corresponds to one fully specified interpretation of the original PSBN. Each instance contains one or more attractors, each with its own set of enriched GO terms. The EnrichmentPSBNInstance class provides methods to characterise functional behaviour at this level.

For each instance, the implementation computes:

- the intersection of GO term identifiers across all attractors in the instance,
- the intersection of GO term names across attractors,
- the mapping of intersecting GO term identifiers to their GO term objects,
- the set of GO terms that appear uniquely in individual attractors,
- frequencies of GO terms, measured as the number of attractors containing each term,
- intersections and frequency distributions for mapped and unmapped gene identifiers.

These computations allow the pipeline to distinguish between functional processes that are stable within the instance and those that are attractor-specific. Instance-level aggregation is implemented through methods such as goterm id intersection, goterm name intersection, goterm intersection, count go ids frequencies, count goterms frequen-

cies, unmapped ids intersection, count unmapped ids frequencies, and count mapped ids frequencies.

4.6.2 Cross-instance aggregation

Aggregation across instances is performed at the level of the entire PSBN. For this purpose, the pipeline uses the EnrichmentPSBN structure, which collects all instances and maintains a global dictionary of enriched GO terms.

For a given PSBN, the pipeline computes:

- intersections of GO term identifiers across all instances,
- intersections of GO term names across instances,
- the mapping of intersecting GO identifiers to GO term objects,
- global frequencies of GO terms across all attractors in all instances,
- global frequencies of mapped and unmapped identifiers,
- the total number of attractors analysed.

These operations make it possible to determine which biological processes are robust with respect to unresolved regulatory logic and which processes are sensitive to specific interpretations. The corresponding methods include goterms id intersection on all instances, goterms name intersection on all instances, goterms intersection on all instances, count go ids frequencies in all instances, count goterms frequencies in all instances, count unmapped ids frequencies in all instances, count mapped ids frequencies in all instances, and count attractors.

Together, the instance-level and PSBN-level aggregation mechanisms provide a structured approach for quantifying functional robustness and variability in the presence of logical uncertainty.

4.7 Export and result storage

To facilitate inspection and postprocessing, the implementation exports results into Excel-compatible .xlsx files. A helper function is used to append data as new columns to a given file, creating the file if it does not already exist.

For each network, the following outputs are generated:

- network_name_OnAttractors.xlsx – attractor-level enrichment profiles, organised by color and attractor index,
- network_name_OnColor.xlsx – intersections of GO terms within each behaviour class (color),
- network_name_OnAllColors.xlsx – GO term intersections across all behaviour classes of the network.

This file-based representation allows direct inspection in standard spreadsheet tools and simplifies the generation of tables and figures for the results chapter.

4.8 Visualization of GO term relationships

In addition to numerical aggregation, the pipeline includes a visualisation module designed to represent structural relationships between enriched GO terms. The implementation follows a multi-step process that combines GO-term intersections produced by the enrichment aggregation with ontology metadata retrieved from external resources. All visualisation routines are implemented in the accompanying Python module Visualization.py.

4.8.1 Retrieval of GO relationships

The visualisation workflow begins with a selected set of GO term identifiers, typically obtained from:

- the intersection of GO terms across attractors within a single PSBN instance,
- the intersection across all instances of a PSBN.

To reconstruct the ontology structure restricted to this set, the pipeline queries the QuickGO API in batch mode. The helper function get quickgo terms batch sends a single request for all terms and returns their associated metadata. For each term, the returned data includes its children together with relation types such as is a, part of, or regulates. The functions set nodes for the graph and get roots and leaves, process this metadata by:

- attaching parent and child references to the corresponding EnrichmentGOterm objects,

- identifying root terms, which have no parents within the selected subset,
- identifying leaf terms, which have no children within the subset.

This step reconstructs a reduced GO ontology tailored to the enriched terms relevant for the PSBN.

4.8.2 Graph construction and layout

Once the ontology relationships have been integrated into GO term objects, the pipeline constructs a directed graph using the NetworkX library. The function make graph creates a node for each GO term and an edge for each recorded parent–child relation, annotated with the relation type. For visualisation, the function visualize subgraphs produces subgraphs rooted at each identified root term. Each subgraph contains the root and all of its descendants. Graph visualizations are produced with Graphviz via the networkx-agraph interface. This setup is used to obtain hierarchical layouts that match the structure of the ontology. Nodes are labeled with biological process names, while edges indicate the corresponding relationship types.

4.8.3 Network-level and instance-level views

Two types of visualisations are supported:

- Network-level visualisation, produced by visualizing subgraphs on the whole net, shows subgraphs constructed from GO terms that are common to all PSBN instances. This reveals functional processes that are robust under all parameter interpretations.
- Instance-level visualisation, produced by visualizing subgraphs on each instance, renders subgraphs independently for each PSBN instance using its internal GO term intersections. This highlights how GO-term hierarchies vary across different interpretations.

These visualisations complement numerical summaries by revealing hierarchical organisation, shared functional roots, and the degree of specificity of enriched terms.

4.9 Execution of the complete workflow

The complete pipeline is executed using a network-specific orchestration function implemented in Pipeline.py. This function integrates all components of the pipeline and processes a PSBN from model loading to final enrichment visualisation. For each network, the workflow proceeds as follows:

1. A new EnrichmentPSBN object is created to store enrichment results across all instances of the PSBN.
2. The PSBN model is loaded, and a symbolic context is constructed.
3. All admissible colors, corresponding to fully specified instances of the PSBN, are generated.
4. For each color, a concrete BN instance is instantiated, and its asynchronous transition graph is constructed.
5. Stable states and attractors are computed using AEON, and for each attractor, the set of active nodes is extracted.
6. Each attractor is converted into an EnrichmentAttractor object by running GO enrichment and collecting mapped and unmapped identifiers.
7. All attractors corresponding to one color are grouped into an EnrichmentPSBNInstance.
8. The instance is added to the global EnrichmentPSBN structure.
9. The pipeline exports per-attractor results, per-instance intersections, and global intersections into spreadsheet files.
10. Optional visualisation is performed using the routines described in section 4.8.

In the current implementation, this workflow is applied to three biological networks: the death receptor signalling pathway, the MAPK pathway, and the interferon signalling pathway. Each network is analysed independently, producing a complete set of attractor enrichments, instance-level summaries, global PSBN-level summaries, and GO-term visualisations. These outputs form the basis for the analyses presented in chapter 5.

5 Results

This chapter presents the results of the attractor-based GO enrichment analysis performed on three PSBNs. Each case study is evaluated at three complementary levels: attractor-level, instance-level, and network-level. The analysis follows the workflow introduced in chapter 3 and chapter 4

5.1 Case study 1 – Death receptor signalling pathway

5.1.1 Background

This BN is adapted from the death-receptor signalling model of [13], which describes how cells integrate extrinsic death signals to choose between survival and different modes of cell death. This model is available in the Cell Collective via a researcher under the name Death Receptor Signaling. The BN consists of 28 nodes, and it includes three input nodes: FASL, TNF, and FADD, which initiate and trigger apoptotic or survival pathways. Every input node's update function was set as unspecified. The BN can reach one of three output states: survival, apoptosis, or NonACD. In cancer-related models, these outcomes correspond to resistance against death-receptor signalling, activation of programmed cell death, or non-apoptotic death caused by loss of cellular energy. When none of these outputs is active, the model describes a state of passive survival.

5.1.2 Attractor-level analysis

The PSBN produced 8 fully specified instances with a total of 27 attractors. Each attractor corresponds to a stable long-term phenotype, characterised by a specific output node and a correlated set of active internal nodes:

- Apoptosis attractors consistently activate the nodes CASP3, CASP8, BAX, MOMP, Cyt_c, SMAC, and ATP. Among these, CASP3, CASP8, and BAX are the most specific markers of apoptotic states, while components such as MOMP, Cyt_c, and SMAC are shared with NonACD states.

- Survival attractors are characterised by activation of the NFKB. Nodes consistently present include NFKB, IKK, cIAP, XIAP, cFLIP, BCL2, RIP1, RIP1ub, RIP1k, and ATP. These nodes do not form a mandatory core in apoptotic or NonACD attractors, making them specific markers of survival phenotypes.
- NonACD attractors activate the nodes ROS, MPT, MOMP, Cyt_c, and SMAC. Among these, ROS and MPT are the most specific to non-apoptotic cell death.
- Passive survival occurred when almost any nodes were active, except mainly ATP, or the input node, and sometimes cIAP.

5.1.3 Instance-level analysis

At the instance level, attractors were grouped according to their parameter assignment. In terms of input–output dependencies across instances, the results are:

- TNF activation consistently enables the TNFR, RIP1, and NFKB. In instances with TNF set to true, survival attractors always include TNF, TNFR, RIP1, RIP1ub, RIP1k, IKK, NFKB, cIAP, XIAP, cFLIP, BCL2, and ATP. The same input condition also permits a NonACD alternative, characterised by ROS and MPT activation, when TNF was off, passive survival could also occur, except in the case of FASL and FADD activation.
- FASL activation alone, without TNF, leads mainly to apoptotic or NonACD states, and survival is not observed. When FASL and FADD are active together, DISC_FAS is formed and survival can occur via NFKB signalling.
- FADD functions primarily as an adaptor rather than a phenotype switch. Its presence enables DISC formation when the corresponding ligand is active: DISC_FAS for FASL=1 and DISC_TNF for TNF=1. In these configurations, FADD appears in both death and survival attractors.

In terms of intersections of GO terms across attractors in this pathway, several instances showed empty intersections, meaning that their attractors did not share a common enriched term. For others, intersections contained apoptosis-related GO processes.
Roots and leaves based on intersections per instance show that most of the intersections are empty or very small, and so 4 instances had no

roots or leaves, 1 has only 1 root and 1 leaf. In those with non-empty intersections, roots typically corresponded to high-level apoptosis processes, while leaves corresponded to specific subtypes of cell death signalling. Also observable is little influence of FADD on rising numbers of roots and leaves in case TNF is active. The folowing graph visualizes relationships between obtained GO terms on concrete instance.

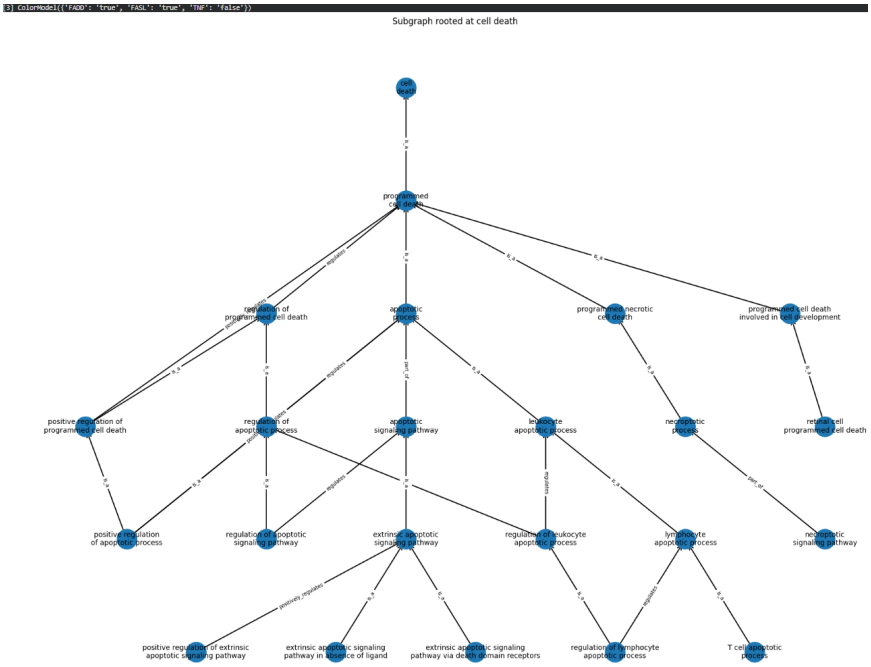

Figure 5.1: Example of graph on concrete instance of death receptor signalling pathway

5.1.4 Network-level analysis

At the network level, aggregation across all 8 instances was performed. Across all attractors, the frequencies of output phenotypes were:

Table 5.1: Output frequencies

Output	Frequency
NonACD	8
apoptosis	8
survival	5
(passive survival)	6

This confirms that all three cell-fate outcomes are repeatedly realised across different dynamical regimes.
Across all attractors, 355 enriched GO terms were identified. The most frequent terms are strongly dominated by apoptosis-related processes:

Table 5.2: The most frequent biological processes

Biological process	Frequency
extrinsic apoptotic signaling pathway via death domain receptors	16
apoptotic signaling pathway	15
positive regulation of apoptotic process	15
positive regulation of programmed cell death	15
extrinsic apoptotic signaling pathway	15
cell death	15
programmed necrotic cell death	15
T cell homeostasis	14
lymphocyte homeostasis	14
apoptotic process	14

These results confirm that, despite phenotypic differences between attractors, death-receptor-mediated apoptosis remains the dominant functional theme at the attractor level. It also confirms a correlation with the frequencies of output nodes.

Although apoptosis-related GO terms are highly frequent across attractors, the intersection of GO terms across all instances is empty. Consequently:

- no global GO-term intersection exists,
- no network-level GO graph could be constructed,
- no roots or leaves were identified at the whole-network level.

The absence of a globally invariant functional core indicates sensitivity of long-term functional behaviour to unresolved regulatory logic. While death-related processes dominate locally, their exact functional realisation depends on how the unspecified update rules are resolved in each instance.

5.2 Case study 2 – MAPK pathway

5.2.1 Background

This BN is based on the MAPK signalling model introduced by [14], which investigates how cancer cells integrate growth signals, stress responses, and DNA damage in order to decide between proliferation, growth arrest, or apoptosis. The model is publicly available in the Cell Collective via a researcher under the name MAPK Cancer Cell Fate Network. The BN consists of 53 nodes, including input nodes EGFR_stimulus, FGFR3_stimulus, TGFBR_stimulus, and DNA_damage, whose update functions are unspecified. Output nodes represent the possible cell-fate decisions: proliferation, growth arrest, and apoptosis.

5.2.2 Attractor-level analysis

The PSBN generated 16 fully specified instances with a total of 18 attractors. Across all attractors, only two output phenotypes were observed: growth arrest and apoptosis. Their frequencies were:

Table 5.3: Output frequecies

Output	Frequency
growth_arrest	13
apoptosis	13

No attractor exhibited an active Proliferation output, indicating that under asynchronous dynamics, the network does not support proliferation as a stable long-term state. Consequently, proliferation cannot be analysed at the attractor level, limiting the interpretation of cancer-related growth decisions in this model. According to Cell Collective analysis, the proliferation state (proliferation = 1) is reachable,

but it is such an unstable state that, in terms of attractors, it does not occur.
Growth arrest and apoptosis always co-occur in the identified attractors, implying that they share the same set of consistently active internal nodes. These shared nodes are: PPP2CA, FOXO3, p14, DUSP1, AP1, PTEN, p21, kinases (JNK, ATF2, p38, MTK1, p53, MSK), MYC, MAX, GADD45, CREB, ELK1 and JUN. Together, they represent a coordinated programme of cell-cycle inhibition, stress signalling, and pro-death regulation.

5.2.3 Instance-level analysis

At the instance level, most parameter configurations produced one or two attractors, so intersections of GO terms within the single attractor instances were the same as the attractors themselves. Intersection within instances with 2 attractors was empty except for the instance with DNA_damage on, others off.
Correlations between input nodes and output phenotypes emerge at this level.

- Whenever DNA_damage is active, attractors consistently include nodes as ATM, p53, p21, p14 and GADD45, together with shared kinases nodes.
- When TGFBR_stimulus is active, TGFBR, SMAD, and MAPK adaptor components are consistently present, and growth arrest and apoptosis are both observed.
- In contrast, when both DNA_damage and TGFBR_stimulus are inactive, attractors do not exhibit growth arrest or apoptosis, regardless of EGFR_stimulus or FGFR3_stimulus. Instead, these instances stabilise in non-phenotypic states with activated nodes: GAB1, PI3K, and PDK1, and sometimes AKT and MDM2. These states represent signalling activity without commitment to a stable cell-fate decision.

On the number of roots and leaves per instance, we can also see that they are highly dependent on the DNA_damage input node. When the node is active, there are 81 leaves and 35 roots, and in case of inactivity, the numbers are 69 leaves and 36 roots. Despite this fact, the most significant GO terms persist in both cases, such as cell death(one of roots), or cellular response to growth factor stimulus(one of leaves).

This subgraph shows relations between GO terms on specific instance, found by enrichment analysis. Additionally in the black part on the top of figure there is instance specification.

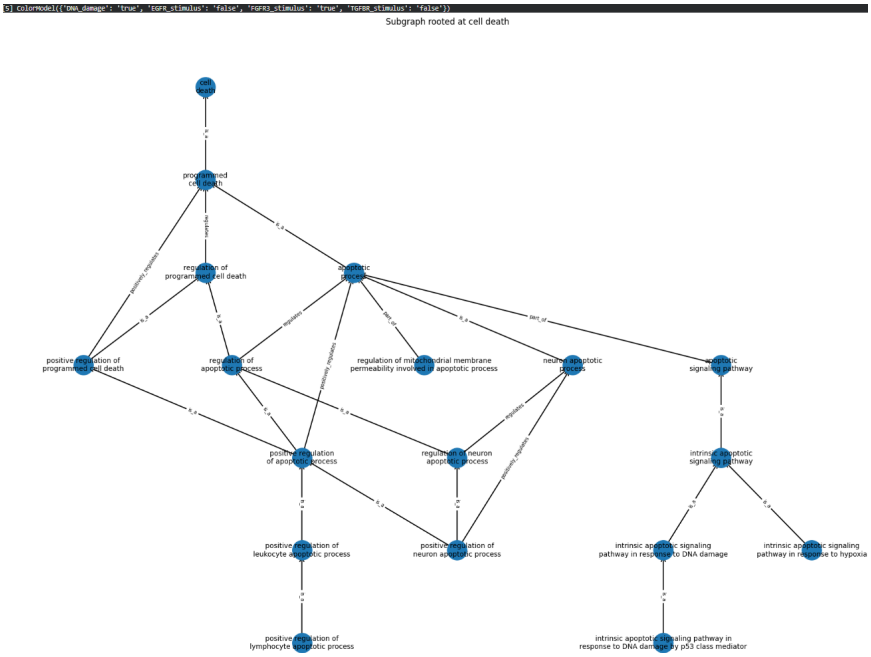

Figure 5.2: Example of graph on concrete instance of mapk pathway

5.2.4 Network-level analys

At the network level, aggregation across all 16 instances revealed no GO term that appeared in the intersection across every instance. Consequently:

- no global GO-term intersection exists,
- no network-level GO graph could be constructed,
- no roots or leaves were identified at the whole-network level.

At the GO level, 340 enriched biological processes were identified across all attractors. The most frequent terms are:

Table 5.4: The most frequent biological processes

Biological process	Frequency
intracellular signal transduction	13
response to abiotic stimulus	13
signal transduction in response to DNA damage	13
negative regulation of nucleobase-containing compound metabolic process	13
response to radiation	13
cellular response to ionizing radiation	13
cellular response to gamma radiation	13
cellular response to radiation	13
regulation of nucleobase-containing compound metabolic process	13
cellular response to abiotic stimulus	13

This behaviour matches the prevalence of stress- and damage-related attractors in the model. Growth arrest and apoptosis frequently co-occur, but the underlying functional processes are not shared by all instances.

5.3 Case study 3 – Interferon pathway

5.3.1 Background

This BN is adapted from the interferon signalling model described by [15], which captures how innate immune cells detect viral RNA and activate the type I interferon response. The network includes SARS-CoV-2 proteins (N, Nsp3, Nsp15, Orf6, Orf8, Orf9b) that inhibit signalling nodes, allowing simulation of immune evasion. The BN contains 87 nodes and includes two high-level output phenotypes: Immune_Response_phenotype and Inflammation_phenotype.

5.3.2 Attractor-level analysis

The PSBN generated 32 instances with a total of 48 attractors. Unmapped identifiers include both output phenotypes and multiple

viral components. Across all attractors, the two phenotypes appeared with equal frequency:

Table 5.5: Output frequecies

Output	Frequency
Inflammation_phenotype	48
Immune_Response_phenotype	16

Attractor-level correlation analysis reveals a strong modular structure. All attractors containing Immune_Response_phenotype shared a common signalling components IFNAR complex, JAK1, TYK2, STAT1, STAT2, IRF9, interferon regulators IRF7, TBK1 and IKBKE. These attractors also consistently contained the viral RNA sensing and restriction context, including viral proteins and reduced viral dsRNA, indicating that immune-response attractors correspond to an active antiviral state.
In contrast, Inflammation_phenotype was present in every attractor. Its core correlated nodes include MAPK14, MAPK8, JUN, FOS, AP_1_complex_cell, RELA, NFKBIA_NFKB1_component, RIPK1s, MYD88, TCAM1, TANK, TRIM25, RNF135, ITCH, and the NLRP3 inflammasome complex. These elements define a baseline inflammatory programme that persists independently of interferon activation. Importantly, components such as TBK1, IKBKE, and IRF7 were absent from some inflammation-only attractors, indicating that full interferon signalling is not required for inflammatory resolution.

5.3.3 Instance-level analysis

At the instance level, intersections of GO terms across attractors were non-empty for all instances, demonstrating stable functional behaviour within individual parameter assignments
Correlation analysis between inputs and phenotypes shows that Other inputs opposed to GRL0617_drug did not alter phenotype outcome:

- GRL0617_drug acts as a decisive switch for immune-response activation. Immune_Response_phenotype appeared if and only if GRL0617_drug was set to true. In these instances, two attractors typically coexisted: one representing inflammation alone

and one representing combined inflammation and interferon-mediated immune response.
- PAMP_signalling_phenotype, TREML4, Azithromycin_drug, NFKB1_cell did not alter the phenotype outcome.

Root and leaf analysis of GO subgraphs confirmed this stability. Roots consistently corresponded to high-level immune processes such as type I interferon-mediated signalling, cytokine-mediated signalling, and innate immune response. Leaves captured specific downstream mechanisms, including regulation of interferon-regulated transcription, viral genome replication control, and cytokine biosynthesis.

5.3.4 Network-level analysis

At the network level, the interferon PSBN exhibited a large non-empty GO-term intersection across all 32 instances. This enabled the full construction of a global GO graph, unlike in the death receptor and MAPK networks.

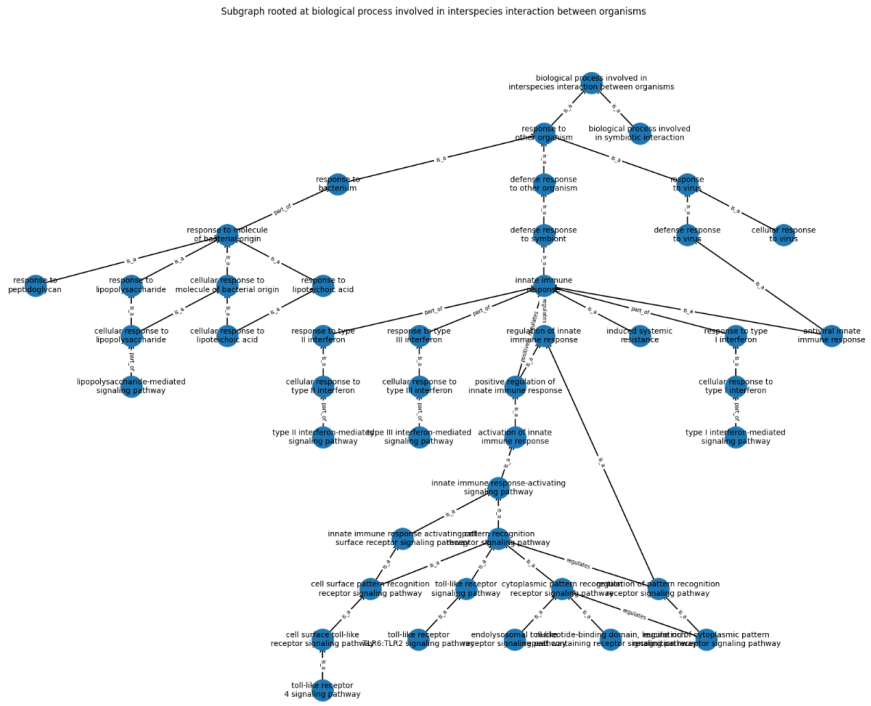

Figure 5.3: Example of subgraph visualising relationships between GO terms from interferon pathway

Network-level roots corresponded to the most general immune processes, including the type I interferon-mediated signalling pathway, the cytokine-mediated signalling pathway, the innate immune response, and the immune system process. Leaves represented highly specific antiviral and inflammatory functions, such as interferon-alpha signalling, regulation of viral genome replication, and positive regulation of cytokine biosynthetic processes.
GO enrichment at the attractor level showed complete functional consistency across all attractors. All 48 attractors were enriched for immune-dominant biological processes, the 10 most frequent are:

Table 5.6: The most frequent biological processes

Biological process	Frequency
response to virus	48
biological process involved in interspecies interaction between organisms	48
response to other organism	48
response to external biotic stimulus	48
response to biotic stimulus	48
defense response	48
response to external stimulus	48
defense response to virus	48
intracellular signal transduction	48
response to stress	48

Frequency analysis across all attractors further confirmed the robustness of this functional programme. Immune-related GO terms appeared in every attractor, and inflammatory signalling was invariant across all parameter assignments. This demonstrates that, despite regulatory uncertainty and viral inhibition mechanisms, the interferon network maintains a globally conserved immune-response signature. Overall, the interferon pathway differs fundamentally from the other two case studies: while death receptor and MAPK networks fragment into functionally distinct regimes depending on unresolved logic, the interferon network converges on a stable and biologically coherent immune-dominant behaviour at attractor, instance, and network levels.

6 Conclusion

This thesis introduced a complete workflow for analysing PSBNs through attractor-based functional interpretation using Gene Ontology enrichment. The proposed pipeline integrates model instantiation, dynamical analysis, enrichment computation, aggregation across attractors and parameter instances, graph-based visualisation of GO-term relationships, and visualisation of postprocessing data. The implementation was tested on a simple test case and validated on three biologically established pathways: death receptor signalling, the MAPK network, and the interferon signalling pathway.

Across all case studies, the obtained results aligned with the known biological behaviour of the respective pathways, with one exception in MAPK. In the death receptor model, apoptosis-related terms dominated the enrichment profiles. The MAPK network exhibited strong enrichment for signal transduction and stress-response pathways, consistent with its role in integrating environmental and oncogenic stimuli, but the problem lies in enrichment analysis of attractors - unstable, but significant behaviour was overlooked. The interferon signalling model produced a coherent and robust immune-response signature across all instances, including processes such as response to virus, interferon-mediated signalling, which corresponds precisely to the expected antiviral behaviour of this pathway. No unexpected or emergent functional behaviour outside the established biological context appeared in any of the analysed models.

From a methodological perspective, the pipeline performed reliably across all three pathways. It successfully handled the enumeration of fully specified instances, computed their attractors, conducted enrichment analysis, and aggregated results across attractors, instances, and whole networks. The visualisation of GO-term subgraphs further demonstrated that the pipeline can reconstruct interpretable functional hierarchies. The main limiting factor, computational complexity, remains: larger or more heavily underspecified networks rapidly lead to double-exponential growth in the number of instances and state-space size. Nevertheless, within the feasible model sizes considered here, the pipeline proved fully operational and robust.

The work presented in this thesis provides a foundation for several

potential extensions. First, the pipeline can be expanded to support larger networks by integrating more advanced symbolic computation techniques. Second, the enrichment layer can be extended to additional ontologies, such as molecular function or cellular component, or to pathway databases such as Reactome. Finally, the current implementation enables the analysis of user-defined PSBNs, which opens the possibility of applying this workflow to newly constructed models or to automated model inference frameworks.
Overall, this thesis demonstrates that systematic functional interpretation of dynamical behaviours in PSBNs is both feasible and informative. The developed pipeline offers a reusable software framework for future studies and provides a solid basis for advancing the integration of formal dynamical modelling with functional biological interpretation.

A An appendix

The complete source code, input models, case-study notebooks, and all output data used in this thesis are publicly available in a GitHub repository:

`https://github.com/xsvaby/aeon_extension_bc_project`

The repository contains the Python implementation of the PSBN attractor enrichment pipeline, AEON network models used as input, Jupyter notebooks for executing individual case studies, and the generated Excel files with enrichment results.

Bibliography

1. KHATRI, Purvesh; DRĂGHICI, Sorin. Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*. 2005, vol. 21, no. 18, pp. 3587–3595.
2. HUANG, Da Wei; SHERMAN, Brad T.; LEMPICKI, Richard A. Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucleic Acids Research*. 2009, vol. 37, no. 1, pp. 1–13.
3. ZHAO, Kangmei; RHEE, Seung Yon. Interpreting omics data with pathway enrichment analysis. *Trends in Genetics*. 2023, vol. 39, no. 4, pp. 308–319.
4. MERICO, Daniele; ISSERLIN, Ruth; STUEKER, Oliver; EMILI, Andrew; BADER, Gary D. Enrichment Map: a network-based method for gene-set enrichment visualization and interpretation. *PLOS ONE*. 2010, vol. 5, no. 11, e13984.
5. ASHBURNER, Michael; BALL, Catherine A.; BLAKE, Judith A.; BOTSTEIN, David; BUTLER, Heather; CHERRY, J. Michael; DAVIS, Allan P.; DOLINSKI, Kara; DWIGHT, Selina S.; EPPIG, Janan T.; THE GENE ONTOLOGY CONSORTIUM. Gene Ontology: tool for the unification of biology. *Nature Genetics*. 2000, vol. 25, no. 1, pp. 25–29.
6. ALEKSANDER, Suzi A.; BALHOFF, James; CARBON, Seth; CHERRY, J. Michael; DRABKIN, Harold J.; EBERT, Dustin; FEUERMANN, Marc; GAUDET, Pascale; HARRIS, Nomi L.; THE GENE ONTOLOGY CONSORTIUM. The Gene Ontology knowledgebase in 2023. *Genetics*. 2023, vol. 224, no. 1, iyad031.
7. MI, Huaiyu; MURUGANUJAN, Anushya; HUANG, Xiaosong; EBERT, Dustin; MILLS, Caitlin; GUO, Xinyu; THOMAS, Paul D. Protocol update for large-scale genome and gene function analysis with the PANTHER classification system (v. 14.0). *Nature Protocols*. 2019, vol. 14, no. 3, pp. 703–721.

8. THOMAS, Paul D.; EBERT, Dustin; MURUGANUJAN, Anushya; MUSHAYAHAMA, Tremayne; ALBOU, Laurent-Philippe; MI, Huaiyu. PANTHER: Making genome-scale phylogenetics accessible to all. *Protein Science*. 2022, vol. 31, no. 1, pp. 8–22.

9. SHERMAN, Brad T.; HAO, Ming; QIU, Ju; JIAO, Xiaoli; BASELER, Michael W.; LANE, H. Clifford; IMAMICHI, Tomozumi; CHANG, Weizhong. DAVID: a web server for functional enrichment analysis and functional annotation of gene lists (2021 update). *Nucleic Acids Research*. 2022, vol. 50, no. W1, W216–W221.

10. BENEŠ, Nikola; BRIM, Luboš; HUVAR, Ondřej; PASTVA, Samuel; ŠAFRÁNEK, David. Boolean network sketches: a unifying framework for logical model inference. *Bioinformatics*. 2023, vol. 39, no. 4, btad158.

11. BENEŠ, Nikola; BRIM, Luboš; KADLECAJ, Jakub; PASTVA, Samuel; ŠAFRÁNEK, David. AEON: attractor bifurcation analysis of parametrised Boolean networks. In: *Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21–24, 2020, Proceedings, Part I*. Springer, 2020, vol. 12224, pp. 569–581. Lecture Notes in Computer Science.

12. BENEŠ, Nikola; BRIM, Luboš; HUVAR, Ondřej; PASTVA, Samuel; ŠAFRÁNEK, David; ŠMIJÁKOVÁ, Eva. AEON.py: Python library for attractor analysis in asynchronous Boolean networks. *Bioinformatics*. 2022, vol. 38, no. 21, pp. 4978–4980.

13. CALZONE, Laurence; TOURNIER, Laurent; FOURQUET, Simon; THIEFFRY, Denis; ZHIVOTOVSKY, Boris; BARILLOT, Emmanuel; ZINOVYEV, Andrei. Mathematical modelling of cell-fate decision in response to death receptor engagement. *PLOS Computational Biology*. 2010, vol. 6, no. 3, e1000702.

14. GRIECO, Luca; CALZONE, Laurence; BERNARD-PIERROT, Isabelle; RADVANYI, Francois; KAHN-PERLES, Brigitte; THIEFFRY, Denis. Integrative modelling of the influence of MAPK network on cancer cell fate decision. *PLOS Computational Biology*. 2013, vol. 9, no. 10, e1003286.

15. BENEŠ, Nikola; BRIM, Luboš; KADLECAJ, Jakub; PASTVA, Samuel; ŠAFRÁNEK, David. Exploring attractor bifurcations in Boolean networks. *BMC Bioinformatics*. 2022, vol. 23, p. 173.