



BioNetParse: A Multimodal, LLM-Based System for Biochemical Network Analytics

Submitted by:

Yash Goyal(2022588), Devansh Kumar(2022152)

Independent Project

Instructor: Sriram K

Date: July 30, 2025

**Indraprastha Institute of Information Technology,
New Delhi**

Acknowledgements

We would like to express our sincere gratitude to our professor, **Dr. Sriram K**, for his invaluable guidance throughout the project. His mentorship and feedback were instrumental to our success. We would also like to acknowledge the continuous support and assistance from **Harika GL**.

Additionally, tools such as **ChatGPT**, **Gemini**, and **Mistral** provided valuable assistance in building, analyzing, and validating our pipeline. More details on the AI tools used can be found in Chapter [9](#).

Abstract

This report presents a multimodal system for analyzing biological networks through both visual and textual data. We explore the conversion of biological diagrams into reaction networks and subsequently into mathematical representations such as Ordinary Differential Equations (ODEs) and Jacobians. Leveraging open-source and proprietary AI tools, including Mistral, Gemini, and ChatGPT, our framework offers an interface for interpreting and querying biological networks. Comparisons among LLMs are presented, with Mistral emerging as a high-performing open-source option. This report provides insights into the implementation, performance, and future prospects of multimodal AI-powered biological modeling.

Contents

1 Acknowledgements	1
2 Abstract	2
3 Introduction	5
4 Hugging Face and other AI Models	7
5 BIOCHAM	9
5.1 Model Visualization and Reaction Parsing	9
5.2 Automatic ODE Generation	9
5.3 Parameter Handling and Limitations	10
5.4 Feedback Detection and Dynamical Behavior	10
5.5 Formal Verification and Semantics	10
5.6 Symbolic Function Synthesis	11
5.7 Flowchart of BIOCHAM Capabilities and Gaps	11
5.8 Conclusion and Integration into the Pipeline	12
6 Mechanisms of Regulation: Feedback, Feedforward, and Inhibition	13
6.1 Feedback and Feedforward Loops	13
6.2 Regulation and Inhibition	13
6.3 Effect of Edge Modifications on Network Dynamics	14
6.4 Implications for Modeling and Interpretation	15
7 Methodology	16
7.1 Overview	16
7.2 System Architecture	17
7.3 Text-to-Graph Conversion	17
7.3.1 Parsing and Tokenization	17
7.3.2 Graph Representation	18
7.3.3 Validation with BIOCHAM	18
7.4 Image-to-Graph Extraction	18

7.4.1	Multimodal Parsing Pipeline	18
7.4.2	Fallback and Robustness Strategies	19
7.5	Mathematical Modeling and Symbolic Analysis	19
7.5.1	ODE Derivation	19
7.5.2	Jacobian Computation	19
7.5.3	Graph-Theoretic Analysis	20
7.6	AI Model Evaluation and Integration	20
7.6.1	ChatGPT (GPT-4)	20
7.6.2	Gemini	21
7.6.3	Mistral (Open-Source)	21
7.7	User Interface and Workflow Integration	22
7.8	End-to-End Workflow Summary	22
8	Results and Discussion	23
8.1	Pipeline Development and Key Components	23
8.2	Integration of Large Language Models	24
8.3	LLM-Based Extraction of Reaction Networks from Images	24
8.3.1	Motivation for Using LLMs	25
8.3.2	Test Diagrams Used for Evaluation	25
8.3.3	Model-wise Results and Observations	26
8.3.4	Comparison and Evaluation	32
8.4	Illustrative Example	33
8.5	Discussion	33
9	Use of AI Tools and ChatGPT	34
10	Challenges and Problems Solved	36
10.1	Computational Challenges	36
10.2	Data Format Challenges	36
10.3	Model Assumptions and Limitations	36
10.4	Uncertainty in Interpretation and Parsing	37
10.5	Generalization to Broader Biological Networks	37
11	Future Work	38
12	Conclusion	39
13	Implementation Details	40
14	References	41

Introduction

Biological systems are inherently complex, characterized by intricate networks of molecular interactions that govern cellular behavior. These networks are often represented as directed graphs, where nodes denote molecular species, and edges signify biochemical reactions or regulatory influences. Such representations provide valuable information about both the structural organization and dynamic behavior of biological processes.

Although symbolic and mathematical representations of reaction networks allow for rigorous analysis using differential equations and algebraic methods, a substantial portion of biological data continues to exist in visual or diagrammatic formats. These image-based representations, commonly encountered in research articles and experimental documentation, pose significant challenges to computational modeling and integration. Bridging the semantic gap between visual diagrams and machine-readable symbolic forms remains a critical need for systems biology and computational bioengineering.

The objective of this project is to develop an end-to-end multimodal pipeline capable of translating between textual and image-based representations of biochemical networks. This pipeline enables the automatic extraction of reactions from network diagrams, the symbolic modeling of these reactions as systems of Ordinary Differential Equations (ODEs) and Jacobians, and the subsequent structural and dynamic analysis of the resulting models.

To address this challenge, we leverage recent advances in large language models (LLMs) and multimodal AI systems. Our approach combines proprietary models such as ChatGPT and Gemini with open-source alternatives like Mistral, integrated via robust backends including Hugging Face and Google APIs. These models are tasked with interpreting network diagrams and converting them into structured reaction formats suitable for downstream analysis.

A critical component of our methodology is the evaluation of these models in terms of precision, robustness, and feasibility of integration. Although models like ChatGPT offer strong reasoning capabilities, their closed-source nature and reliance on commercial APIs limit adaptability. In contrast, open-source models such as Mistral provide greater flexibility and are better suited for reproducible and customizable workflows.

This report presents the full development and evaluation of the proposed multimodal

system. It outlines the methodology for graph construction, symbolic equation generation, and integration of AI models. In addition, it highlights the challenges encountered, including ambiguities in diagram interpretation, the need for precise symbol standardization, and limitations in current model capabilities. Through this work, we aim to contribute a versatile tool for automated biochemical network modeling, capable of interfacing with both human-readable formats and machine-executable models.

Hugging Face and other AI Models

Large Language Models (LLMs) are a class of deep learning systems designed to understand, generate, and reason over natural language. These models, such as **BERT**, **GPT**, **T5**, and **BART**, are built on the transformer architecture and have redefined performance standards across various natural language processing tasks. Pre-trained on vast text corpora, they are capable of capturing contextual relationships and patterns, making them highly adaptable for specialized tasks. The Hugging Face platform provides a robust ecosystem of these pre-trained models along with pipelines and tools that simplify their integration into diverse workflows.

In this project, we adopted LLMs both as engines for domain-specific reasoning and as tools for automating complex tasks like text summarization and symbolic regression. For example, the **BART-based summarizer** (`facebook/bart-large-cnn`) allowed us to condense technical papers and datasets into concise summaries, while **Flan-T5** served as an interactive question-answering engine, assisting in the exploration of biological network properties.

Beyond text processing, a key challenge of our project was handling biological networks that are often presented as intricate diagrams. These diagrams, with nodes representing biological entities and edges indicating reactions, are difficult to interpret computationally. To address this, we explored vision-language AI models, including **Gemini 2.0 Flash**, **Donut**, and **Pix2Struct**. These models were used to transform biological network images into structured text descriptions, such as reaction expressions (e.g., $A + B \rightarrow C$). While these models worked well for simpler layouts, their performance degraded on more complex diagrams due to limitations in graph awareness, over-reliance on OCR, and the lack of training data tailored to scientific schematics.

To improve accuracy, we developed a hybrid pipeline that combined object detection techniques, such as those offered by Detectron2, with LLM-based symbolic reasoning. This approach generated more reliable text-based representations of biological networks, which then served as input for our analytical pipeline. Once converted to text, these reactions were processed through the *Biological Network Analyzer*, where directed graphs were constructed using **NetworkX**, ordinary differential equations (ODEs) were generated using **SymPy**, and symbolic regression was applied through **PySRegressor** with guidance

from LLMs. This integrated pipeline allowed us to extract mathematical models and dynamics directly from visual network data, as discussed in Chapter 7.

Several AI models and tools were central to our work. Hugging Face Transformers, particularly BART and Flan-T5, supported summarization, QA, and symbolic equation reasoning. Vision-language models such as Gemini 2.0 Flash provided image-to-text conversion, while Donut and Pix2Struct were used for parsing complex diagrams. We also evaluated **Mistral-Small-3.2-24B**, comparing its performance with Gemini to assess its suitability for structured network interpretation.

The integration of these AI models significantly streamlined the process of converting complex biological diagrams into computationally analyzable formats. The technical implementation of this pipeline is elaborated in Chapter 7, with performance benchmarks and key observations presented in Chapter 8. Challenges encountered during model integration, as well as potential directions for improvement, are further discussed in Chapter 10.

BIOCHAM

BIOCHAM (Biochemical Abstract Machine) was one of the foundational tools used during the initial stages of our project to model, analyze, and visualize biochemical networks. It is a domain-specific environment tailored for the formal representation and simulation of biochemical reaction systems. Throughout the early exploration of our pipeline, BIOCHAM played a critical role in validating graph topology, generating differential equations, and serving as a benchmark against which AI-generated outputs could be evaluated.

5.1 Model Visualization and Reaction Parsing

In the initial phase, BIOCHAM was employed to transform textual representations of biochemical reactions—such as $A + B \rightarrow C$ —into graphical network diagrams. The commands `draw_reactions` and `draw_influences` provided visual validation of the reaction map and interaction graph, respectively, (see [Figure 5.2](#) (Right) and [Figure 5.3](#)). These representations were essential for establishing the ground truth, against which the performance of image-based extraction models (e.g., GPT-4 and Mistral) could later be compared. The visualization outputs were especially useful for understanding the connectivity and influence flow within networks such as the minimal mitotic oscillator.

5.2 Automatic ODE Generation

One of BIOCHAM’s central strengths lies in its capability to automatically derive the system of Ordinary Differential Equations (ODEs) governing the network dynamics. Using the `list_ode` command, the system outputs a LaTeX-formatted set of differential equations that describe species concentration changes over time. These equations were then cross-validated against those produced by our custom SymPy-based pipeline. BIOCHAM thus served as a gold standard to ensure semantic and structural correctness in mathematical modeling.

5.3 Parameter Handling and Limitations

Although BIOCHAM supports detailed ODE generation, it does not inherently assign or infer parameter values such as reaction rate constants. Parameters like k_d , v_i , or Michaelis-Menten constants must be explicitly declared using `parameter(...)` directives. This manual specification introduces friction, especially in automated or large-scale workflows where parameter estimation is dynamic or data-driven. Moreover, parameter constraints must be defined by the user, limiting the tool's ability to explore parametric uncertainty or robustness natively.

5.4 Feedback Detection and Dynamical Behavior

An important observation during our usage of BIOCHAM was its inability to explicitly identify or classify network feedback loops. While the system allows the drawing of influence graphs and computation of steady states (e.g., using `reachable(steady(Species))`), it does not automatically label interactions as positive or negative feedback. To identify oscillatory circuits or conserved motifs, one must interpret the influence graphs manually or rely on semantic queries like `oscil(Cyclin)`. Consequently, we had to supplement BIOCHAM's output with our own feedback loop detection module to categorize motifs and dynamical patterns, especially for networks with regulatory cycles.

5.5 Formal Verification and Semantics

BIOCHAM supports multiple semantic layers, including differential, stochastic, and Boolean models. The differential semantics generate ODEs; stochastic semantics simulate networks as continuous-time Markov chains; and Boolean semantics model networks as asynchronous logical systems. Each of these paradigms enables different forms of analysis:

- Using `search_conservations`, we identified mass-conserved species sets.
- With CTL (Computation Tree Logic), BIOCHAM supports logical queries such as `reachable(steady(Cyclin))` or `oscil(Cyclin)` to infer properties like periodicity or convergence.
- Model reduction using `reduce_model` allows simplification while preserving CTL properties.

These features were instrumental in evaluating complex behaviors like oscillation and stability. However, their expressiveness is limited by the user's familiarity with temporal logic syntax and interpretation.

5.6 Symbolic Function Synthesis

BIOCHAM includes advanced functionality for the symbolic synthesis of reaction networks from real-valued mathematical expressions using `compile_from_expression`. For example, we tested the generation of a reaction system that mimics the cosine function over time. This demonstrated the tool’s potential in synthetic biology, particularly for computational tasks encoded biochemically. The simulation confirmed that the compiled system approximated the desired waveform under deterministic and stochastic semantics alike.

5.7 Flowchart of BIOCHAM Capabilities and Gaps

To better contextualize BIOCHAM’s role in our project, we summarize its strengths and limitations in the following flowchart:

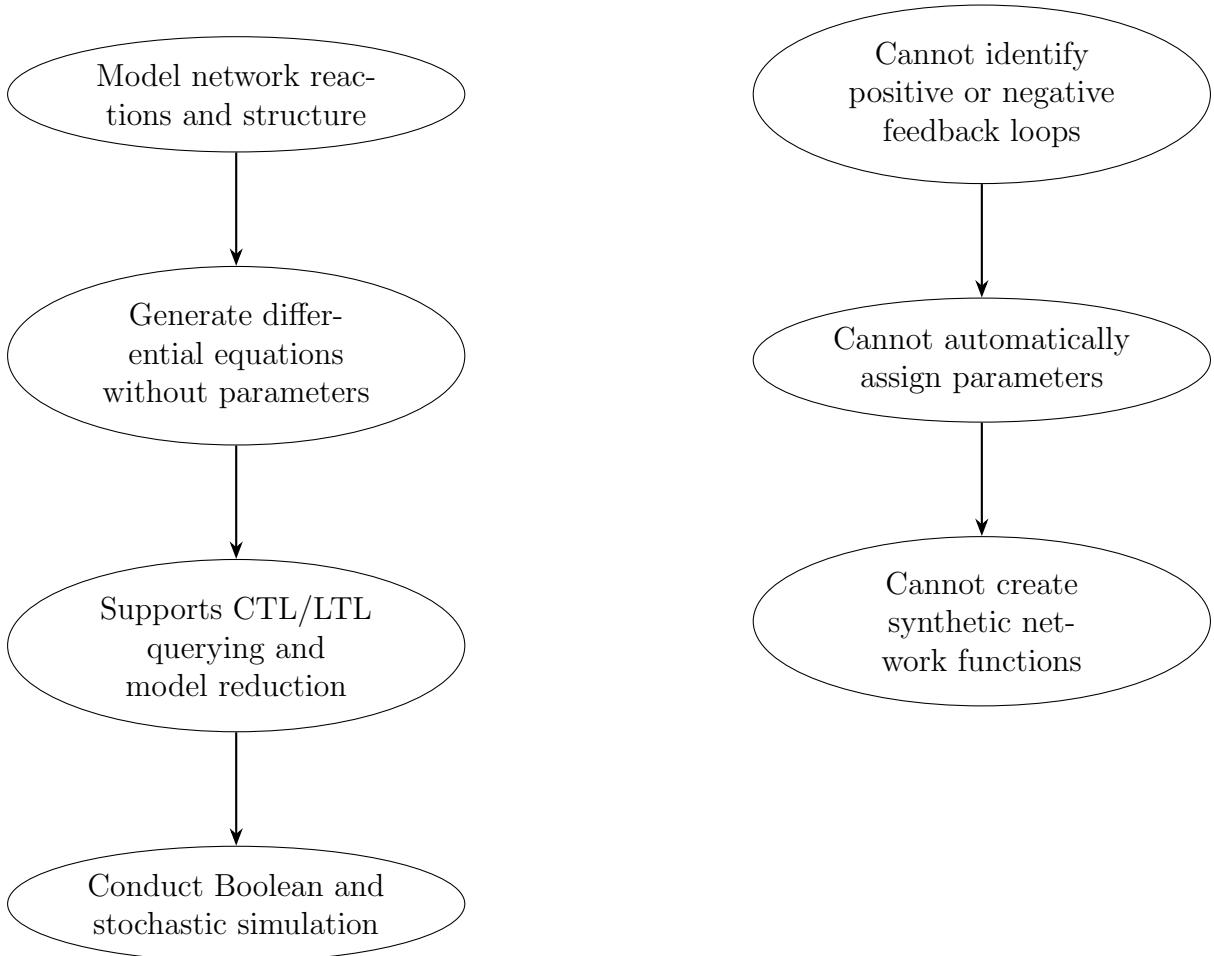


Figure 5.1: Flowchart of BIOCHAM Capabilities (left) and Limitations (right).

5.8 Conclusion and Integration into the Pipeline

BIOCHAM provided a reliable baseline for early modeling and for validating AI-parsed networks. Its graphical outputs, ODE generation, and logical queries enabled us to benchmark the multimodal pipeline. Despite requiring manual parameter specification and lacking automated feedback detection, it remained invaluable for cross-validation and comparative analysis throughout the project.

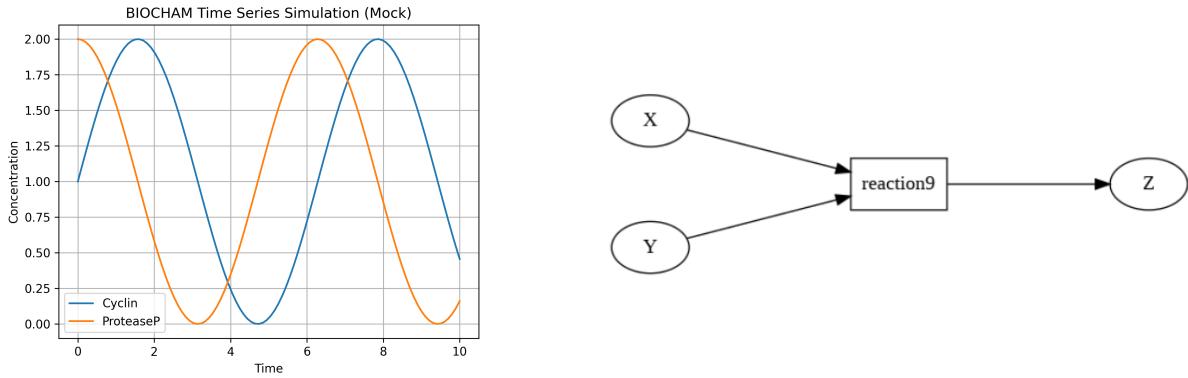


Figure 5.2: (Left) Mock time-series simulation showing the oscillatory behavior of Cyclin and ProteaseP, replicating BIOCHAM's dynamic output to validate periodic dynamics. (Right) A simple reaction diagram where species X and Y combine via reaction9 to form Z, used for testing initial parsing and symbolic equation generation.

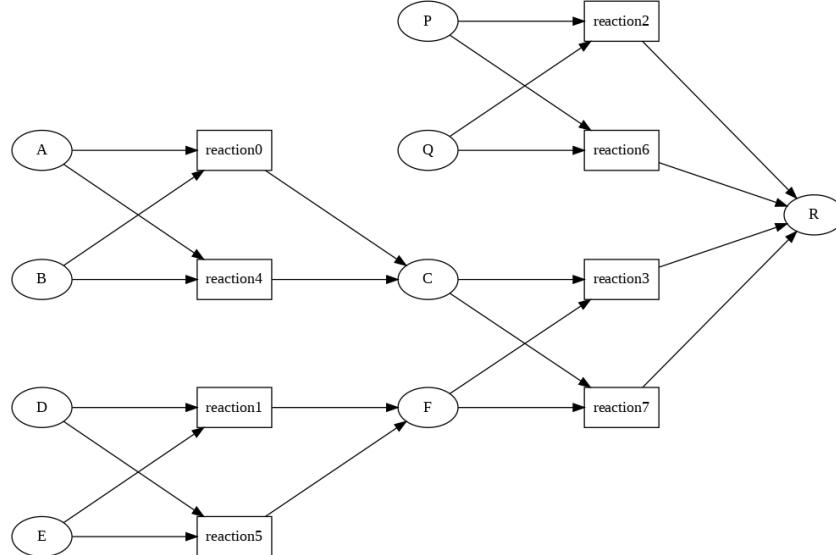


Figure 5.3: A multi-level biochemical reaction network diagram illustrating how species like A, B, D, E, P, and Q combine through intermediate nodes and reactions to ultimately produce the final product R. This serves as a representative test case for both BIOCHAM and AI-based extraction models.

Mechanisms of Regulation: Feedback, Feedforward, and Inhibition

Regulatory motifs such as feedback and feedforward loops are central to the dynamic behavior of biochemical systems. These motifs dictate how signals are propagated, amplified, or attenuated in reaction networks. Understanding their structural and functional implications is essential for interpreting biological models and for designing synthetic circuits with desired temporal behaviors.

6.1 Feedback and Feedforward Loops

A **feedback loop** is formed when a product or intermediate of a reaction pathway influences its own production, either directly or indirectly. Feedback can be:

- **Positive**, enhancing the production rate (autocatalysis or signal amplification),
- **Negative**, suppressing its own formation (oscillation or homeostasis).

Conversely, a **feedforward loop** involves a regulator that influences a downstream target both directly and indirectly through an intermediate. These motifs are often involved in filtering noise, accelerating responses, or generating pulse-like dynamics.

6.2 Regulation and Inhibition

Regulatory edges in a network diagram indicate influence over reaction dynamics. These are typically represented as arrows for activation and hammerhead edges (---|) for inhibition. Converting a regulatory edge into an inhibitory one can drastically alter the system's qualitative behavior, for instance, shifting from sustained oscillations to damped responses.

6.3 Effect of Edge Modifications on Network Dynamics

In biochemical networks, the qualitative behavior of a system, such as whether it exhibits oscillations, bistability, or convergence to a steady state, can depend sensitively on its topological structure. Small modifications, such as the addition, removal, or reclassification of a single regulatory edge, may lead to drastic changes in system dynamics. These regulatory rewiring effects are especially pronounced in motifs involving feedback and feedforward loops.

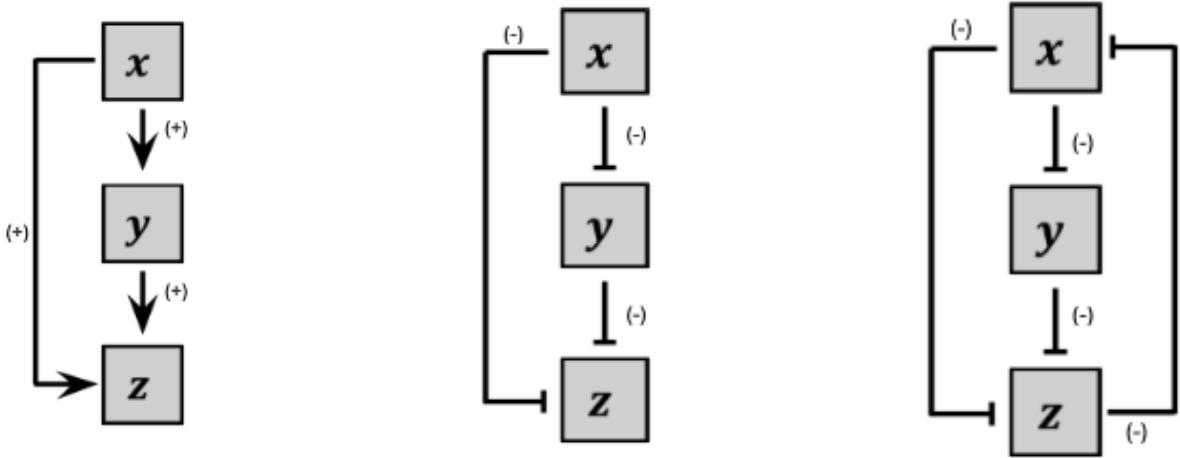


Figure 6.1: Topological rewiring of a biochemical motif. The leftmost diagram represents a coherent feedforward loop consisting solely of positive (activatory) interactions. In the middle, all activations are replaced with inhibitory edges, resulting in an incoherent motif with altered functional behavior. The rightmost configuration introduces an inhibitory feedback edge from terminal node z to upstream node x , converting the motif into a composite feedback–feedforward system. Such changes can invert regulatory influence and reshape the system’s dynamic class. Adapted from Harika et al. (2025) [0].

In the original feedforward loop (leftmost figure), node x activates y , and both x and y activate z . This constitutes a *coherent feedforward loop* where the effects reinforce each other, often associated with signal filtering or delay-insensitive activation.

In the second variant (middle), all edges are inhibitory. Here, x inhibits y , and both x and y inhibit z . This rewiring forms an *incoherent feedforward loop*, which may induce transient pulses or adaptation. The effective regulatory sign along a path can be computed using the product of individual edge signs. For example, the indirect path $x \rightarrow y \rightarrow z$ becomes:

$$\text{Regulation} = (-) \times (-) = (+),$$

indicating a net positive influence from x to z , despite both individual interactions being inhibitory. However, the direct edge from x to z remains inhibitory $(-)$, causing a regulatory conflict that drives dynamic complexity.

In the final motif (rightmost), an additional inhibitory edge is added from z back to x , forming a closed feedback loop. The presence of this *negative feedback* alters the system's stability and can give rise to oscillatory behavior. Depending on the polarity and number of inhibitions in the loop, the overall feedback can be classified as:

Positive feedback if the product of edge signs is positive (e.g., $(-) \times (-) = (+)$),

Negative feedback if the product is negative (e.g., $(-) \times (-) \times (-) = (-)$).

This classification is crucial for understanding the dynamical potential of motifs. Positive feedback loops often support multistability, while negative feedback can stabilize or destabilize steady states, or induce oscillations.

Therefore, accurate parsing and interpretation of edge types, particularly the distinction between activation and inhibition, is vital for downstream mathematical modeling. Even the smallest structural change, such as reversing an arrow or introducing a single inhibitory interaction, can transform the regulatory logic and shift the system between distinct dynamic regimes.

6.4 Implications for Modeling and Interpretation

When building or interpreting biochemical networks, particularly through automated pipelines such as those developed in this study, it is essential to carefully distinguish between activation and inhibition. Misclassifying a regulatory edge can lead to fundamentally different interpretations of system behavior. Equally important is the identification of regulatory loops, including both feedback and feedforward motifs, as these structures govern critical dynamic properties such as stability, oscillation, or bistability. Even small structural modifications, such as reversing an edge or replacing an arrow with a hammerhead, can shift a network from feedforward to feedback-driven dynamics, altering its qualitative behavior.

The pipeline implemented in this project incorporates logic for parsing and validating such regulatory relationships, ensuring that inferred models faithfully reflect the original biological intent. In future extensions, this framework will be expanded to support automatic motif classification and dynamic behavior prediction using structural heuristics and symbolic signatures derived from the underlying graph topology.

Methodology

7.1 Overview

The primary objective of this study is to develop a comprehensive multimodal pipeline that facilitates the automated translation between textual and visual representations of biological reaction networks, and subsequently derives their mathematical models, including Ordinary Differential Equations (ODEs) and Jacobians. Accurate and efficient reconstruction of such networks is crucial for downstream tasks such as symbolic analysis, dynamic simulation, and network property inference. To address this, our methodology integrates deterministic rule-based systems with advanced Large Language Models (LLMs) and symbolic computation frameworks.

The modeling pipeline is divided into six interconnected modules: system architecture design, text-to-graph conversion, image-based reaction extraction, symbolic modeling via differential equations, AI model evaluation, and user interface integration. Each module plays a distinct role in transforming raw textual or visual biological inputs into structured network models and formal dynamical representations.

At the core of the pipeline lies the conversion of biochemical reactions—whether provided as textual strings or embedded within images—into graph-based structures that reflect molecular species and their interactions. These graphs serve as the foundation for symbolic modeling using ODEs, where the rate of change of each species is described as a function of its participation in various reactions. The Jacobian matrix, a critical construct for sensitivity and stability analysis, is then derived from these equations.

Given the variability in input modalities and biological diagrams, multiple LLMs were evaluated for their robustness in interpreting noisy or ambiguous images. The evaluation included both proprietary models like ChatGPT and Gemini, as well as open-source alternatives such as Mistral. Each model was assessed based on its accuracy, generalization ability, and integration flexibility within the pipeline.

All components were implemented using Python, leveraging libraries such as `NetworkX`, `Sympy`, and `Gradio`. This ensured modularity, transparency, and ease of extension. The pipeline not only supports dynamic querying of network properties but also lays the

groundwork for future enhancements, including training of domain-specific LLMs and integration with simulation engines.

In summary, this study presents a structured and extensible approach to automated biological modeling by bridging advances in machine learning, symbolic computation, and systems biology. The methodology supports reproducible analysis while offering a flexible foundation for both research and educational applications.

7.2 System Architecture

The system is composed of two synergistic processing pipelines that ultimately converge at a shared analysis engine. The first is a text-based pipeline, which handles explicitly written reaction formulas. The second is an image-based pipeline, which interprets biological diagrams and converts them into reaction sets. Both pipelines subsequently feed into a unified module that constructs reaction graphs, derives mathematical models, and performs network analyses.

The **text-to-graph pipeline** begins with reaction strings such as $A + B \rightarrow C$, which are parsed using regular expressions to identify reactants and products. These parsed components are then used to construct a directed reaction graph through NetworkX, and this structure is optionally validated through a visual overlay using BIOCHAM.

In contrast, the **image-to-graph pipeline** processes biological network diagrams by leveraging large language models such as ChatGPT, Gemini, or Mistral. These models are prompted with image inputs and instructed to extract the underlying reaction schema. The extracted text undergoes normalization before being passed to the same graph-construction engine used in the text pipeline.

The final analysis engine performs symbolic modeling using SymPy, generates corresponding systems of differential equations and Jacobians, and supports dynamic interaction and visualization via a Gradio-based interface.

7.3 Text-to-Graph Conversion

7.3.1 Parsing and Tokenization

Reaction equations in textual form, such as:

$$A + B \rightarrow C$$

are first parsed using Python’s regular expression capabilities. The parser identifies and isolates reactants and products by detecting common delimiters such as $+$, \rightarrow , or $\rightarrow.$. In the case of stoichiometrically unbalanced reactions like $2A + B \rightarrow 3C$, the parsing

logic captures coefficients and standardizes them to ensure compatibility with the graph-construction phase. These coefficients are retained to preserve the integrity of the reaction model.

7.3.2 Graph Representation

Once parsed, the reactions are translated into a directed graph $G = (V, E)$ using NetworkX. Here, the node set V comprises all unique molecular species, while the edge set E includes directed connections from each reactant to each product. For instance, the reaction $A + B \rightarrow C$ produces two directed edges: one from A to C and another from B to C , thus formalizing the directionality of influence in the biochemical system. This construction allows downstream modules to operate on well-defined topological structures and simplifies further symbolic analysis.

7.3.3 Validation with BIOCHAM

To ensure correctness, parsed reaction graphs are cross-validated using BIOCHAM. The textual input is rendered into a canonical reaction diagram within the BIOCHAM environment, and this visual representation is compared with the graph generated via NetworkX. Both topological and stoichiometric consistency are examined, which serves as a sanity check for the parsing pipeline.

7.4 Image-to-Graph Extraction

7.4.1 Multimodal Parsing Pipeline

To handle diagrams, especially those found in academic figures or scanned documents, the system first uploads the image to a temporary hosting platform to obtain a public URL. This URL is then supplied as part of a multimodal prompt to an LLM such as ChatGPT, Gemini, or Mistral, along with a structured instruction designed to elicit the underlying reactions from the image.

The returned textual description is often noisy, containing inconsistent symbols (e.g., use of \rightarrow , \Rightarrow , or $\rightarrow\!$), varying spacing, or redundant characters. A normalization pipeline standardizes the notation and removes artifacts before the reaction list is passed into the graph-construction engine. This allows the image pipeline to benefit from the same robust symbolic analysis infrastructure used for textual inputs.

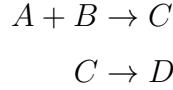
7.4.2 Fallback and Robustness Strategies

To address errors arising from image ambiguity, low resolution, or partial occlusion, multiple robustness mechanisms were incorporated. These include prompt reformulation to enhance the LLM’s reasoning or optical character recognition (OCR) accuracy, post-extraction validation by comparing extracted species against a known list of expected entities, and rule-based correction of common diagrammatic issues such as missing reaction arrows or misidentified nodes. These strategies improve the reliability of the extraction process across a wide range of image quality and styles.

7.5 Mathematical Modeling and Symbolic Analysis

7.5.1 ODE Derivation

Once a reaction graph is established, its dynamic behavior is modeled through a system of ordinary differential equations (ODEs). Consider the reaction set:



If $[X]$ denotes the concentration of species X , and assuming rate constants k_1 and k_2 for the two reactions respectively, the resulting ODE system is:

$$\begin{aligned} \frac{d[A]}{dt} &= -k_1[A][B] \\ \frac{d[B]}{dt} &= -k_1[A][B] \\ \frac{d[C]}{dt} &= k_1[A][B] - k_2[C] \\ \frac{d[D]}{dt} &= k_2[C] \end{aligned}$$

These equations are automatically generated from the parsed graph using symbolic differentiation provided by SymPy.

7.5.2 Jacobian Computation

To analyze the sensitivity and stability of the system, the Jacobian matrix J is computed. Each entry J_{ij} represents the partial derivative of the i^{th} rate equation with respect to

the j^{th} species concentration. For the system above, the Jacobian matrix is:

$$J = \begin{bmatrix} -k_1B & -k_1A & 0 & 0 \\ -k_1B & -k_1A & 0 & 0 \\ k_1B & k_1A & -k_2 & 0 \\ 0 & 0 & k_2 & 0 \end{bmatrix}$$

This formulation enables the application of numerical and symbolic stability analysis techniques.

7.5.3 Graph-Theoretic Analysis

In addition to differential modeling, the topological structure of the graph is examined to understand its biological and computational properties. The number of nodes corresponds to the unique molecular species, while the number of edges reflects the total reaction interactions. Connectivity metrics determine whether the system is globally reachable or composed of isolated subgraphs. The presence of cycles is also recorded, indicating potential feedback loops or autocatalytic structures.

For the earlier example, the reaction graph includes four nodes: {A, B, C, D}, and three directed edges: (A, C), (B, C), and (C, D). This configuration is acyclic and weakly connected, indicating a unidirectional reaction cascade without feedback.

7.6 AI Model Evaluation and Integration

A central component of this project involved evaluating the use of Large Language Models (LLMs) for parsing visual biochemical networks. The following subsections describe the specific integration and usage patterns of three models—ChatGPT, Gemini, and Mistral—within the multimodal pipeline.

This is the prompt we used in all these models "Analyze this network diagram and list the network only, e.g. Q + W → R. The arrows represent reactions. If there are multiple reactions, give them comma separated like A → B, B → C, etc."

7.6.1 ChatGPT (GPT-4)

ChatGPT was employed in a manual, exploratory role during the early phases of model development. Reaction diagrams were uploaded directly to the ChatGPT web interface, where carefully designed prompts were used to request structured reaction networks as output. This approach allowed rapid prototyping and qualitative assessment of the

model’s multimodal reasoning capabilities without requiring additional backend integration.

The typical prompt used in this setting instructed ChatGPT to interpret network diagrams and output only the parsed reaction equations, avoiding explanatory text. These outputs were then used for spot validation and to establish a reference for designing automated parsing pipelines.

7.6.2 Gemini

Gemini was integrated programmatically using Google’s `generativeai` Python module. The pipeline involved uploading images of reaction diagrams (e.g., hand-drawn sketches or software-generated networks) and querying the model using a structured prompt. The prompt asked Gemini to extract and return only the reaction network in a clean, comma-separated format.

The model was accessed via an API key and executed using the `gemini-2.0-flash` variant. Image inputs were passed using Python’s `PIL` module, and responses were captured programmatically for downstream processing. Gemini’s integration was designed for batch testing and automated extraction from image sources, particularly in the structured image-to-graph phase.

7.6.3 Mistral (Open-Source)

Mistral was the most tightly integrated model in the pipeline, serving as the default backend for multimodal parsing tasks. We used the `Mistral-Small-3.2-24B-Instruct-2506` variant, hosted via Unslloth on Hugging Face, and interfaced with it using the `InferenceClient` API through the `featherless-ai` provider. Authentication was handled via Hugging Face tokens, and the model was accessed using structured message sequences compatible with chat completion APIs.

Two core functions were implemented in Python to handle its integration. The first function, `extract_network_from_url`, accepts a public image URL and prompts the model to return only the reaction equations present in the diagram, excluding any commentary or explanation. The second function, `process_network`, allows the user to query additional information such as ODEs, Jacobians, species lists, edge lists, or cycle detection. It combines structured prompts with the parsed network data and optionally includes the image itself to augment the model’s reasoning.

Fallback logic was also implemented using Mistral. If the extracted network was incomplete or ambiguous, the model was re-prompted using a combination of parsed graph information and the image to refine the output. This approach enabled a flexible, interactive querying layer, fully integrated into our Gradio interface and usable offline in resource-constrained environments.

Together, these models were incorporated as modular components within the overall architecture. Each served a specific methodological purpose—manual validation (ChatGPT), structured batch inference (Gemini), and full programmatic integration with fall-back support (Mistral).

7.7 User Interface and Workflow Integration

A lightweight and user-friendly interface was implemented using **Gradio**, enabling researchers to interact with the entire modeling pipeline via a web browser. Users are given the option to input either a reaction description in text format or an image file. The back-end system automatically processes the input, constructs the associated reaction graph, and derives both ODEs and Jacobians. The results are displayed within the interface, and users can query various aspects of the model including network structure, dynamic behavior, and symbolic properties. This approach enables intuitive and reproducible experimentation without requiring coding expertise.

7.8 End-to-End Workflow Summary

The end-to-end process can be abstracted as a modular pipeline:

```
Text or Image Input → LLM Parsing → Graph Construction →  
ODE/Jacobian Generation → Visualization and Query Interface
```

This design ensures extensibility for future enhancements, such as dynamic simulation, incorporation of reaction kinetics, or integration with external biological knowledge bases. The modular separation between input parsing, symbolic modeling, and visualization promotes maintainability and facilitates ongoing research development.

Results and Discussion

This chapter presents the results of our work on designing and implementing an automated pipeline for analyzing biological networks. The primary goal was to create a system that can interpret both textual and diagrammatic representations of reaction networks, extract the underlying structure, and automatically generate symbolic mathematical models. Our pipeline combines multiple components, graph analysis, symbolic mathematics, and large language models (LLMs), to provide users with a robust, interactive tool for querying networks, generating mass-action ordinary differential equations (ODEs), and computing Jacobian matrices.

The pipeline was iteratively developed and refined, progressing from a basic text-based analyzer to an advanced multimodal system capable of understanding network diagrams using image-based inputs. This chapter describes the evolution of the pipeline, presents the key results from each stage of development, and discusses the performance of the integrated components.

8.1 Pipeline Development and Key Components

The initial version of the pipeline focused on interpreting text-based inputs of the form:

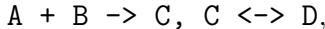
$$A \rightarrow B, \quad B \leftarrow \rightarrow C, \quad C \rightarrow D,$$

where nodes represent species and edges represent reactions. We developed a custom parser to extract directed and reversible reactions from the input and represent them as a directed graph using the `NetworkX` library. This graph structure formed the foundation for subsequent computations, allowing us to determine essential properties such as the number of variables, the set of edges, and the presence of cycles.

We then introduced a symbolic ODE generator based on the principles of mass-action kinetics. Each reaction was translated into a rate equation, and species concentrations were represented as symbolic variables using `Sympy`. For example, the reaction $A \rightarrow B$ with rate constant k_1 contributes $-k_1 A$ to the rate of change of A and $+k_1 A$ to the rate of change of B . This approach enabled the automatic construction of full ODE systems

for any given network.

As the pipeline evolved, support for multi-reactant reactions such as:



was introduced. This required modifying the parser and rate computation to handle products of species concentrations. In parallel, we implemented the symbolic computation of Jacobian matrices, which are essential for studying the stability and sensitivity of dynamic systems. Using the ODE expressions generated by the pipeline, the Jacobian was computed directly using symbolic differentiation.

8.2 Integration of Large Language Models

A key enhancement to the pipeline was the incorporation of LLMs to create an intuitive, natural language interface. Initially, the pipeline employed `ChatGPT` (GPT-4) through the `transformers` library, which allowed users to ask high-level questions such as:

*“How many variables are present in this network?” “What are the edges?”
“Is the network cyclic?”*

The model provided accurate and context-aware responses in most cases.

To broaden our evaluation, we compared `ChatGPT` and `Gemini`, and `Mistral-Small-3.2-24B-Instruct-2506`. While ChatGPT achieved the highest accuracy, Mistral demonstrated strong performance combined with the flexibility of open-source deployment. Gemini’s performance was competitive, but its integration was less seamless compared to the other models.

A major milestone was the extension of the pipeline to multimodal analysis. Using Mistral’s vision capabilities, we enabled the extraction of reaction networks directly from biological diagrams. For example, given a diagram with arrows denoting reactions, the system could output:



which was then processed by the text-based pipeline for further analysis.

8.3 LLM-Based Extraction of Reaction Networks from Images

A major extension of our pipeline was the development of a multimodal interface that can interpret biological network diagrams and convert them into structured textual reactions. This capability allows users to bypass manual transcription and directly analyze network

images using symbolic mathematics and graph-based analysis. The central challenge here is to accurately translate a visual representation—often consisting of nodes, arrows, and reaction labels—into a well-defined set of reactions suitable for model generation.

To address this, we explored the use of Large Language Models (LLMs) with vision capabilities, as traditional image processing or rule-based methods proved too brittle for the diversity found in biological diagrams. LLMs trained on multimodal data can understand contextual visual information and generate textual descriptions that reflect relationships and symbols within the image. For our purposes, this meant recognizing species, arrows, reactants and products, and forming clean reaction equations such as “A + B → C” or “C D.”

8.3.1 Motivation for Using LLMs

The decision to use LLMs was driven by the limitations of conventional computer vision techniques, which often fail to generalize across diagrams with varying layouts, fonts, or annotation styles. Unlike rule-based or OCR-based methods, LLMs are capable of high-level reasoning and can infer relationships even when the diagram is imperfect or lacks explicit labels. Their ability to understand symbolic representations within a visual context made them a natural choice for this task.

Moreover, using LLMs allows for rapid integration into the pipeline without building separate parsing rules for each new diagram style. Given the input of a network image, the LLM outputs a set of reactions, which is then passed to the symbolic ODE and Jacobian generator modules. This streamlines the modeling process and significantly reduces manual intervention.

8.3.2 Test Diagrams Used for Evaluation

To evaluate the capability of vision-language models in interpreting biological network diagrams, we curated a diverse set of five input images, shown in Figure 8.1. These diagrams span a range of complexity, visual styles, and structural patterns to test model performance under varied conditions.

- **Multi-reactant converging reaction (Figure 8.1a):** This diagram shows a reaction where two inputs, **A** and **B**, converge into a single output **C** via an intermediate reaction node. It tests the model’s understanding of converging inputs and intermediate process representation.
- **Dense and complex signaling network (Figure 8.1b):** A biologically realistic diagram using Biocham featuring numerous entities and multiple reactions (labeled `reaction0` to `reaction7`), including feedback loops and overlapping paths. It simulates the complexity of real-world biochemical networks.

- **Hand-drawn cyclic network (Figure 8.1c):** A manually sketched network forming a cycle: **A** → **B** → **C** → **D** → **A**. It evaluates robustness to informal, noisy, or handwritten input.
- **Bidirectional reaction with regulatory arrow (Figure 8.1d):** Shows a reversible reaction between **S5** and **S5S** using thick bidirectional arrows, with a vertical control or regulatory arrow pointing toward one node. This tests the model's understanding of directionality and regulatory notation.
- **Annotated modular pathway (Figure 8.1e):** A stylized pathway with modular entities (**S**, **Sp**, **E1**, **E2**, etc.) and reaction labels (**R₁** to **R₆**). This dense layout combines enzyme-substrate interactions and phosphorylation dynamics, testing detailed structural interpretation.

Each image was evaluated by multiple models for correctness, consistency, and fidelity to the original network structure.

8.3.3 Model-wise Results and Observations

Simulation 1: Multi-Reactant Converging Reaction

This image (see Figure 8.1a) tests the model's ability to interpret multiple input species converging at a single reaction node. See Figure 8.2 for visual outputs from the models.

ChatGPT: Accurately extracted the reaction structure, identifying all species and the correct directionality.

Gemini: Correctly captured the multi-input structure, but used inconsistent labeling for reactants.

Mistral: Correct overall interpretation, though occasionally omitted a label or misaligned a node.

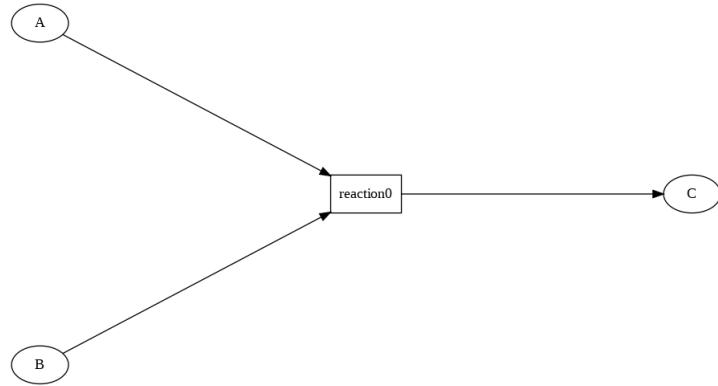
Simulation 2: Dense and Complex Signaling Network

This image (see Figure 8.1b) tests model performance on dense networks with many species, reactions, and interconnections. See Figure 8.3 for visual outputs.

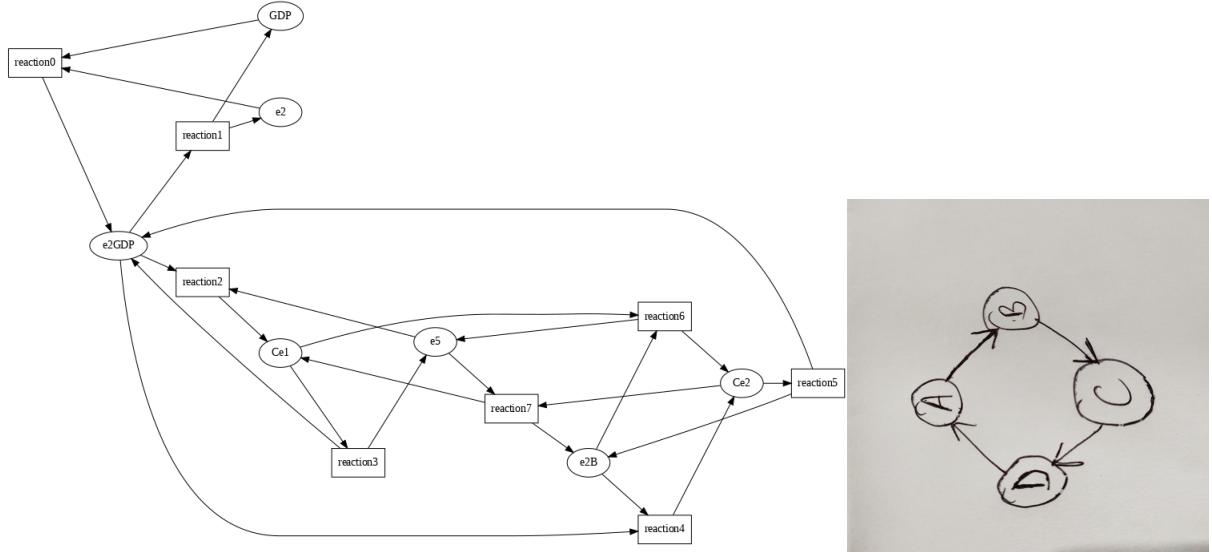
ChatGPT: Parsed a significant portion of the network, maintaining most labels and reactions, though omissions were observed in deeper branches and not able to identify reversible reactions.

Gemini: Identified key hubs and paths but struggled with overlapping arrows and node proximity.

Mistral: Generated a sparse interpretation, missing more than 40% of entities and edges.

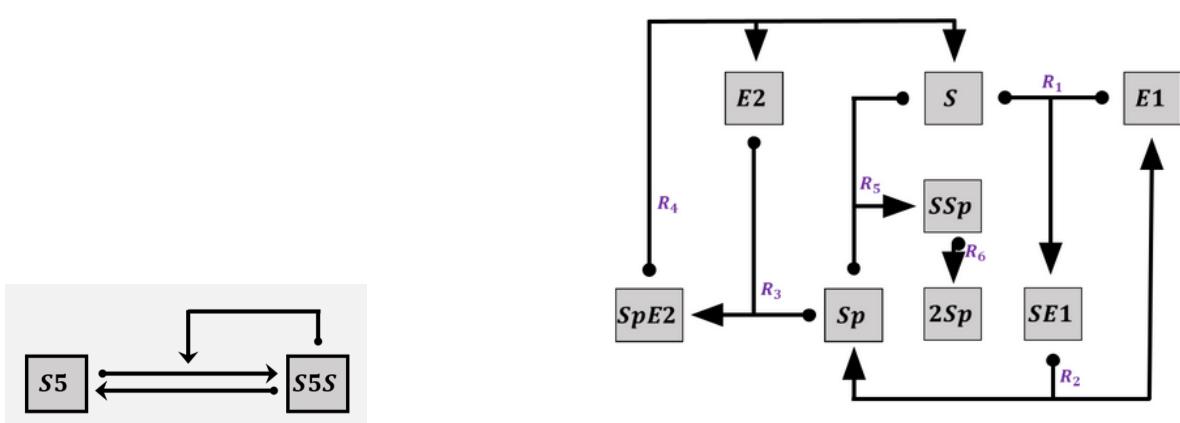


(a) Multi-reactant converging reaction



(b) Dense and complex signaling network

(c) Hand-drawn cyclic network



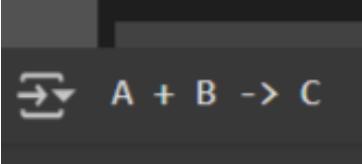
(d) Bidirectional reaction with regulation

(e) Annotated modular pathway

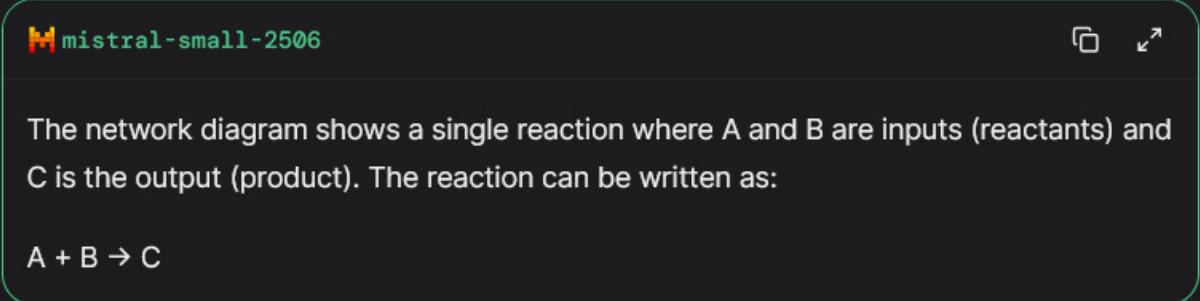
Figure 8.1: Input diagrams used to test vision-language models. Each image presents a unique challenge in terms of layout, structure, or visual clarity.

Final Reaction:
 $A + B \rightarrow C$

(a) ChatGPT Output


 $\text{A} + \text{B} \rightarrow \text{C}$

(b) Gemini Output


mistral-small-2506
The network diagram shows a single reaction where A and B are inputs (reactants) and C is the output (product). The reaction can be written as:
 $A + B \rightarrow C$

(c) Mistral Output

Figure 8.2: Model outputs for Simulation 1 showing a simple converging reaction where two reactants (A and B) combine to form product C. [\[gemini_code.py\]](#) [\[mistral_code.py\]](#).

Simulation 3: Hand-Drawn Cyclic Network

This image (see Figure 8.1c) evaluates the model’s robustness to hand-drawn, informal diagrams. See Figure 8.4 for visual outputs.

ChatGPT: Successfully reconstructed the cyclic path with all nodes and directions intact.

Gemini: Detected the cyclic layout but occasionally misread hand-drawn arrows.

Mistral: Failed to complete the cycle in most attempts; nodes were often labeled incorrectly or omitted.

Simulation 4: Bidirectional Reaction with Regulation

This image (see Figure 8.1d) tests interpretation of reversible reactions with stylized arrowheads and regulatory cues. See Figure 8.5.

ChatGPT: Correctly recognized bidirectional flow but not able to identify self-loop.

Gemini: Reconstructed the reaction path but confused the regulation arrow with the hammer head arrow.

Mistral: Successfully able to identify the network including bidirectional arrows and self loop.

Simulation 5: Annotated Modular Pathway

This image (see Figure 8.1e) evaluates the ability to interpret complex diagrams with modular layouts, embedded text, and layered semantics. See Figure 8.6.

Final Reactions (Simplified Summary)

This is a consecutive reaction pathway with multiple intermediate steps. A high-level reaction chain could be expressed like this:

1. $e2 + GDP \rightarrow e2GDP$
2. $e2GDP + Ce1 \rightarrow e5$
3. $e5 + Ce2 \rightarrow e2B$
4. $e2B \rightarrow Ce2$ (via loop)
5. $e2B \rightarrow e2GDP$ (recycled back)

Or more compactly, if we remove the intermediates for a top-level reaction flow:

$$e2 + GDP + Ce1 + Ce2 \rightarrow e2B \rightarrow e2GDP \text{ (recycled)}$$

(a) ChatGPT Output

```
reaction0 -> e2GDP,
e2 -> reaction1,
GDP -> reaction1,
reaction1 -> e2GDP,
Ce1 -> reaction2,
e5 -> reaction2,
reaction2 -> e2GDP,
Ce1 -> reaction3,
e5 -> reaction3,
e2B -> reaction4,
Ce2 -> reaction5,
Ce1 -> reaction6,
e5 -> reaction6,
e2GDP -> reaction6,
reaction6 -> Ce2,
reaction6 -> e2B,
Ce1 -> reaction7,
e5 -> reaction7,
reaction7 -> e2B,
reaction7 -> Ce2,
e2B -> reaction4,
e2B -> e2GDP
```

(b) Gemini Output

Mistral-small-2506

The network diagram can be described as follows:

reaction0 → e2GDP, reaction0 → GDP, reaction1 → e2, reaction1 → GDP, e2 → e2GDP, e2GDP → reaction2, e2GDP → reaction3, e2GDP → reaction6, e2GDP → reaction7, reaction2 → Ce1, reaction3 → e5, reaction3 → Ce1, e5 → reaction6, e5 → reaction7, e5 → Ce2, Ce1 → reaction6, Ce1 → reaction7, reaction6 → Ce2, reaction6 -> e2B, reaction7 → e2B, Ce2 → reaction5, e2B → reaction4, e2B → reaction5, reaction4 → e2GDP, reaction5 → e2GDP, reaction6 → e2GDP.

(c) Mistral Output

Figure 8.3: Model outputs for Simulation 2 illustrating a complex biochemical network with multiple interconnected reactions and dense node topology. [\[gemini_code.py\]](#) [\[mistral_code.py\]](#).

Final Reaction Summary
 $A \rightarrow B \rightarrow C \rightarrow D \rightarrow A$

(a) ChatGPT Output

→ A → B, B → C, C → D, D → A

(b) Gemini Output

Mistral-small-2506
The network diagram can be described as follows:
 $A \rightarrow B, B \rightarrow C, C \rightarrow A$

(c) Mistral Output

Figure 8.4: Model outputs for Simulation 3 demonstrating a cyclic hand-drawn pathway with informal notation and unstructured layout. [\[gemini_code.py\]](#) [\[mistral_code.py\]](#).

Reaction Summary
 $S5 \leftrightarrow S5S$

(a) ChatGPT Output

→ S5 ↔ S5S, S5S -| S5

(b) Gemini Output

Mistral-small-2506
The network diagram can be described as follows:
 $S5 \rightarrow S5S, S5S \rightarrow S5, S5S \rightarrow S5S$

(c) Mistral Output

Figure 8.5: Model outputs for Simulation 4 involving a bidirectional reaction with self-regulation, testing recognition of reversible arrows and feedback loops. [\[gemini_code.py\]](#) [\[mistral_code.py\]](#).

Reaction Summary (R₁ to R₆):

1. R₁:
$$S + E1 \rightarrow SE1$$
2. R₂:
$$SE1 \rightarrow Sp + E1$$
3. R₃:
$$Sp + E2 \rightarrow SpE2$$
4. R₄:
$$SpE2 \rightarrow S + E2$$
5. R₅:
$$Sp + Sp \rightarrow SSp \quad (\text{or}) \quad Sp \rightarrow SSp$$

(depending on whether dimerization or second phosphorylation is intended)

6. R₆:
$$SSp \rightarrow 2Sp$$

(a) ChatGPT Output

```
S + E1 -> SE1, SE1 -> E1 + Sp, E2 + Sp -> SpE2, SpE2 -> E2 + Sp, S -> SSp, SSp -> 2Sp
```

(b) Gemini Output

Mistral-small-2506

The network diagram can be described as follows:

E2 → SpE2, SpE2 → Sp, Sp → SSp, S → SSp, SSp → 2Sp, 2Sp → Sp, Sp → SE1, SE1 → E1, E1 → E2, E2 → S, S → SE1, SE1 → S, SpE2 → E2, E2 → E1, E1 → SE1, SE1 → E1.

(c) Mistral Output

Figure 8.6: Model outputs for Simulation 5 containing a modular pathway diagram with embedded annotations and multiple reaction modules labeled (e.g., R₁, R₂)
[\[gemini_code.py\]](#) [\[mistral_code.py\]](#).

ChatGPT: Extracted a structurally consistent representation of most modules and edges.

Gemini: Preserved the core structure but failed to map many reaction labels (e.g., R₁, R₄).

Mistral: Omitted most of the labels and connections, returning a minimal layout.

Other Models Tested

We also tested several additional models, including LLaVA, BLIP-2 and Donut. Most of these struggled with even simple inputs. Common issues included:

- Misidentifying arrows as lines or incorrect connections between nodes.
- Omitting species labels or confusing them with arrow annotations.
- Producing hallucinated reactions that did not appear in the input diagram.

These results underline the importance of using vision-language models that are either pretrained on or fine-tuned for structured diagrams, biological networks, and symbolic content. The simulations are available at [\[huggingfacellms_code.py\]](#).

8.3.4 Comparison and Evaluation

ChatGPT consistently produced accurate and parser-friendly outputs, correctly identifying species, reaction directionality, and connectivity with near-perfect fidelity. Its responses were the most reliable across a variety of diagram types, including complex and handwritten inputs.

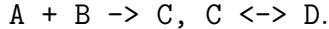
Gemini performed reasonably well on simpler diagrams but frequently omitted edges or failed to capture the full structure of reversible reactions. Its accuracy declined with increasing diagram complexity, and it struggled with dense or overlapping components.

Mistral’s outputs were generally correct, with occasional misinterpretations in crowded layouts or when visual elements overlapped. However, its errors were typically minor and easily correctable. In many cases, simple preprocessing steps—such as enhancing contrast or isolating nodes—were sufficient to recover accurate reaction sets.

While ChatGPT offered the highest accuracy, its closed-source nature and dependence on paid cloud APIs limit its accessibility and integration flexibility. Gemini, though competitive for basic tasks, lacks the extensibility and reliability needed for more complex workflows. Mistral, on the other hand, strikes a strong balance between performance and adaptability. Its open-source availability and local deployability make it particularly well-suited for long-term research use, where reproducibility, customization, and control are essential.

8.4 Illustrative Example

To illustrate the full capabilities of our system, consider the network:



The pipeline first constructs a directed graph with nodes A, B, C, D and edges corresponding to the forward and reverse reactions. The mass-action ODEs generated for this system are:

$$\frac{dA}{dt} = -k_1AB, \quad \frac{dB}{dt} = -k_1AB, \quad \frac{dC}{dt} = k_1AB - k_2C + k_3D, \quad \frac{dD}{dt} = k_2C - k_3D.$$

The Jacobian matrix is computed symbolically as:

$$J = \begin{bmatrix} -k_1B & -k_1A & 0 & 0 \\ -k_1B & -k_1A & 0 & 0 \\ k_1B & k_1A & -k_2 & k_3 \\ 0 & 0 & k_2 & -k_3 \end{bmatrix}.$$

This output demonstrates the pipeline's ability to combine parsing, symbolic mathematics, and graph analysis in a seamless workflow.

8.5 Discussion

Our results show that combining LLMs with symbolic computation can significantly reduce the manual effort required to model biological networks. The pipeline is not only accurate but also explainable, as it provides users with both the structural and mathematical representation of the system. The introduction of multimodal analysis represents a major step forward, enabling researchers to directly convert visual diagrams into computational models without manual transcription.

The system's modular design allows for further extensions, such as parameter estimation, steady-state analysis, or integration with experimental data. While LLMs like ChatGPT excel in reasoning, the open-source nature of Mistral makes it particularly appealing for long-term and large-scale deployment scenarios. Overall, our approach provides a powerful and flexible framework for computational biology applications.

Use of AI Tools and ChatGPT

AI tools, particularly GPT-based large language models, played an integral role in multiple stages of this project. Their application extended far beyond biological diagram interpretation, encompassing development support, content generation, error diagnosis, and conceptual refinement. ChatGPT (GPT-4) was the primary assistant throughout the development process, while open models such as Mistral were used in later integration and testing phases.

From the early planning stages, ChatGPT contributed significantly to designing the overall project structure. It helped articulate the objectives of the pipeline, propose modular breakdowns, and refine the initial implementation strategy. This assistance proved especially valuable in defining the methodology for converting visual diagrams into symbolic reaction networks.

During implementation, GPT was extensively used for code generation, optimization, and debugging. It assisted in writing functions for graph construction, symbolic ODE generation using SymPy, and visualizations through Gradio. The integration logic for processing model outputs, such as handling ambiguous or noisy reactions, was also iteratively developed and refined with its help. Errors arising from symbolic differentiation, network parsing, and Jacobian computation were often debugged interactively using GPT, which expedited the resolution of non-trivial logical bugs.

Beyond code, GPT supported mathematical formalism and explanation. It aided in structuring sections involving ODE derivation, Jacobian matrices, and graph-theoretic analysis by suggesting LaTeX representations, clarifying terminology, and improving the precision of symbolic expressions. This proved especially beneficial in communicating dynamic system properties such as feedback regulation, network motifs, and oscillatory behavior.

The report itself was largely structured and iteratively refined with the help of ChatGPT. It assisted in organizing sections logically, drafting initial explanations, converting bullet points to paragraph-based academic writing, and improving the clarity of technically dense content. It also offered suggestions for harmonizing tone across chapters and ensuring coherence between methodology and evaluation sections.

In the literature review phase, ChatGPT helped explore prior work on biochemical

networks, symbolic modeling, and AI-based diagram parsing. While not a replacement for peer-reviewed sources, it was useful in summarizing key concepts, framing comparisons among different LLMs, and identifying relevant terminology for motif-based classification, feedback loops, and regulation types.

Despite its many contributions, several limitations were encountered. GPT occasionally offered syntactically correct but semantically inaccurate code suggestions, particularly in cases involving symbolic manipulation or network motifs with subtle biochemical interpretations. Some outputs lacked the specificity needed for domain-relevant problems, necessitating manual refinement or external validation. There were also instances where over-reliance on AI-generated text risked reducing academic rigor, especially when interpreting ambiguous outputs or automating figure captions.

Moreover, AI tools had limited support for interpreting figures with inconsistent notation or partial occlusion, a common feature of biological diagrams. In such cases, results had to be cross-checked against outputs from BIOCHAM or manually curated examples. When working with large symbolic expressions or equation systems, ChatGPT's memory constraints also led to simplifications that required reconstruction.

Nonetheless, the integration of GPT-based models significantly enhanced the efficiency and depth of this project. Their utility in automating repetitive tasks, guiding structural decisions, and improving the clarity of technical communication allowed for a more focused and systematic development process. With appropriate validation and critical oversight, AI-assisted workflows contributed meaningfully to both implementation and documentation, enabling a more robust and reproducible outcome.

Challenges and Problems Solved

10.1 Computational Challenges

Building a multimodal pipeline for parsing biochemical networks came with its fair share of computational challenges. Using large language models for extracting reactions from images introduced noticeable latency, mainly due to the heavy model size and reliance on external inference calls. Handling symbolic tasks like Jacobian computation for large systems also required memory-efficient techniques to avoid slowdowns. On top of that, making tools like NetworkX, SymPy, and Hugging Face work seamlessly together took careful design to prevent integration issues and unnecessary overhead.

10.2 Data Format Challenges

Rather than data scarcity, the primary difficulty lay in the heterogeneity of input formats. Network diagrams ranged from hand-drawn sketches to software-generated graphs, each introducing unique parsing challenges. While BIOCHAM outputs followed formal structure, other diagrams displayed inconsistent arrowheads, overlapping nodes, or ambiguous annotations. Addressing this required adaptable parsing routines capable of handling structural irregularities across sources.

10.3 Model Assumptions and Limitations

Each AI model integrated in the project had trade-offs. Closed-source models like ChatGPT and Gemini lacked tunability and occasionally failed on cluttered visuals. Even Mistral, despite its local deployability, struggled with atypical symbols or handwritten elements. On the symbolic side, the system assumes deterministic kinetics and mass-action laws, which overlook stochastic or spatial effects that are often critical in real-world systems.

10.4 Uncertainty in Interpretation and Parsing

Visual ambiguity remained a recurring issue, with non-standard arrows, poor alignment, and overlapping text impairing accurate parsing. These errors, if not caught early, propagated into the generated ODEs and Jacobians. Though fallback mechanisms using textual queries and manual intervention improved reliability, full automation remains an ongoing challenge, especially under noisy or low-quality inputs.

10.5 Generalization to Broader Biological Networks

Scaling to more complex systems, those featuring interlocked feedback loops, inhibitions, or multi-compartment structures, required modular and extensible design. Parsing logic had to be flexible enough to incorporate various edge semantics and regulatory topologies. While the current system handles several canonical motifs, full generalization to arbitrary biochemical models will necessitate enhanced abstraction and learning-based motif recognition.

Despite these limitations, the project established a robust proof-of-concept for integrating AI vision with symbolic modelling. The challenges encountered directly informed system design, prompting fallback strategies, modularity, and validation layers that now serve as a foundation for future expansion.

Future Work

Our current work demonstrates the feasibility of leveraging large language models (LLMs) to convert biological network images into textual reaction representations. Among the models we evaluated, the Mistral-Small-3.2-24B-Instruct-2506 (Unsloth) model [0] showed promising results due to its open-source nature and adaptability. However, this model, in its current state, is not fully optimized for the complexity and variability of biological network data.

A key future direction is to fine-tune the Mistral-Small-3.2-24B-Instruct-2506 model on curated biological datasets, enabling it to better interpret intricate network diagrams and generate highly accurate reaction schemes. Through targeted open-source contributions and domain-specific training, we aim to transform this model into a specialized tool for computational systems biology.

Another avenue of improvement involves enhancing image processing pipelines to support a wider range of input types, including hand-drawn networks, biochemical pathway diagrams, and noisy or incomplete sketches. This would allow the model to accurately parse complex networks and convert them into structured reaction mechanisms, including mass-action kinetics and Michaelis–Menten (MM) kinetics, enabling direct integration into mathematical models.

Furthermore, we plan to integrate advanced graph analytics such as stability analysis, bifurcation analysis, and sensitivity studies, providing deeper insights into the dynamical properties of biological systems. To improve usability, we will also incorporate download options for computed outputs such as ODEs, Jacobians, and reaction schemes, facilitating seamless integration with simulation platforms.

In the long term, our vision is to develop a generalized framework capable of converting any biological network image into an interpretable, executable mathematical model, serving as a powerful tool for drug discovery, synthetic biology, and systems biology research.

Conclusion

This project presents a multimodal system for the interpretation and mathematical modeling of biological reaction networks. By integrating large language models with symbolic computation, the system bridges the gap between visual and textual representations of biochemical processes. It enables the automatic parsing of network diagrams—whether hand-drawn, software-generated, or synthetically modeled—into structured reactions and subsequently into mathematical forms such as ODEs and Jacobians.

The methodology combines graph theory, image analysis, and symbolic algebra, supported by both proprietary models (ChatGPT, Gemini) and open-source frameworks (Mistral via Hugging Face). Each tool was critically evaluated for accuracy, accessibility, and adaptability within the workflow. The system proved capable of analyzing key structural properties such as node connectivity, edge directionality, and regulatory motifs, while also supporting user-driven queries through an interactive interface.

A significant contribution of this work lies in its extensibility. The modular pipeline is designed to accommodate future enhancements, including automatic detection of feedback and feedforward loops, motif classification, and integration with biological databases. The project’s use of AI was not limited to inference; tools like ChatGPT also contributed to code generation, debugging, and report structuring—thereby streamlining development across multiple stages.

Despite notable progress, several challenges persist, including visual ambiguity in network diagrams, limitations in symbolic assumptions, and the generalization of the system to more complex or stochastic models. Addressing these issues will require deeper integration of learned representations, adaptive preprocessing, and possibly training domain-specific LLMs for biological contexts.

In conclusion, this work establishes a foundational framework for AI-assisted biochemical modeling, highlighting the feasibility of multimodal pipelines for reaction network interpretation. It offers a practical step toward automated, reproducible, and interpretable systems biology workflows, with potential applications in education, research, and computational modeling.

Implementation Details

The biochemical network analysis framework was implemented in **Python** on Google Colab, leveraging its cloud-based environment for rapid prototyping and integration with AI-powered tools. BIOCHAM was also utilized for validating reaction parsing and ODE generation.

12.1 Data Handling

Biochemical reactions were provided as textual inputs or extracted from diagrams using AI-based parsing. Network data was represented as directed graphs and stored in structured formats for further analysis.

12.2 Model Construction

Custom Python scripts were developed to generate mass-action ODEs and Jacobian matrices symbolically. BIOCHAM was used for cross-validation of the reaction networks and to ensure correctness of the derived models.

12.3 Visualization

Graph structures and symbolic equations were visualized using Python plotting tools and the Gradio interface, enabling interactive exploration of network motifs and dynamics.

12.4 Code Repository

The complete implementation, including source code, datasets, and detailed instructions for reproducing the analysis, is available on GitHub: <https://github.com/ygyashgoyal/VAMP-Sys>

References

- BIOCHAM: <https://lifeware.inria.fr/biocham/>
- SymPy Documentation: <https://docs.sympy.org/>
- NetworkX Documentation: <https://networkx.org/>
- Mistral Model (Unsloth): <https://huggingface.co/unsloth/Mistral-Small-3.2-24B-Instruct-2506>
- Hugging Face: <https://huggingface.co/>
- Gradio: <https://gradio.app/>
- OpenAI ChatGPT: <https://openai.com/chatgpt>
- Gemini (Google): <https://gemini.google.com/app>
- LMArena: <https://lmarena.ai/>
- Harika, G.L., and Sriram, K. (2025). Emergent robust oscillatory dynamics in the interlocked feedback-feedforward loops. *IET Systems Biology*, 19(1), e12111. doi:10.1049/syb2.12111. PMID: 39846379; PMCID: PMC11771794.
- GitHub Links: <https://github.com/ygyashgoyal>, <https://github.com/kumardevansh>
- Google Colab: <https://colab.research.google.com/>,