

Wright State University

CORE Scholar

---

[Browse all Theses and Dissertations](#)

[Theses and Dissertations](#)

---

2022

## Deep Understanding of Technical Documents : Automated Generation of Pseudocode from Digital Diagrams & Analysis/ Synthesis of Mathematical Formulas

Nikolaos Gkorgkolis  
*Wright State University*

Follow this and additional works at: [https://corescholar.libraries.wright.edu/etd\\_all](https://corescholar.libraries.wright.edu/etd_all)



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

---

### Repository Citation

Gkorgkolis, Nikolaos, "Deep Understanding of Technical Documents : Automated Generation of Pseudocode from Digital Diagrams & Analysis/Synthesis of Mathematical Formulas" (2022). *Browse all Theses and Dissertations*. 2667.

[https://corescholar.libraries.wright.edu/etd\\_all/2667](https://corescholar.libraries.wright.edu/etd_all/2667)

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact [library-corescholar@wright.edu](mailto:library-corescholar@wright.edu).

DEEP UNDERSTANDING OF TECHNICAL DOCUMENTS: AUTOMATED  
GENERATION OF PSEUDOCODE FROM DIGITAL DIAGRAMS &  
ANALYSIS/SYNTHESIS OF MATHEMATICAL FORMULAS

A dissertation submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

by

NIKOLAOS GKORGKOLIS

Master of Computer Science and Technology, University of Patras, Greece, 2019

Bachelor of Science in Mathematics, University of Patras, Greece, 2016

2022

Wright State University

WRIGHT STATE UNIVERSITY  
GRADUATE SCHOOL

November 30, 2022

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Nikolaos Gkorgkolis ENTITLED Deep Understanding of Technical Documents: Automated Generation of Pseudocode from Digital Diagrams & Analysis/Synthesis of Mathematical Formulas BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

---

Nikolaos G. Bourbakis, Ph.D.  
Dissertation Director

---

Thomas Wischgoll, Ph.D.  
Program Director, Ph.D. in  
Computer Science and  
Engineering

---

Shu Schiller, Ph.D.  
Interim Dean of the Graduate  
School

Committee on Final Examination:

---

Nikolaos G. Bourbakis, Ph.D.

---

Soon M. Chung, Ph.D.

---

Bin Wang, Ph.D.

---

Maria Virvou, Ph.D.

---

Ioannis Hatzilygeroudis, Ph.D.

## ABSTRACT

Gkorgkolis, Nikolaos. Ph.D., Department of Computer Science and Engineering, Wright State University, 2022. *Deep understanding of technical documents: Automated generation of pseudocode from digital diagrams & Analysis/synthesis of mathematical formulas.*

The technical document is an entity that consists of several essential and interconnected parts, often referred to as modalities. Despite the extensive attention that certain parts have already received, per say the textual information, there are several aspects that severely under researched. Two such modalities are the utility of diagram images and the deep automated understanding of mathematical formulas.

Inspired by existing holistic approaches to the deep understanding of technical documents, we develop a novel formal scheme for the modelling of digital diagram images. This extends to a generative framework that allows for the creation of artificial images and their annotation. We contribute on the field with the creation of a novel synthetic dataset and its generation mechanism. We propose the conversion of the pseudocode generation problem to an image captioning task and provide a family of techniques based on adaptive image partitioning.

We address the mathematical formulas' semantic understanding by conducting an evaluating survey on the field, published in May 2021. We then propose a formal synthesis framework that utilized formula graphs as metadata, reaching for novel valuable formulas. The synthesis framework is validated by a deep geometric learning mechanism, that outsources formula data to simulate the missing a priori knowledge. We close with the proof of concept, the description of the overall pipeline and our future aims.

# Table of Contents

1. Introduction .....	1
1.1 Motivation and Outline .....	2
2. Literature Overview .....	10
2.1 Understanding of Mathematical Expressions in Technical Documents.....	11
2.1.1 Overview.....	11
2.1.2 Field Publications.....	13
2.1.3 Estimation of Maturity .....	22
2.1.4 Survey Conclusion.....	32
2.2 Pseudocode Generation from Digital Diagram Images.....	33
3. Pseudocode Generation from Digital Diagram Images .....	36
3.1 Display of the previous work.....	38
3.2 A Revisit to the Pseudocode Generation Process .....	39
3.3 Block Utility .....	41
3.3.1 Formal Representation .....	41
3.3.2 Function Knowledge Base .....	45
3.4 Dataset Generation .....	47
3.4.1 Automatic Random Generation .....	47

3.4.2 Data Augmentation .....	50
3.4.3 Graphical Representation of Diagram DataFrames.....	52
3.4.4 Generative Framework Summary .....	55
3.5 Inference through Image Captioning .....	56
3.5.1 Model Architecture .....	56
3.5.2 Results.....	57
3.6 Hierarchy of Digital Circuit Elements & Automated Pseudocode Extraction .....	58
3.6.1 Classification .....	59
3.6.2 Scanning Patterns .....	64
3.7 Conclusion .....	91
4. Analysis & Synthesis of Mathematical Formulas .....	93
4.1 Introduction.....	94
4.2 Synthesis Methodology .....	95
4.2.1 Formal Modelling of Synthesis Framework.....	95
4.2.2 Syntactic Perspective.....	98
4.2.3 Semantic Perspective .....	99

4.2.4 Synthesis Formulation .....	100
4.2.5 Indeterminacy Checks & Imposed Conditions.....	101
4.2.6 Termination Criteria .....	105
4.2.7 Space Complexity.....	106
4.2.8 Hierarchy of Calculations.....	109
4.2.9 Methodology .....	110
4.2.10 Intra-level Synthesis .....	114
4.2.11 Framework’s Proof of Concept.....	115
4.3 A-Priori-Knowledge Acquisition .....	121
4.3.1 Knowledge Outsourcing .....	121
4.3.2 Dataset.....	123
4.3.3 Differentiation from Symbolic Regression .....	124
4.3.4 Graph Representation of Formulas .....	125
4.3.5 Description of the Graph Neural Network model (Graph Prediction) .....	127
4.3.6 Synthesis Prediction on Disjoint Graphs.....	128
4.3.7 Results & Discussion .....	129
4.3.8 Self-evaluation through Validity Classification.....	131
4.4 Conclusion .....	136
5. Epilogue.....	139

Publications.....	143
References .....	144
Appendix .....	158
A.3.1 Pseudocode LaTeX Customize Templates.....	159
A.3.2 Example Internal Representations.....	160
A.3.3 Data Augmentation Methods.....	162
A.3.4 Application on Newtonian Motion.....	167
A.3.5 Application on Newtonian Motion.....	169
A.3.6 List of Composite Elements.....	169
A.3.7 Block Hierarchy.....	170
A.3.8 Detailed Grid Partitioning Example.....	170
A.3.9 Ensemble Configuration Changes .....	172
A.4.1 Application on Newtonian Motion.....	173
A.4.2 Paths to valid expressions for inner and outer vector product as binding operators. ....	181



# List of Figures

Figure 1: Submission rates in arXiv per year, arXiv submission rate statistics   arXiv e-print repository .....	3
Figure 2: Monthly submissions in arXiv (August 1st, 1991 – May 4th, 2022) Monthly Submissions (arxiv.org) .....	3
Figure 3: taken from [14], the core methodology (left) the technical document’s hierarchy of modalities (right) .....	6
Figure 4: maturity scores (left) & feature distribution (right).....	31
Figure 5: description of the methodology followed in [49].....	38
Figure 6: description of the Diagram $\rightarrow$ Function algorithm .....	40
Figure 7: step visualization of the block-to-expression conversion through SymPy.....	46
Figure 8: random generation stages .....	48
Figure 9: experimental setup (4) – Loss & Accuracy, SchemDraw, 50e .....	58
Figure 10: Example of gates-based and box-based graphical representation .....	60
Figure 11: classification label and caption sample .....	60
Figure 12: Classification train & validation accuracy .....	63
Figure 13: Image captioning train & validation monitoring.....	64
Figure 14: General scanning patterns methodology .....	64
Figure 15: Grid Image Partitioning.....	66
Figure 16: Center discovery methods .....	67
Figure 17: Hough-based random kernels algorithm .....	69
Figure 18: Application of random kernels and final partitions.....	69

Figure 19: Prediction probabilities majority vote. Classifies to full-adder and predicts caption < <i>start</i> > <i>unk3, unk4 = full_adder ( v , o )</i> < <i>end</i> > .....	70
Figure 20: Exclusion of covered centers.....	70
Figure 21: Hough-based expanding kernels algorithm.....	72
Figure 22: expanding kernels and block hierarchy .....	72
Figure 23: majority vote of most prominent partition .....	73
Figure 24: initial, valid 7 most prominent partitions .....	73
Figure 25: Genetic algorithm flowchart.....	75
Figure 26: Creation & filtering of random partitions.....	76
Figure 27: Intersection-based filtering.....	77
Figure 28: Next generation composition.....	78
Figure 29: Crossover scheme.....	79
Figure 30: Monitoring of the fittest individual per generation, in a 15-generation run....	80
Figure 31: Ensemble method partitioning.....	82
Figure 32: result after intersection filtering .....	83
Figure 33: Formal modeling and production of mathematical formulas .....	99
Figure 34: Example of minimal G-res graph .....	100
Figure 35: G-res with (right) and without (left ) intra-level synthesis.....	103
Figure 36: Pseudocode describing the synthesis methodology.....	112
Figure 37: Flowchart of the main process.....	113
Figure 38: Intra-level synthesis.....	115
Figure 39: Occurrences of F relation after substitution in the 3rd level of synthesis .....	118
Figure 40: Level-2 outcome of binding operation (*) with multiple arguments .....	118

Figure 41: Tracing graphs of complex unclassified nodes .....	120
Figure 42: Validation Scheme .....	121
Figure 43: Tree representation of $x/(y-\lambda)$ .....	126
Figure 44: Graph attribute details .....	127
Figure 45: Extraction of disjoint graph and label .....	129
Figure 46: Formulas derived from the synthesis framework & their predicted class probabilities.....	131
Figure 47: GNN train and test accuracy on the constructed dataset .....	134
Figure 48: Training, inference & self-evaluation pipeline.....	136
Figure 49: Vector space projections of mathematical tokens .....	138
Figure 50: LaTeX Custom Template (1) .....	159
Figure 51: LaTeX Custom Template (2) .....	160
Figure 52: Example Internal Representations.....	162
Figure 53: Data Augmentation Methods – Composition.....	164
Figure 54: Data Augmentation Methods – Merging.....	166
Figure 55: Data Augmentation Methods – Rotation.....	167
Figure 56: GraphViz Visualization Example.....	168
Figure 57: SchemDraw Visualization Example.....	168
Figure 58: Image Captioning model architecture .....	169
Figure 59: Grid partitioning Example.....	172
Figure 60: Level-1 byproducts.....	176
Figure 61: Confirmed semantically valid byproducts.....	177
Figure 62: Pair synthesis (1) .....	179

Figure 63: Pair synthesis (2) .....	180
Figure 64: Combined G-res .....	180
Figure 65: Synthesis Methodology Example .....	181

## List of Tables

Table 1: weights per feature.....	26
Table 2: feature values assignment.....	30
Table 3: simple diagram creation & hierarchy levels.....	45
Table 4: example of distinguishing non-labeled elements.....	54
Table 5: block class distribution.....	62
Table 6: caption-based filtering.....	76
Table 7: Number of partitions and stride size relation.....	84
Table 8: Single-entities dataset experimental results.....	85
Table 9: Results on complex samples.....	87
Table 10: Comparative experimental results.....	89
Table 11: conversion to formal representation.....	90
Table 12: Running time per method.....	90
Table 13: Example of invalidity occurrence.....	104
Table 14: Imbalanced class distribution & predictions analytics.....	130
Table 15: Confidence score through GNN logits.....	130
Table 16: Composite elements.....	170
Table 17: Composite Elements.....	170
Table 18: Ensemble Configuration Changes.....	173
Table 19: Level-0 Augmented Subjects.....	175

Table 20: Level-0 analysis of multiplication .....	176
Table 21: Pair synthesis (1).....	178
Table 22: Pair synthesis (2).....	179

# Acknowledgement

First of all, I would like to express my gratitude to my advisor, Dr. Nikolaos Bourbakis, for the opportunity he gave me through his research projects, his valuable advice and guidance throughout these past two years, and for providing his research funding and making it possible for me to pursue my graduate studies. Dr. Bourbakis literally helped me shape up my future and I am deeply grateful for that.

I would also like to thank the committee members, Dr. Chung, Dr. Wan, Dr. Virvou and Dr. Hatziligeroudis, for dedicating their time and interest in this defense. Specifically, I extend my thanks towards Dr. Hatziligeroudis, for being my Master program advisor and for cooperating with me for the past four years.

Last but not least, I am more than grateful to my family, to my partner, Marianna, and to Mike, Andrea, Irina, Andrey, Andrew and Themis for always being there, either literally or metaphorically.

# 1. Introduction



## 1.1 Motivation and Outline

Starting from two decades ago, every computational field has been overflowed with the vast development of machine learning approaches. The overcoming of persistent bottlenecks like the vanishing/exploding gradients (Hochreiter et al, 2001) [1], the increasing amount of available data, the use of high impact computational resources, like graphic card units, for the efficient parallel training of otherwise prohibitively-expensive techniques (Raina et al, 2009) [2] (Krizhevsky et al, 2012) [3], as well as the interest and demand due to the highly efficient approaches and the impressive results has established machine learning as the dominant way in numerous scientific fields, with the most tense activity observed in computer vision and natural language processing and understanding. In almost any category of sub-problems in these two enormous families of problems, with a quick overview in the relative literature one can easily spot astonishing results and substantial progress yearly, if not in shorter terms. The research activity is blooming in an incremental manner, to the point where related public searches [4] showcase a nearly exponential increase of submissions per year related to machine learning, reaching approximately 100 research papers per day submitted in 2018 just in one single, popular repository [5]. Official statistics from the same repository [6] [7] justify these claims to a greater extent. The rates involve all the categories of the arXiv repository.

Data for 1991 through 2021, updated 3 January 2022.

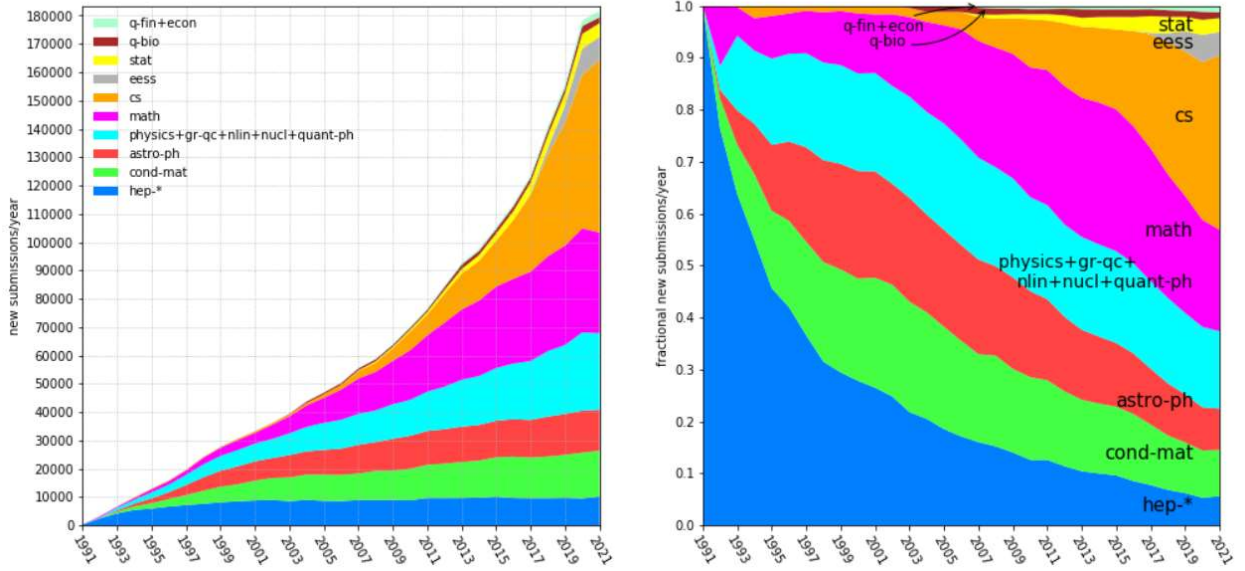


Figure 1: Submission rates in arXiv per year, arXiv submission rate statistics | arXiv e-print repository

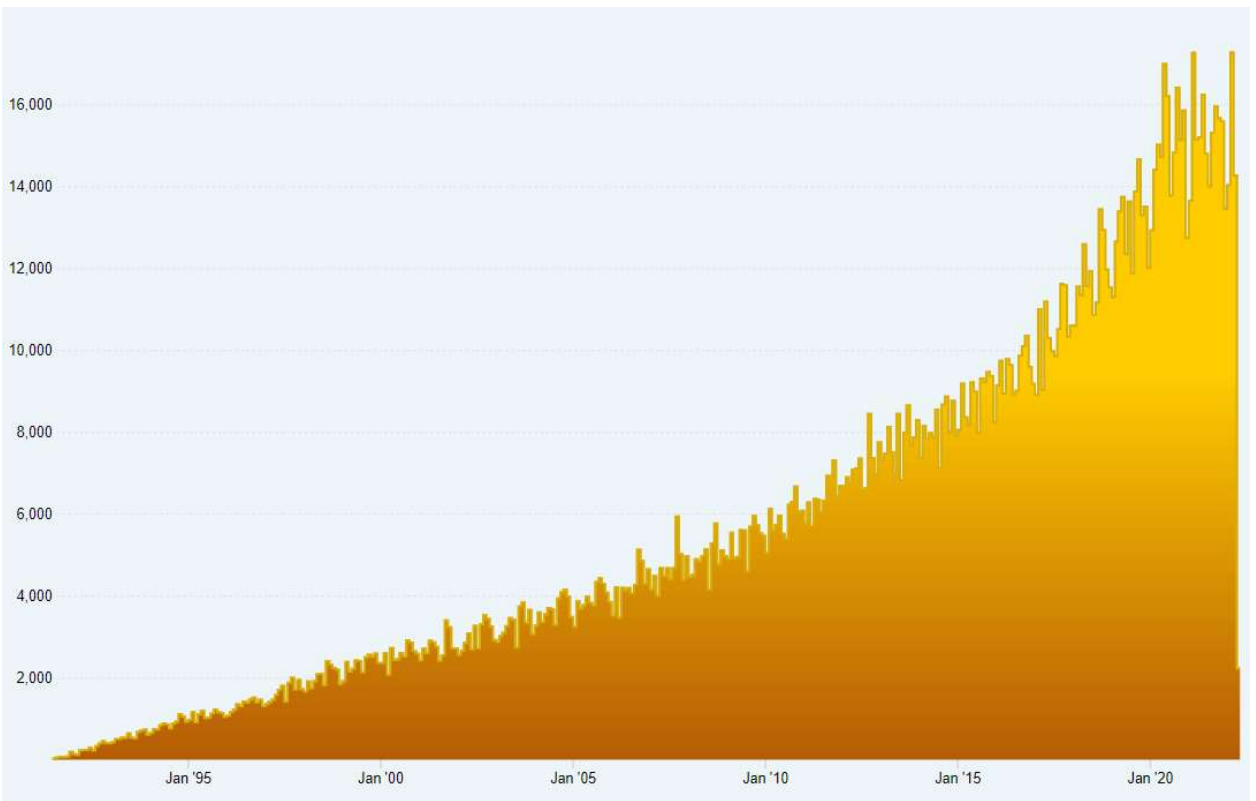


Figure 2: Monthly submissions in arXiv (August 1st, 1991 – May 4th, 2022) Monthly Submissions (arxiv.org)

This increased production of publications results in a plethora of needs that are yet to be sufficiently addressed, at least to our knowledge. For instance:

- In a publication's acceptance through the peer review method, there is an expanding disproportion between the produced publications and the available reviewers. This becomes even more profound after considering the manual labor, in terms of expertise, time and energy, that is demanded to properly review a research article.
- There is an increasing number of articles that are heavily based on past publications and increment on them. While there is still contribution, this generates redundancy, and due to the volume of articles it is hard for it to be spotted and dealt with. Subsequently, it results to noisy data that shadows other publications.

Despite these developments on the publications' late rates and the provoked concerns, the flourishing movement of machine learning has not yet shown a great interest into automatically analyzing and understanding technical documents. It would be unfair to say that there has been no interest at all, but the majority of efforts is either outdated or omits the semantic part of the analysis and focuses on quite important but also quite specific areas, e.g., optical character recognition on the article, or extraction of the articles modalities (Ahmed & Dandekar, 2015) [8], or others are restricted to a-priori knowledge dependence (Antoine et al, 1992) [9] or restrict a certain type of document (Caldas & Soibelman, 2003) [11]. There are also instances of modern publications with state-of-the-art results that also refer to the same concerns that were stated above (Aristodemou & Tietze, 2018) [12], but are mostly limited to specific tasks, like road-mapping of

Intellectual Property Analytics (IPA) in [12] and do not cover the full scope of a technical document in its general form.

A step further would be to see the technical document as a whole entity and identify the correlation between its components, often referred to as modalities. There is a limited number of publications that incorporate such a holistic approach. One of the first works spotted is the one in (Shrihari et al, 1992) [50], which address the importance of different representations of the technical document and the identification of modalities. The article then focuses on the format of the document. It is quite crucial though that the creators mention the significance of understanding a document through its seemingly independent components.

In a quite recent effort such as (Zhang et al, 2021) [10], the authors capitalize on the text, the images, and the IPA of a document by computing latent representations through well-known deep-learning architectures and subsequently fusing them to perform a multi-modal hierarchical classification. Yet the result is restricted to the classification output, therefore no additional knowledge extraction from each modality occurs. Furthermore, the incorporated textual information is limited to the title and the abstract of each document. This may help in the classification task, but discards the most crucial part of a document, that being the main content, the methodology description, and the results. The goal of a literature overview may be the actual use of a technique or a specific modification, the preprocessing and such, information which ideally should not be omitted.

Another effort of joined modalities can be seen in (Bourbakis et al, 2016) [13], where the authors make use of main textual content of the document and the images, by utilizing a more traditional approach of a Natural Language Understanding (NLU) and

Diagram Image Modelling (DIM) to convert the available information into a Generalized Stochastic Petri-Net (GSPN) representation. In this case, while a promising approach, the results are not as clearly shown, while the robustness is questionable as the authors do not incorporate into their evaluation a challenging dataset similar to [10].

The same first author proposes a complete holistic approach to the automated understanding of technical documents in (Bourbakis & Mertoguno, 2020) [14], complemented by a series of independent works, examples of which are (Bourbakis et al, 1999) [15] and (N. Bourbakis, 2017) [16]. As seen in [14], the authors suggest a synergistic analysis of the main modalities of the technical document (and occasionally their derivatives) for an automated but deep understanding of its content. They propose a synergistic approach that decomposes a technical document into crucial components, where each component is part of the abstracted properties of the document and characterizes its functionality and purpose (see Figure 3 & 4). Each component is analyzed, correlated, and modelled through an GSPN representation; the document is then reconstructed by synthesizing the GSPNs.

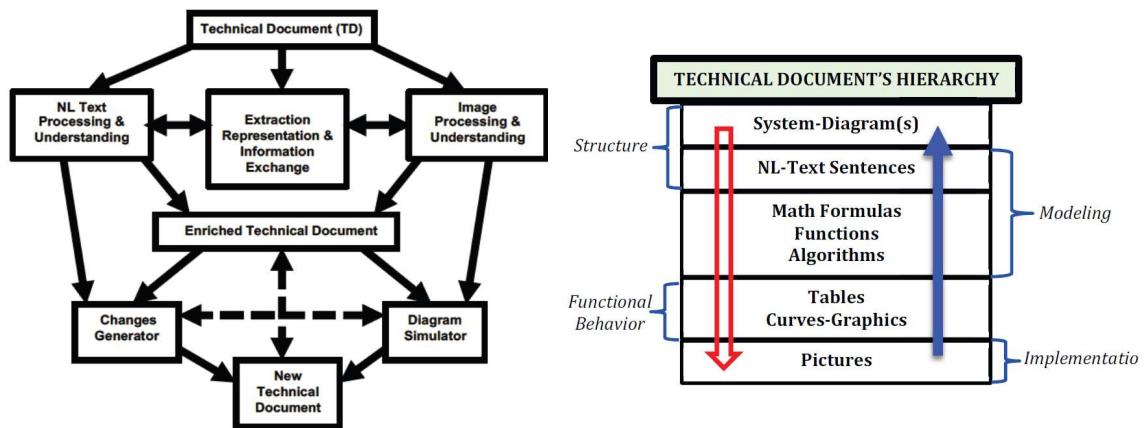


Figure 3: taken from [14], the core methodology (left) the technical document's hierarchy of modalities (right)

Our work follows on these efforts. Inspired by the aforementioned endeavors, it aims to provide a novel and efficient step further into the automated deep understanding of a technical document, by contributing to the analysis of two of its major modalities: the automated generation of pseudocode from diagram images, and the modelling of mathematical expressions.

Concerning the understanding of pseudocode generation, the task set can be loosely defined as the generation of a sequence of commands or instructions, that result in an executable description of the diagram image at hand. During this specific effort, we focus on the situation where the only available resource is the diagram image itself and no complementary elements are provided, but once an inference has been established, we suggest ways of incorporating additional info from other modalities of the document, that potentially improves the efficiency. To limit the vastness of the diagram image types, we chose to operate the on-going research on the subset of digital diagram images, mainly for two reasons: firstly, we believe they offer an appropriate benchmark in terms of complexity and variety, while secondly, their structural features allow for an approximate reconstruction of instances, which is crucial in our approach. The majority of the methodologies used involves around the idea of modelling the components of a digital diagram image through a custom-specified formal grammar and enable a controllable instantiation of an image through the production rules set. Based on the formal modelling, we build a generative framework for the stochastic creation of digital diagram images. Capitalizing once again on the formal modelling, we create 1-1 mappings from the production rules to desired annotations, which results in the creation of a labeled dataset and unlocks the potential of incorporating conventional supervised learning methods. This

concept can be applied for multiple different annotations and provide a variety of inference options.

The approach of formal modelling and instantiated samples is also followed during the analysis of mathematical expressions. We set up a context-free formal language that is able to express and create mathematical expressions. As the domain of mathematical expressions is also vast, we again limit the scope to a certain set of operations, which aims in encapsulating the domain of Newtonian motion; the subject is thoroughly discussed in Chapter IV. Our main contribution lies in the proposed synthesis framework, a methodology that treats state-of-the-art mathematical formulas as metadata and aims to discover novel and valid synthetic formulas for the interpretation of scientific subjects. We complement our framework with contemporary geometric learning models, that aim to fill the absence of a priori knowledge, by outsourcing and leveraging relevant existing information.

Through the aforementioned research efforts, we have already reached substantial results in the automated pseudocode generation, while we openly introduced the mathematical formula synthesis task and successfully achieved its proof of concept. Our aspiration is that this work will contribute into highlighting the importance and the potential of the automated technical document analysis, while hopefully inspire others to engage in these emerging topics.

The rest of the document continues on chapter 2, with an overview on the literature of the two subjects, as a detailed expansion of what has already been discussed above. Next follows chapter 3, with a detailed description of the methodologies applied on the pseudocode generation from digital diagram images. In chapter 4, we present our on-going

effort on the modelling and understanding of mathematical formulas synthesis. The dissertation ends in chapter 5 by reflecting on the presented work, our ambitions, and our aspirations for future development. Achieved publications, references and the Appendix sections follow exactly after the end of the report.



## 2. Literature Overview

## 2.1 Understanding of Mathematical Expressions in Technical Documents

We begin chapter II with a bibliographic survey of mathematical expressions in technical documents. The discussion below is derived from our survey publication in ICTAI 2021 on the same subject.

### 2.1.1 Overview

In terms of their deep, automated understanding, the mathematical formula is a modality that is surrounded by unique conditions. It can be either interpreted as textual or image data. When in an image format, the usual approach is to try to recognize the formula, separate it from its context and parse it into a string format. Hopefully, there is a lot of past and ongoing research on the subjects of recognition and retrieval, with quite successful results. The approaches used range from traditional OCR (Berman & Fateman, 1994) [18] and bottom-up (Chen et al, 2000) [19] (Toyota et al, 2006) [20] or top-down (Guo et al, 2007) [21] tree parsing techniques, to SVMs (Lin et al, 2012) [22] and advanced segmentation techniques (Nazemi et al, 2014 ) [23]. There is a plethora of high- quality articles on mathematical formulas retrieval, an active section that is still blooming (Mahdavi et al, 2019) [24] and will not get covered here. Some representative survey articles on the areas of retrieval and paring of mathematical equations in technical documents are the following: (Zanibbi & Blostein, 2012) [25] (Kostalia et al, 2020) [26].

The current work skips these steps and focuses on the deep understanding of the mathematical formulas. As deep understanding is referred the process of analyzing every possible feature of a formula, e.g., structural, syntactic, or spatial, and its correlation with

elements of other modalities, e.g., contextual text, to produce a semantic outcome for the formula itself. This level of understanding on the mathematical formulas of a technical document could complement in several ways the technical document analysis. To name a few, an obvious use is the archiving and the matching of documents with similar mathematical equation. Another application is in the identification of sub-formulas and the immediate labeling, which could contribute both in educational purposes as a helping utility, and in research for the creation of structured labeled datasets for the use of supervised techniques. Compared to the retrieval and parsing sections, the literature found on understanding of mathematical formulas is quite more restricted. Thankfully, the radical accomplishments of the past decade have ignited the interest on the subject, by offering new approaches.

For the sake of this study, approximately (30) papers on the subject of mathematical formulas understanding were investigated. The approaches and the techniques used varied greatly per article, while the vast majority had other primary goals. To achieve a certain level of coherence, we focus on a subset of (9) representative articles, each of them utilizing a different approach on the subject of mathematical formulas understanding. In the next sections, the papers are presented in a timeline fashion, focusing on their philosophy and the techniques that they incorporate. Then, we try to evaluate the level of completeness each work offers on the understanding of the formulas. To achieve that, we define an approximation scheme based on custom features, which function as criteria for the completeness of the approach. Finally, a comparison of the works at hand is present and the study concludes with a discussion on future ideas.

## 2.1.2 Field Publications

By searching the relative literature, various articles were chosen as a representative sample of the work on the understanding of mathematical formulas in technical documents. A realization made after researching the literature is that the approaches used vary greatly and the main goals of each work differ, with the articles end up having only a few aspects in common. Based on this observation, we chose to organize the presentation of the related work based on the approach used to deal with the mathematical formulas, as any other matching between them appears impractical. Although overlapping and correlations occur frequently, the most distinguished paths followed are the approaches based formal modeling and theory provers, those that utilize contextual information and the those based on artificial neural models. Later on, they are also organized as neural and non-neural approaches. The selection of articles is based on the novelty of the introduced approach and the correlation it may have with other articles, without considering any of the included works lesser or greater than others.

### A. Formal Modeling & Theory Provers

Probably one of the most interesting approaches met, yet not focused on our main subject, is the one used by Kaliszyk, Urban and Viskocil in (Kaliszyk et al, 2017) [27]. Based on a series of works (Kaliszyk et al) [28][29][30][31][32], on the same or related projects, the authors aim in automating the formalization of mathematical formulas, contributing greatly to the effort on machine understanding of mathematical formulas. Their approach is based on utilizing the HOL Light theorem assistant system as an evaluation scheme and formalized data provided by the Flyspeck Project (Hales et al, 2017)

[33][34], to develop automated techniques that aim at the inform-to-formal translation task, as described in [27][31][32]. The main techniques used are:

- (1) modeling of the parse trees through a context-free formal grammar,
- (2) a probabilistic context-free grammar parser (PCFG), specifically a modified version of the CYK bottom-up parsing technique, for the parsing of formal/informal sentences from the preferred formulas corpus,
- (3) semantic checks as a workaround to ambiguity,
- (4) the use of probabilities of deeper subtrees and their incorporation into the grammar as production rules to overcome limitations of the context-free grammar modeling, and finally,
- (5) the use of discrimination and AVL trees for the efficient implementation of their system.

The article concludes with its main contribution, which is the introduction of an efficient context-based learning and parsing through the aforementioned techniques, as well as a major improvement (64%) in the informal-to-formal translation task. Although the goal of the specific translation task differs from the overall understanding of the mathematical formulas in technical documents, and despite the article residing on the domain of theorem provers, it still constitutes a great example on how to approximate formulas understanding, thanks to its well-explained ideas and the smart utilization of the introduced techniques. As authors point out in the introduction, their hope is that by gradually improving the computer understanding on how humans manipulate the mathematical vocabulary to express semantics, could benefit the NLP field.

## B. Formulas Context

Another approach on the understanding of mathematical formulas, found in several articles across the literature, is the exploitation of the surrounding textual (or other) context of the formulas to help extract semantic features.

A representative example of such approaches, with a novel and interesting application, can be found in (Jiang et al, 2018) [35], an article that introduces a math assistant for PDF reading for educational purposes, mainly aiming in assisting graduate students in understanding mathematical expressions found in technical documents. Initially, the authors showcase the need of special handling of mathematical formulas, by highlighting the differences between natural language text and mathematical expressions, referring to the ambiguity of symbols, the recursiveness and the special structure / formatting (specifically stating LaTeX, MathML). Another contribution of the article is that they state the hierarchical nature of mathematics and the deep, gradual interconnection of aspects that translate into sub-formulas of an original, complex formula.

In an educational fashion, the authors address the issues stated above through a two-scope solution:

(1) Offline, they introduce a Formula Evolution Map (FEM), that is based on a knowledgebase of more than 4 million documents of associated articles and OER in general associated to the formulas and aims at encapsulating the evolutionary information the formulas include, over time. The implementation of FEM in essence is a weighted directed graph  $G = (F, R, \tau)$ , where  $F$  is the formula

vertex set and  $R = F \times F$  is the directed evolution relation set. Wikipedia constitutes the main source of the knowledgebase. The graph is determined using two procedures, described in detail in the article:

a. Formula Evolution Relation Generation

b. Formula Evolution Direction Determination

Both of the above procedures are mainly based on the context of the formulas found in each Wikipedia page. All the formulas encountered are initially in a LaTeX format, then converted into Presentation MathML and parsed by a semantic tree-based approach, presented in (Lin et al, 2014) [36]. Terms are then extracted in a hierarchical manner, through a proposed algorithm. Details on the extraction algorithm, as well as the rest of the procedures used, can be found in [35].

(2) Online, they associate the context and the layout of the formulas with FEM, in order to solve the Mathematical Content Understanding (MCU) problem at hand through appropriate suggestions of explanatory articles.

Together with the above, they introduce a novel reading environment (PRMA), which consists of a PDF Reader with a Math-Assistant, something that resides outside of our scope.

As before, although the primary focus of the paper is not on the understanding of the formulas itself, several novelties and methodologies are introduced, that contribute to the progress of automated understanding.

Another representative work on the understanding of mathematical expressions based on their context is found in (Grigore et al, 2009) [37]. The main goal of the article is the assignment of a concept to a given mathematical formula, while focusing on dealing with the disambiguation of the symbols or terms found in the formulas, a common problem in the mathematical notation, based on the surrounding context. The authors focus on disambiguating a certain subset of expressions that have the following linguistic format:

*noun phrase*  $\rightarrow$  *symbolic math expression*

, where they use the offered context on the left side of the formula for the disambiguation.

To achieve the disambiguation, the work leverages several Term Clusters, which are created based on a large, annotated corpora provided by Open Math Content Dictionaries (J. Davenport, 2000) [38]. Using co-occurrence statistics on the Term clusters and the textual context of the formula, they are able to assign a properly established concept to the formula at hand. Three similarity metrics were investigated:

(1) Dice,

(2) Pointwise Mutual Information (PMI),

(3) z-score,

in accordance with custom scoring functions and other. Details can be found in [37].

The experiments are performed based on two custom baselines, one being the random concept assignment and the other the use of limited contextual information, The



results are quite promising, outperforming the baselines by far with all three similarity metrics, on Recall, Precision, F0.5 and Mean Reciprocal Rank.

### **C. Hybrid & Neural Models**

There are numerous works that make use of the relatively recent advances in machine learning and are based on the use of neural or hybrid models. A representative example of such approaches is a series of articles from Minh-Quoc Nghiem et al (Nghiem et al, 2012) [39] (Nghiem et al, 2013) [40]. As the title of [37] denotes, the authors model the problem of automated understanding of mathematical formulas as the translation from Presentation MathML Markups to Content MathML Markups. Math ML (Ausbrooks et al, 2014) [41] stands for Mathematical Markup Language and constitutes one of the main ways of transcribing a mathematical formula in a digital document in the web. In short, the Presentation MathML Markup represents the notational structure of a formula, while the Content MathML Markup represents the mathematical structure of a formula. Considering the benefits of the Markup representation and the wealth of relative data on the web, the authors in [39] try to approach formula understanding by building a successful, generalizing mapping function from Presentation to Content Markup.

To overcome the limitations of past, rule-based systems, they make use of statistical machine translation models, for the task of translating from Presentation Markup to Content Markup. The translation rules are automatically derived using a separate heuristic rule set, defined as Fragment Rules, that break the original Presentation parsed tree into sub-trees. Each small subtree constitutes a sub-expression, which is subsequently translated into Content MathML Markup format, by using an enhanced translation rule set. More details are offered in [39]. The evaluation of the system is performed on datasets taken

from the Wolfram Function Site, with the results being quite satisfactory. The authors contribute further by defining a custom metric Tree Edit Distance Rate (TEDR) for their specific task, which is a combination of Tree Edit Distance and Translation Error Rate.

As a continuation of the work presented in [39], the authors develop a new hybrid model on the same task [40]. The new model incorporates an extra processing step, in between the procedures of statistical-based rule extraction and translation. The new addition contains a specific Support Vector Machine model that deals with the disambiguation of identifiers, by assigning to potentially ambiguous terms their correct content.

The training and test datasets for the SVM disambiguation models both consists of several aligned and ambiguous Presentation and Content MathML Markup pairs. The features provided to the model are derived from both the formula Presentation Markup and the surrounding text, which characterizes the overall model as a hybrid approach. The experimental results improve the overall score on the initial dataset, while on a newly introduced corpus, the new approach unexpectedly fails to provide improvements. The authors justify the result to the insufficient amount of surrounding text.

One of the most recent works on the subject of mathematical formulas understanding, following a very interesting and unique approach, is the one found in (Saxton et al, 2019) [17]. The article parleys on the ability of neural models to capture and represent mathematical reasoning as humans do. It begins with a discussion on the unique characteristics of mathematical reasoning, how we perceive it, and the challenges this brings for neural models and other machine learning techniques.

Based on these observations, the authors proceed in developing a dataset as part of an evaluation scheme on the machine understanding of the aspects of mathematical reasoning. The dataset consists of school-level mathematics problems of different type (i.e., Algebra, Calculus, Probability etc.) and their compositions, following a sequence-to-sequence string format of question and answers. It encapsulates the hierarchy and interconnections that characterize mathematics, the need for generalization and problem-solving, while it still possess all the aforementioned characteristics of mathematical formulas understanding, e.g., ambiguation. The dataset is also complemented by both interpolation and extrapolation test sets, that contribute to the determination of the reasoning abilities of the neural models. It is synthetically generated, instead of crowd-sourced, as it offers modular structure benefits. The authors genuinely provide profound reasons on choosing freeform text format instead of a parsed tree or graph structures, as, in between others, it is a more flexible and expressive format that can generalize in more question types, more natural and closer to real mathematical formulations, while parsing is controversial. This modular structure of the freeform sequence-to-sequence formatting also allows for compositionality of questions and enrichment of the dataset. Another notable characteristic is the question generation procedure: they are generated per category in a reverse engineering approach, when first the answer is sampled, then the question is constructed. Per category, the dataset offers  $2 \times 10^5$  training instances and  $10^6$  testing instances.

Each utilized model is evaluated by comparing the model answer with the label answer. For each character same as the original answer, the model is either awarded 1, otherwise 0. The performance metric is then determined by the average for each category.

The models that are evaluated are neural nets, as they are by construction the less biased or domain specific. Two different architectures are addressed: LSTM Neural Networks (Hochreiter & Schmidhuber, 1997) [42] with attention and Attention/Transformer Networks (Vaswani et al, 2017) [43]. The experimental results are more than promising, with the Attention-Transformer model to achieve 0.76 interpolation accuracy and 0.5 extrapolation accuracy. The authors then proceed in offering an extensive presentation of the experimental setup, the established baselines, and the reasoning evaluation method. Overall, it is an innovative work that contributes greatly to the understanding of mathematical formulas and opens the path for extensive research on similar approaches.

Finally, in (Allamanis et al, 2017) [44] Allamanis et al present another breakthrough work on the continuous representation of mathematical expressions. They introduce a newly created neural architecture called Neural Equivalence Networks (EQNET), which models both the syntactic and semantic properties of an expression, and result in its enriched continuous representation. Together with the EQNETs, they also propose a new training method called subexpression autoencoding, which correctly clusters the equivalence classes. Their work is partially based on the TreeNN architecture or recursive neural networks, proposed in a series of articles by Socher et al (Socher et al, 2012-2013) [45][46]. During the experiments phase, they evaluate their novel model on two datasets, one on Boolean expressions and another one on polynomial expressions. More details on the datasets as well as the techniques themselves and the evaluation scheme, can be found in [45]. In the end, the authors manage to show that EQNETS outperform every previous approach (e.g., TreeNN, tf-idf, RNN) on well-established

semantic calculations task, noting at the end that variable-sized representations could prove beneficial.

Many other works can be found in the literature, that approach the understanding of mathematical formulas through neural models. Several of them are either similar to the presented articles, or predecessors, so there they are omitted. As a last, notable reference we state the work of Zaremba & Sutskever in (Zaremba & Sutskever, 2015) [48], that train an LSTM network to map small code programs to their output, and (Zaremba et al, 2014) [47] by Zaremba et al, that build a neural model to discover mathematical identities given mathematical expressions by building symbolic trees and developing search strategies through machine learning approaches.

### **2.1.3 Estimation of Maturity**

At the previous section we presented a representative sample of our inspection of the scientific literature on the subject of mathematical formulas understanding. In this section, we try to take this discussion a step further, by introducing a set of features that evaluate different aspects of

understanding of mathematical formulas. We focus our evaluation setup in capturing the desired characteristics that we believe an approach on mathematical formulas understanding should have.

We also want to make clear that with this effort, the goal is not to compare different works from one another. This would be inaccurate in various levels, as the presented articles may deal with the subject of understanding, but each one follows a different philosophy with a different end goal. On the contrary, we regard all the pre-referenced

publications as equal, each one of them contributing to the ultimate goal of mathematical formulas understanding in each own, unique way. Our objective through the following evaluation scheme is to assess every presented work on the ends that should be met by an ideal approach on mathematical formulas understanding. Therefore, each approach is evaluated based on the “maturity” it exhibits towards the understanding, independently from one another. With that being said, we proceed to the presentation of the evaluation scheme.

### **A. Features**

We express the aforementioned “maturity” by calculating an overall score for each work, which is factored by a set of custom features. Through these features, we try to represent the desired characteristics that should be existent in an approach on mathematical formulas understanding. They are introduced below:

- Awareness ( $f_1$ ): a metric that tries to encapsulate the sphericity of the sources used to understand or interpret a mathematical formula, e.g., context, or external databases.
- Adaptiveness ( $f_2$ ): measures the potential of the approach to be applied on a different setup, or how broad its current setup is.
- Accuracy ( $f_3$ ): states the efficiency of the subject model on the given task, as reported by the authors of the article.
- Robustness ( $f_4$ ): evaluates the performance of the application when applied in a different setup but similar to the initial one.

- Scalability ( $f_5$ ): determines the ability of the approach to scale on a dataset of different size, while still remaining efficient and robust.
- Complexity ( $f_6$ ): refers to potential restrictions to the approach's use or evolution, due to complicated structure.
- Novelty ( $f_7$ ): an estimation on the degree of novelty the current work expresses, in relation to other works in the field.
- Association ( $f_8$ ): how relative the end goal of the article is to the subject of mathematical formulas' understanding.

We incorporate awareness to estimate the roundness of the means an approach uses to understand a mathematical formula. This can refer to the context of the formula that is taken under consideration, the format of the elements that are considered, or the introduction of external material, e.g., relative labeled datasets. Typically, more high-quality, relative elements contribute to a deeper understanding of the formula. We also consider adaptiveness, as to complement works that are developed independently of any specific dataset or setup, rendering them more adaptive. Accuracy is as well included, as stated in the experimental section of each article. As usual, robustness refers to the performance of the approach on a new test dataset which is similar to the training set. When provided, we evaluate robustness by comparing the training and testing performances. If such info is not provided, we are unable to self-perform such evaluations, so we base our outcomes on associating the detailed description of the articles and the estimated adaptiveness. Scalability is also an important aspect in an approach, so we take it into consideration based on the described techniques used and the insight that is provided. With

complexity we refer to the structure of the approach and how it may affect its compatibility, or potential future improvements. Being a negative aspect, low complexity score implies high unwanted complexity in the approach. Obviously, we could not omit novelty, as it determines the unique contributions and introduction of new approaches in terms of mathematical formulas understanding, opening new paths to explore. Finally, we also incorporate the association of each work with respect to the understanding of formulas, as we have already state that the main subject of each work varies.

### B. Score

After estimating the aforementioned features' values for each work, we continue with the calculation of the maturity score. We follow an approach similar to [26], where each feature is multiplied by a certain weight. We then use the weighted average of these values to assign a unique score to each work, which expresses its estimated level of maturity towards the understanding of mathematical formulas. Let  $f_i$  denote the feature value assigned to an article, here  $i \in F$ , and  $F = \{1, 2, 3, 4, 5, 6, 7, 8\}$  is an index set corresponding to the set of the six pre-referred features. Each  $f_i$  is multiplied by a specifically determined weight  $w_i$ , depending on the importance of the feature to the maturity calculation. As our task is an abstract estimation, the weights are not subject to any fine-tuning procedure, but are subjectively chosen, based on our beliefs and reflections, as stated in this article.

	<b>Value</b>	<b>Normalized</b>	<b>Feature</b>
<b><math>w_1</math></b>	1.2	0.21	Awareness
<b><math>w_2</math></b>	0.9	0.16	Adaptiveness



$w_3$	0.7	0.12	Accuracy
$w_4$	0.5	0.07	Robustness
$w_5$	0.3	0.05	Scalability
$w_6$	0.3	0.05	Complexity
$w_7$	0.8	0.14	Novelty
$w_8$	1.1	0.19	Association

Table 1: weights per feature

With  $m_k$  we denote the maturity score per article, where  $k \in K$  and  $K$  is an index set corresponding to the list of articles. As we already stated, it is specified by the weighted average of features:

$$m_k = \frac{\sum_{i \in F} w_i f_i}{\sum_{i \in F} w_i} \quad (2.1)$$

The main difference between [26] and our implementation resides in omitting the user’s perspective, focusing only on the understanding of the formulas. We believe that currently, the understanding of mathematical formulas is not in a tool state (e.g., parsing), but rather in a developing phase, therefore ease of use or user friendly nature should not be taken under consideration. Additionally, we restrict the features around the aspect of understanding and performance, aiming for a purer estimation of maturity towards the subject.

### C. Maturity evaluation

Based on the aforementioned and the authors’ detailed description of each approach, we try to evaluate in what degree each work satisfies each feature. The articles

that we include are the ones that were briefly analyzed in section II. The assignments overall can be seen in *Table 2.2*.

In [27] the authors incorporate 22,000 informal/formal pairs of Flyspeck theorems. While it is a large number of instances for a training dataset, it is restricted to HOL Light material and informal-to-formal parsing, without generalizing to the regular form of a technical document's mathematical formula. This also determines the adaptiveness and the association of the work. The context aware approach increases both the awareness and robustness of their previous efforts, while offering novelty. Scalability is not clear but is assumed according to the tree-based and formal grammar-based techniques used. The structure of work is also quite complex. By observing the accuracy on all the ATP experiment variations, we mainly lean on the Theorem metric. Considering the nature of the problem, the results are satisfying.

In [35] we can see a greater adaptiveness and awareness, as the work deals with regular mathematical content and its context. Scalability and robustness are assigned according to the described techniques, although they are not clear due to high complexity. The end goal of the work also differs from understanding mathematical formulas, having a more educational purpose, therefore association is not at its higher. Complexity is estimated to be reasonable, offering utility and space for improvement. As for the accuracy, while depended on human feedback, it achieves satisfying results. Novelty is quite present, especially through the introduction of concept hierarchy and the Formula Evolution Map approach.

In [37], the authors are focused on the disambiguation of mathematical formulas, that offers decent association to the topic of understanding, but do not expand any further.

The dataset used is comprised of various technical documents, which, in combination with the created TCNs, contribute to an ideal setup. The strategy of simply using the left-part context seems restricting. In terms of scalability, co-occurrence term frequencies and minor textual processes seem promising. The complexity of the structure is acceptable. Accuracy for  $\delta=0.9$  is more than satisfying. Robustness is apparent, especially for POS-PMI metric.

The authors of [39] incorporate statistical machine translation, and a MathML dataset from the Wolfram Function Site, with the goal of formula semantic enrichment. Its awareness and adaptivity are relatively limited, due to the restricted setup. Robustness and scalability are both benefited by the machine translation approach. Accuracy is satisfying, at least compared to the presented standard SnuggleTeX. Complexity seems to be acceptable. The TEDR metric in combination with the machine translation approach offers novelty elements. The goal is clearly associated to some extent with the topic of understanding.

The same authors as in [39] offer an improved version of their work in [40]. As described briefly in chapter II and thoroughly in [40], a new processing step based on SVMs is incorporated. Accuracy (in most cases), scalability and robustness are benefited. Novelty is now increased significantly, and the same awareness and adaptivity limitations remain. Association is increased, as disambiguation is now included. This addition is seamlessly performed, without increasing the complexity.

Maybe the most innovative approach can be found in the recent work of [17]. The authors contribute with a new philosophy of approaching mathematical content, by creating a uniquely structured dataset of mathematical problems and evaluating advanced neural language models on the task of recreating the answer. The models learn to distinguish and

classify between different types of problems based on simple, textual format. Awareness and adaptivity are high, as the work includes a well-defined dataset generation mechanism that can be openly modified. Association is debatable, as understanding is eminent but due to neural networks nature it does not provide insights and understanding's metadata on a certain formula. Scalability is ensured due to the techniques used. Accuracy, especially for the transformer's interpolation, is fulfilling, while robustness as extrapolation falls of in general. Due to the nature of the problem evaluations are promising but not competitive. Surprisingly, complexity is not a restrictive factor, as the dataset generation is extremely well defined, and the neural techniques used are standard.

In [44] a new graph neural networks' architecture and a corresponding learning paradigm are presented. Novelty contributions are again eminent. Awareness is also high, due to the approach used. Adaptivity is restricted by the fact the authors use custom datasets with Boolean and polynomial cases. In association, we again encounter the black-box nature of neural systems, but it is simultaneously favored by the algebraic features of the produced embeddings. For scalability, we follow the same as in [17]. Both accuracy and robustness, are impressive, compared both to baselines like tf-idf and more complex approaches. Complexity is acceptable, as the approach poses a complicated structure but is well defined and leaves space for improvements.

In [47] have one of the first approaches to successfully incorporate neural techniques on specific mathematical symbolic structures. Both novelty and association are up. Awareness is remarkable, as the formal presentation of the formula used incorporates all the elements. But it ignores general context. We consider adaptivity limited, as the datasets are specifically structured, address solely polynomials and the created grammar is

hard-coded. Same elements characterize its complexity. Both accuracy and robustness are impressive. Scalability is not clear.

We conclude with [48], a work that could be considered as a predecessor of [17]. Novelty is up due to its application and the incorporation of curriculum learning, although sequential analysis and reconstruction of an answer through a neural language model can be found. Association is low, as the work does not have as its goal the understanding of formulas, nor it provides relative metadata. On the other hand, adaptivity is promising. Accuracy and robustness are humble but promising, while scalability is high, due to the RNNs used. Awareness is also satisfying, as it incorporates a whole corpus of coding, with every element included. There no major complexity issues.

The feature assignments can be seen in *Table 2.2*. Based on these, we calculate the overall maturity score for each article, according to our described evaluation scheme, as seen below. Finally, we provide an illustration of the maturity scores in Graph 2.3.

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$	$f_8$	$m_k$
[10]	6	4	6	7	6	5	7	6	5.93
[18]	8	8	7	5	6	8	8	7	7.44
[20]	8	7	8	6	8	8	7	7	7.26
[22]	6	5	6	7	7	7	7	7	6.40
[23]	6	5	8	8	8	7	4	8	6.21
[25]	9	9	8	6	9	9	9	6	8.02
[28]	8	7	9	9	9	7	8	9	8.00
[31]	7	6	9	9	7	6	9	8	7.35
[32]	7	9	7	7	9	9	7	4	6.65

*Table 2: feature values assignment*

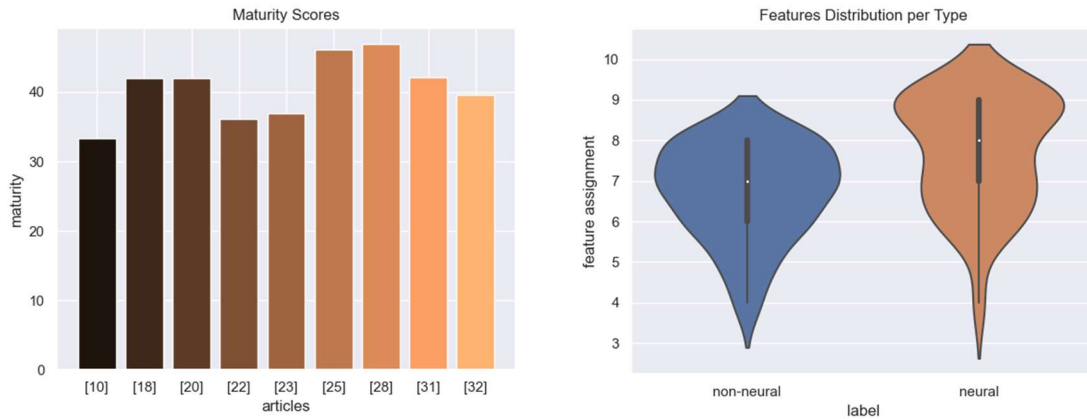


Figure 4: maturity scores (left) & feature distribution (right)

## D. Discussion

In the evaluation scheme we set, the approaches could be distinguished in two major categories: those that are based on tree parsing techniques and utilization of context and formal modelling, and those that depend on neural approaches. By examining the feature values assigned on each category (*Table 3.2*), it can be observed that per category a slightly different pattern is followed.

The approaches based on neural models tend to receive a more varied set of feature values, with more lows and highs. On the other hand, the non-neural approaches stick to a gaussian-like distribution, with a mean close to 7.00. Although our sample space is limited, we believe that this behavior holds in general. The main reason for this is that neural models tend to deal with a single, generic type of data format (e.g., textual) and demand more volumes of data to successfully find potential relations. This provides them with more awareness and adaptivity, and the recent advances on the field result in breakthrough scalability and robustness. For the same reason, they instantly lose some awareness as the incorporation of external data, or data of different format, requires hybrid approaches,

where the advantage of robustness and scalability is questioned. Their inability of providing understanding's interpretation and the lack of insights makes a poor combination for our main purpose, therefore association is usually lower. In the meantime, non-neural approaches, while less aware or robust, are able to provide more understandings metadata, like the hierarchy of formula terms or a formal modeling scheme, which makes them more relevant. As a result, the average case ends up receiving a less varied distribution of feature values.

### **2.1.4 Survey Conclusion**

In this sub-chapter, we investigated all the broad scientific literature on the topic of mathematical formulas understanding, as they are found in technical documents. Based on the point they were introduced and the novelty of the approach they are introducing, we selected a representative sample of articles and analyzed their philosophy, structure, and contribution in section II. In section III, we created an evaluation scheme, where for each selected article a maturity score was estimated, based on a custom set of features that according to our beliefs should be potent in an approach on mathematical formulas understanding. We conclude by detecting the major differences between the selected approaches and interpreting our results. We hope that the work above will prove capable of acting as a starting point of future efforts in the investigating and expansion of the field of mathematical formulas understanding.

## 2.2 Pseudocode Generation from Digital Diagram

### Images

In contrast to the thorough examination of the literature on the subject of mathematical expressions' understanding, we limit the search on the pseudocode generation only on the last four years past, meaning the timeframe from 2018 to May 2022. We do so, as the work we focus on is a revisiting of a previous work of the C.A.R.T. lab, seen at (Rematska et al, 2018) [49], which also dealt with the same subject, therefore the related work has already been addressed. Due to the narrowness of the subject, the following citations are not necessarily tied to the exact task, but rather provide ideas and methodologies that could inspire an adaptation on pseudocode generation from diagram images.

Although it does not address the pseudocode generation process from a technical document, a quite interesting work can be seen in (Luo et al, 2021) [51]. The authors propose a hybrid model architecture on the task of chart understanding, combining reasoning solutions for the extraction of the key-points and the chart type, and a deep neural model (hourglass) for the inference. The article gives insight on the challenges that occur in analyzing chart images with a neural model, due to the various types of charts and the different reasoning that stands per type. Additionally, it showcases the importance of combining rule-based systems with contemporary learning models, to overcome several bottlenecks, generalization, or efficiency, that may occur for both sides. They also contribute with a novel benchmark dataset on Excel chart images. The pursue of the publication at hand for data extraction may differ from our end call, but the problem it



presents still possesses several similarities with the automated pseudocode generation from diagram images.

Another recent work on the subject of pseudocode generation is presented by Sunitha E. V. and Philip Samuel in (S. E. V. and P. Samuel, 2019) [52]. In this article, the generation aims to actual source code from UML charts, by utilizing a rule-based approach with involves the use of state machines. As an evolution from previous similar approaches, the authors introduce a design pattern-based state machine that supports concurrent and history states, resulting in promising results. The idea of the implementation is quite inspirational. Yet the scope is quite limited for the purpose of technical document's understanding, due to the fact that UML charts are provided only with a restricted number of publications, as opposed to digital diagram images or diagram images in general.

As it was stated above, our stronger inspiration comes from the work presented in [49]. The authors address the exact same task that we set during the introduction (see chapter 1), by proposing a methodology for the extraction of pseudocode of a certain diagram, using a DIM analysis on the diagram and complementing the outcome with the surrounding text document, which is also analyzed by a set of NLU actions. Each outcome is represented through a common formal grammar, specifically designed for the task. The synthesis of the corresponding formal words is mapped to a Stochastic Petr-Net, which is in turn mapped to the pseudocode. While the methodology looks quite promising, the end results exhibit a lack of robustness and validation in general. The authors mainly state the approach as an idea, present a couple of representative examples, but do not continue to a solid evaluation on an extensive dataset. Additionally, the implementation is heavily depended on a static knowledge base that is responsible for the interpretation of each

identified block through either the diagram or the corresponding text. This also poses a flaw, as it renders the generalization of the approach to different types difficult. It is also worth to mention that contextual information is not always available.

Below, we analyze how we adapt the modelling idea to a different setup, which allows for robustness and clarity. We also introduce a mechanism for the creation of an annotated dataset, that enables the incorporation of established supervised learning methods and can contribute also to the evaluation of the overall approach.

### 3. Pseudocode Generation from Digital Diagram Images

The technical document is an entity that consists of several essential and interconnected parts. These parts, often called modalities, allow us the ability to process them in depth separately and derive crucial technical and semantic information for the document itself, as well as for other related modalities. Despite the extensive attention that certain parts have already received, per say the textual information, there are several aspects that rarely get referenced throughout the bibliography. The diagram images found in technical documents constitute such a modality. It can provide crucial information about the functionality and the technical details of both the document and related modalities. Driven by the above, and the previous efforts [49] that were discussed during the scientific literature chapter, in this section we limit our scope to the subset of digital diagram images, to focus on the task of automated extraction of the pseudocode describing the functionality of a given diagram. We propose a complete methodology of modelling based on formal representation, introduce three new novel annotated datasets, suggest several predictive setups based on mapping the initial problem to an image captioning task, analyze the inference capabilities, and discuss the future potential of the approach. The rest of the chapter is structured in seven sub-sections, that discuss in detail the continuation of the work and the results that have been achieved until now. In 3.1, we refer to the proposed methodology of the main previous effort. We continue with our differentiation and the main methodology. In 3.3 we present a detailed description of a novel formal modelling scheme. In 3.4 we create the internal representations of the diagram's elements and a generative framework for their manipulation. In 3.5, we introduce a novel dataset of pseudocode-annotated images, incorporate an image captioning pipeline, and discuss in depth our progress. In 3.6, we propose a novel partitioning methodology as an alternative

to the obstacles found in 3.5, introducing another two novel annotated datasets. We conclude in 3.7 with a summary of our progress, our concerns and future steps.

### 3.1 Display of the previous work

The previous works on the pseudocode extraction process from digital diagrams was thoroughly presented during section 2. It was observed that the pseudocode extraction process of [49] is closely related to our effort as we begin with the same setup. We are providing its followed philosophy and main methodology, in order to showcase its workflow and of course highlight our motives for differentiation.

The main idea introduced in [49] is to act based on a given diagram and its natural language context. Using a sequential analysis of the two inputs, the process was able to extract the main meaning of each block in the diagram, complement it through the corresponding text and then use formal modelling to structure the pseudocode and produce it. The process that was followed is loosely depicted in the next flow chart:

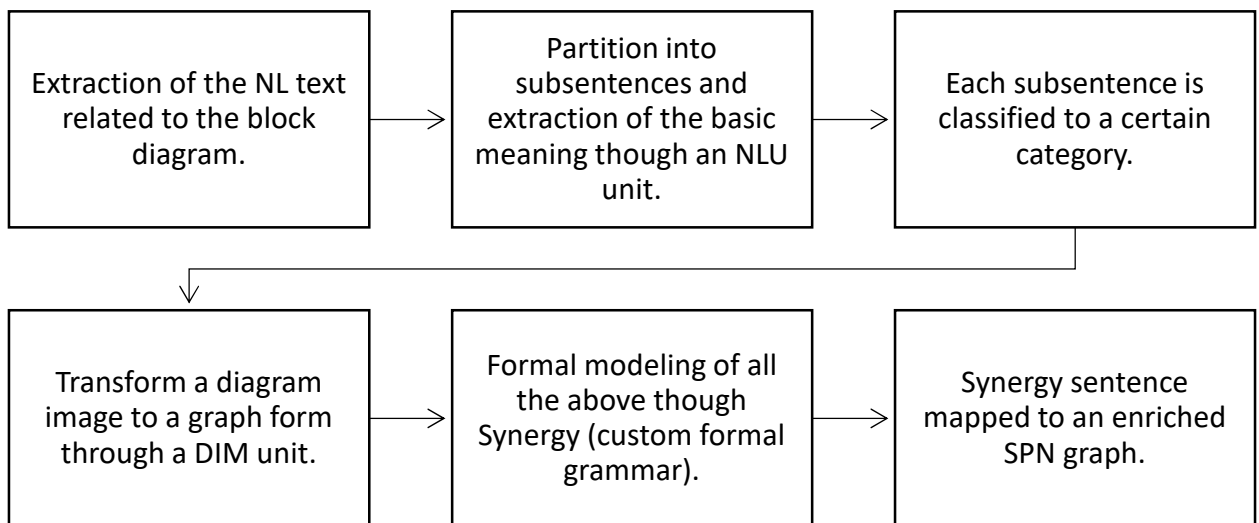


Figure 5: description of the methodology followed in [49]

The end product is a text part of pseudocode, similar to the one provided below, that corresponds to the accompanying diagram. The functionality of each element of the diagram is derived from an external static knowledge base. The relation between the static knowledge base and the diagram is created based on the extracted keywords and its textual context.

## **3.2 A Revisit to the Pseudocode Generation Process**

The main drawback we encountered with the past approach is the difficulty of generalizing. The information extraction process, concerning both the diagram and the textual content, is heavily depended on the static knowledge base, covering only a limited number of specific types and instances, while being hard-coded and difficult to enrich. In response to that, we are now recreating the information extraction steps, making them broader and less dependent on the static knowledge base. Although we have come to consider that it is unlikely to create a knowledge base – free approach, we believe we can render the knowledge base more dynamic and prone to change, by focusing more on the data at hand, remodeling the representation of the provided elements, and incorporating learning techniques, amongst others.

Specifically, during the presented methodology, we ignored the textual information and focus solely on the diagram. We also modified the structural steps of the pseudocode. Briefly describing, each diagram block is translated into a well-documented function in the pseudocode. These functions, their inputs and outputs and the flow of the diagram help then define the functionality of the whole diagram through a main function. The steps

followed, as well as an example of the current results on the previous sample diagram are provided below:

---

**Algorithm** *Diagram*  $\rightarrow$  *Function*

---

**Input:** Blocks' DataFrame

**Output:** The diagram's pseudocode

```

1: for each block  $\in$  Blocks do
2:   preprocessing()
3:   create a Record(name, inputs, outputs, connections)*
4:   define block's pseudocode:
5:     DEFINE  $\rightarrow$  (nameb)
6:     INPUTS  $\rightarrow$  (inputsb)
7:     MAIN  $\rightarrow$  (functionalityb)**
8:     OUTPUTS  $\rightarrow$  (outputsb)
9:   end define
10: end for
11: define diagram's pseudocode:
12:   DEFINE  $\rightarrow$  (named)
13:   INPUTS  $\rightarrow$  ( $I - I \cap O$ )***
14:   MAIN  $\rightarrow$  (Call of block functions)****
15:   OUTPUTS  $\rightarrow$  ( $I - I \cap O$ )***
16: end define

```

---

Figure 6: description of the Diagram  $\rightarrow$ Function algorithm

\* : Record is a struct created to monitor each block's features

\*\* : functionality derived from the dynamically defined knowledge base

\*\*\*:  $I$ ,  $O$  the overall inputs and outputs

\*\*\*\*: calling order is considered by the interconnections

For representation purposes only, we also allow for the modelling of the semantic results of the above steps through custom LaTeX (Leslie Lamport, 1984) [56] templates. The first template builds separate algorithms for the main and the complementary functions. The complementary functions correspond to each individual block, while the main function, as it is customary, aims to describe the overall functionality of the diagram. A second template is also developed, that builds subroutines for each individual block,

while being confined to a single algorithm. Both of the templates can be seen in the Appendix, section A.3.1.

## **3.3 Block Utility**

Next, we address the specification of each function's (i.e., block's) semantic description, where we are experimenting with vectorization of the diagram's elements, forms of formal representation, and the gradual incorporation of the textual information.

### **3.3.1 Formal Representation**

We model the semantics of the elements, the blocks, and the overall diagram through a dynamic formal language, derived subsequently from a formal grammar. The main intuition of this choice is the strong mathematical foundations it sets; formal modellings offer structural support, while an appropriate set of production rules allows for the dynamic manipulation of the formal tokens. This approach helps in countering problematic circumstances, like ambiguity, as well as managing metadata.

We create the formal grammar, by adopting a one-to-one, bidirectional mapping between a diagram's elements and the terminal symbols of the formal grammar. The elements of the diagram are categorized to inputs, outputs, and block names. The set of production rules is then applied on the non-terminal symbols to create words in the formal language, i.e., words that consist of terminal symbols. Each word corresponds to the description of a certain block in the diagram.

The non-terminal set consists of any string that is encapsulated in between capital "N" symbols. It turns out quite convenient to represent such a set using a regular



expression, as it allows for a minimal but precise form. The definition of the set can be seen in (3.3.1). In response, the terminal set incorporates every string that does not belong to the non-terminal set, which is also expressed as a regular expression in (3.3.2). The starting symbol is set to be as "NSN", as seen in (3.3.3), which obviously belongs to  $N$ .

The final set  $P$  in (3.3.4) consists of four basic production rules that manipulate the terminal symbols to create the formal description of a block.  $P_{block}$  (3.3.5) works as a template, setting the structure of the block's formal word and positioning accordingly non-terminal symbols for the block's name, inputs, description, and outputs. Specifically, it incorporates the format "*name:IN(inputs)|DS(desc)|OUT(outputs)*", where *name, inputs, outputs* are non-terminal elements, that are consecutively mapped to the pre-defined terminal symbols through the trivial production rule  $P_{set}$ . The latter, found in (3.3.6), simply replaces a non-terminal symbol with a terminal one.

Likewise,  $P_{initU}$  (3.3.8) is used to initialize a temporary dynamic set as  $U = \{ (, ), |, :, IN, DS, OUT \}$ , which includes the keywords and punctuation used by  $P_{block}$ . Followingly, every terminal symbol created is registered in  $U$ , with every element of  $U$  mapped to a certain element of the diagram. In essence,  $U$  results in a subset of  $T$  and corresponds to the vocabulary of the formal language that models the diagram.

Finally,  $P_{compose}$  (3.3.7) connects two given strings with a right-oriented arrow and is used to interconnect diagrams. The connecting configuration followed is discussed further in the data augmentation chapter.

**Definition 3.1.** *The pseudocode's framework formal grammar is defined as:*

$$G = \{N, T, S, P\}$$

where:

$$N = \{x \mid x \in \text{re}(. * N . N . *)\} \quad (3.3.1)$$

$$T = \{x \mid x \in \text{re}((? ! . * N . N . *) . *)\} \quad (3.3.2)$$

$$S = \{NSN\} \quad (3.3.3)$$

$$P = \{P_{\text{block}}, P_{\text{set}}, P_{\text{comp}}, P_{\text{initU}}\} \quad (3.3.4)$$

$$P_{\text{block}}: NSN \leftarrow NnN:IN(NiN)|DS(NdN)|OUT(NoN), \quad (3.3.5)$$

$$P_{\text{set}}: y \leftarrow x, \quad (3.3.6)$$

$$P_{\text{compose}}: word_x \rightarrow word_y, \quad (3.3.7)$$

$$P_{\text{initU}}: U \leftarrow \{ (, ), :, |, IN, DS, OUT \} \quad (3.3.8)$$

■

We provide an example of creation an ADD block with two input elements  $\{X_1, X_2\}$  and an output element  $SM$ :

$$\text{ADD: IN}(X1, X2)|\text{DS}(\text{None})|\text{OUT}(SM) \quad (3.3.9)$$

$$U = \{ :, |, (, ), IN, DS, OUT, ADD, X1, X2, SM, MUL \} \quad (3.3.10)$$

Inductively, a collection of the occurring words form a sentence in the language that corresponds to the description of the diagram. We define the syntax of the language, equivalent to the order of the words in the sentence, by considering the intersections and disjunctive unions of the inputs and outputs of each block, as well as the overall inputs,

outputs, interconnections, and inner byproducts of the diagram. It naturally occurs for certain blocks to be independent of byproducts of other blocks in the diagram, therefore these blocks are classified as independent, and their functions are called first in the main method that recreates the diagram. This methodology results in the formation of a block hierarchy, with the group of these blocks forming its first level. Any block that receives as input the output produced by a first level block belongs to the second level of hierarchy. Subsequently, the rest of the levels are created. The hierarchy is completed by the blocks whose outputs contain at least one element that is not connected to any other block.

As an example, in the following simple setup can be found three given blocks: ADD, SUB and MUL. The first two are independent of any inner byproduct, therefore they are considered to be in the bottom of the constructed hierarchy and can be called first, regardless their order. MUL is connected to both of the former through their outputs, therefore MUL constitutes the second layer of the hierarchy. Lastly, MUL is also the only block whose output is not connected to any other block, therefore MUL constitutes the final output layer of the diagram.

The application of the production rule  $P_{comp}$  is:

$$\begin{aligned} \text{ADD: IN}(X1, X2)|\text{DS}(\text{None})|\text{OUT}(\text{SM}) \\ \rightarrow \text{MUL: IN}(\text{SM}, \text{SB})|\text{DS}(\text{None})|\text{OUT}(\text{M}) \end{aligned} \quad (3.3.11)$$

$$\begin{aligned} \text{SUB: IN}(X3, X4)|\text{DS}(\text{None})|\text{OUT}(\text{SB}) \\ \rightarrow \text{MUL: IN}(\text{SM}, \text{SB})|\text{DS}(\text{None})|\text{OUT}(\text{M}) \end{aligned} \quad (3.3.12)$$

It can also follow the notation:

$$\begin{aligned}
 & \left[ \begin{array}{l} \text{ADD: IN}(X1, X2) | \text{DS}(\text{None}) | \text{OUT}(\text{SM}), \\ \text{SUB: IN}(X3, X4) | \text{DS}(\text{None}) | \text{OUT}(\text{SB}) \end{array} \right] \\
 & \rightarrow \text{MUL: IN}(\text{SM}, \text{SB}) | \text{DS}(\text{None}) | \text{OUT}(\text{M})
 \end{aligned}
 \tag{3.3.13}$$

The above formal sentence represents the call order of each block’s subroutine, so that the initial diagram is recreated by the pseudocode. The approach generalizes well to other inputs, while it also comes with the structural benefits of the formal representation. In the next chapter, we make use of the latter to define the blocks’ description.

<i>Level 1</i>	<i>Level 2</i>
ADD SUB	MUL

*Table 3: simple diagram creation & hierarchy levels*

### 3.3.2 Function Knowledge Base

We exploit this formal setup in order to address the major missing component of a block’s representation, which is the description of their functionality. The core idea of the approach lies in the repetition and commonality of the blocks found in digital diagrams, and the available knowledge on their functionality. Ideally, we desire to specify each block through symbolic mathematical expressions, which will comprise the body of the block’s subroutine. To access such information, we explore available sets of open-source libraries related to mathematical operations and digital circuits under our framework, searching for any correlation with our existing examples. We mainly consider SymPy (Meurer et al., 2017) [53], a Python module for symbolic calculations of mathematical expressions that resembles the serviceability of Computer Algebra Systems (CAS), like Mathematica

(Wolfram-Research, 2019) [54] or Maple (MathWorks, 2019) [55]. The followed process is described below.

Firstly, each element of the core directory of the module is compared with the name and number of arguments of the subject block, derived from the formal representation. In case of a perfect match, inputs of the block get converted into the symbolic representation of SymPy and are given as arguments to the matched function. The resulting expression is supplied to the pseudocode as the subroutine's core command. If a perfect match is not found, the Levenshtein distance metric is used to find sufficiently similar names of functions, filtered by a high metric threshold, to minimize false relations. The distance metric is expressed in the interval [0,1] for clarity, while the threshold currently used is 0.9. A step by step visual example can be seen underneath:

<b>Formal Representation</b>	<b>Formal Name &amp; Arguments</b>
ADD: IN(X1, X2) DS(None) OUT(SM)	<b>ADD</b> , {X1, X2}, {SM}
<b>SymPy Function &amp; Arguments</b>	<b>Expression</b>
sympy.core.add. <b>Add</b> , *	SM = X1 + X2

*Figure 7: step visualization of the block-to-expression conversion through SymPy*

While the task is mainly resolved through outsourcing, we also incorporate a complementary, manually created static knowledge base, to address cases that elude the outsourced knowledge bases – e.g., division, subtraction, etc., SymPy's documentation offers an article dedicated to the adopted notation system.

## 3.4 Dataset Generation

We now utilize the benefits of formal modelling, in an effort to create a set of randomly constructed diagrams. Such a source of samples, produced in a completely unbiased and stochastic manner, constitutes the main means of evaluating the robustness of the presented pseudocode extraction methodology.

### 3.4.1 Automatic Random Generation

Inspired by the formal grammar that structures the representation, and its resemblance to natural language, we approach the sample generation similarly to the regular text structure. Part of this has already been described during the definition of the formal language. The terminal symbols of the grammar correspond to the letters of a natural language like English, in the sense that they are used to form words that grammatically stand in the language. The words are formed using the two presented production rules,  $P_{block}$  and  $P_{set}$ .

An addition to the formal grammar is the introduction of a new production rule,  $P_{func}$ . Its domain is restricted to the available functions, while it returns an adapted version of the block template – see  $P_{block}$  in (3.3.5) – where the length of inputs and outputs is set to the number of inputs and outputs of the respective function, using non-terminal symbols as placeholders.

The non-terminal symbols that result from the application of rule  $P_{func}$  are then set to terminal symbols through  $P_{set}$  to produce the formal word. This discussion indicates a priority of rule application for the successful creation of a random formal word describing a block:

$$P_{block} \rightarrow P_{func} \rightarrow P_{set}$$

or: (3.4.1)

$$P_{set}(P_{func}(P_{block}(S)))$$

, which summarizes the steps followed to randomly generate blocks.

At this point, a random sampling is performed in the uniformly distributed [0,1] space for each fixed input of the created block. If the sampling exceeds a certain threshold, then a recursion call is performed to expand the existing diagram by creating a new block, interconnected with the existing one. The connection is achieved by randomly setting one of the outputs of the recursive call as the fixed input of the primary block under inspection. The initial block's name is also retained. This results in a top-down construction of the diagram. A detailed example of the random generation is analyzed underneath:

#### Random Generation Stages – Block 1

Template	NoVN: IN(NRWN) DS(NgIN) OUT(NUVN)
Intermediate stage	INV: IN(NdqN) DS(negates a single input) OUT(NhHN)
Terminal stage	INV: IN(tu) DS(negates a single input) OUT(hq)

#### Random Generation Stages – Block 2

Template	NkIN: IN(NnwN) DS(NaDN) OUT(NclN)
Intermediate stage	AND: IN(NrUN, NNN) DS(an AND clause of inputs) OUT(NcYN)
Terminal stage	AND: IN(jo, er) DS(an AND clause of inputs) OUT(tu)

Figure 8: random generation stages

The tables above correspond to the outputs of the random generation sample call. The process begins with the assumption of the starting symbol, “NSN”. The first stage involves the application of the  $P_{block}$  production rule on the starting symbol, which produces the template of the block, as discussed previously. During the intermediate stage,

the template becomes subject to the  $P_{func}$  rule, where the name of the block is set, while the non-terminal symbols corresponding to the inputs and outputs of the block are substituted by other terminal symbols, now corresponding to the number of inputs and outputs of the knowledge base function. In the case of Block 1, this translates to the INV function, with three inputs and one output. At the terminal stage, the  $P_{set}$  rule is utilized to randomly replace the remaining non-terminal symbols with terminal ones, thus producing the formal word:

$$\text{INV: IN(tu)|DS(negates a single input)|OUT(hq)} \quad (3.4.2)$$

As a last step of the process, a random sampling from a uniform distribution [0,1] is compared to the defined threshold, for each input of the INV block. In our example, the threshold is exceeded for the input (tu), therefore a recursive call is performed on the random generation process, now also providing as additional arguments the input of the initial call and the block's name, (tu, INV) in our case. The input variable (tu) is used as one of the outputs of the new block, while the block name is used for the connection between the two blocks, and it is the only information not incorporated into the formal representation. Should the outputs be more than one, the selection between them is random and equiprobable. To avoid disambiguation, the names of blocks and inputs are maintained and omitted during the random sampling. Aside the predefined single output and the monitoring of variable names, the rest of the process stays the same. In the case of Block 2, the output is:

$$\text{AND: IN(jo, er)|DS(an AND clause of inputs)|OUT(tu)} \quad (3.4.3)$$





The recursion call is optional and can be omitted in case a single block output is desired. In the same spirit, it is possible to increase the spanning potential of the process by simply lowering the threshold. The threshold value used for this example, as well as the default option, is 0.8, as it was observed that it results in controllable results in average.

The inclusion of the described random generation process renders the presented pseudocode extraction framework available of producing unbiased and independent diagram examples. Repeated calls of random generation with varying recursion thresholds result in a set of diagrams of diverse complexity and elements. Extended examples can be seen in the Appendix, section A.3.2.

The main benefit of this effort lies in the compatibility of the samples with the pseudocode extraction process in (3.3). Although a seemingly blunt and plain dataset of diagrams, without any deeper insight, the produced formal token set can become subject to the proposed framework, and therefore get converted to a labelled dataset of images and pseudocode as text description. This creates the opportunity to utilize state-of-the-art classification methods and other learning techniques that can boost and improve the introduced until now rule-based approach of pseudocode extraction.

## **3.4.2 Data Augmentation**

### **3.4.2.1 Diagram Composition**

The quality of annotated datasets is heavily considered based on its size, diversity, and objectiveness, in between other metrics. In an effort to enhance the variety of diagram creation in our own framework, we add to the top-down approach the ability to compose

two separate diagrams into one. In a nutshell, the composition is performed by distinguishing two given diagrams as inbound and recipient. The connection between the two distinguished diagrams is achieved by selecting an output element of the last level in the block hierarchy of the inbound diagram, as well as an input element of the first level in the block hierarchy of the recipient diagram. Both selections are random and unbiased. An intersection check is also performed to find common elements. In case of any finding, the recipient's elements are appropriately changed to avoid any ambiguity. The two diagrams are then connected through an edge between the selected elements.

The connection is performed through the intermediate formal representation of the diagrams. Both the diagram image (see 3.4.2) and corresponding pseudocode are automatically reproduced for the composed outcome. Annotated examples for the three augmentation techniques used can be seen in Appendix, section A.3.3.

### **3.4.2.2 Diagram Merging**

In the same spirit as in composition, we incorporate the option of merging two separate diagrams, but without building any connection between them. Here, the only processing required is the intersection check between the two diagrams' sets of elements.

### **3.4.2.3 Diagram Rotation**

Inspired by the data augmentation techniques used in image processing, we also apply rotation of various degrees to randomly selected diagram images, while keeping the pseudocode annotation unchanged. We chose to apply rotation in a way similar to how the mutation operation is performed in genetic algorithms, being an unbiased and stochastic approach, proven to be efficient. Each of the generated diagram images is subject to a

uniform sampling in  $[0,1]$ . If the sample exceeds a certain rate threshold, the image is rotated to a randomly chosen angle in between  $[0, 360]$  degrees; otherwise, the diagram image stays intact. The rotation rate threshold used here is 0.1.

### **3.4.3 Graphical Representation of Diagram DataFrames**

The following section presents a deterministic, automatic approach on mapping any given DataFrame of a diagram to its actual graphical representation. It involves around the fact that each element in the formal representation has a 1-1 correspondence with the elements of the to-be-produced graph. Therefore, the elements are identified accordingly as blocks and edges, then are utilized by a custom tailored mechanism based on GraphViz (Ellson et al, 2003) [57], a popular diagram creation framework, to create the graphical representation. Complementary modification and enrichment takes place, so as to make this utilization possible. More specifically:

1. The blocks and edges included in the formal representation are identified and stored separately.
2. For each element, basic graphic objects (GraphViz – .dot format) are created. Blocks are represented by an elliptic shape, while edges through a single, usually labelled, graph edge.
3. Inductively, the edges of the graph are constructed based on the entries of initial data. To create a clear 1-1 correspondence, it was found that the entries should be clustered into two categories. The first corresponds to the entries whose input simultaneously belongs to the outputs collection, have an existing connection, and produce an existing output (according to the format the initial data are following these usually are the first entries for each block). For each of such

entries, a tuple  $(start, end, label)$  is created that defines the graphical representation of the edge, where  $start$  corresponds to the entry's block,  $end$  to the entry's connection and  $label$  to the entry's output. The entries without outputs are covered inductively. Essentially, the created tuples aim to create the internal edges of the diagram, i.e., the interconnections between blocks.

4. The second cluster aims at creating the external edges of the diagram. It addresses the entries who satisfy at least one of the following conditions: either the entry's block is not included in the overall connections, or the entry's input is not included in the overall outputs. Here, tuples follow the same format, but  $start$  corresponds to the entry's input,  $end$  to the entry's block and  $label$  to the entry's input again. This process helps define the GraphViz arguments of each edge, as they vary depending on the category each edge falls into.

5. As a final processing step, we distinguish the elements of the diagram which belong exclusively to either the overall input or the overall outputs, in respect to the two sets. This is done for representation purposes only, so as the far left or right terminals of the external edges are not represented as elliptic nodes.

6. As described in the first step, each of these tuples, together with the blocks, instantiate GraphViz objects. Distinguished elements are represented by null-shaped nodes, while blocks with elliptic shapes. Edges with missing values in the  $start$  and  $end$  features are omitted.

<b>Edge Tuple</b> <i>(start, end, label)</i>	<b>Clusters</b>
('ADD','MUL','SM')	1 <sup>st</sup>
('b','COPY-2', 'b')	2 <sup>nd</sup>

Table 4: example of distinguishing non-labeled elements

We summarize the visualization process through the following algorithm:

---

**Algorithm 2. Graph Visualization**

---

```
1: define formal representation


---


2: extract blocks
3: extract edges
4: for each block in blocks do:
5:     create block dot object
6: for each block in blocksdot do:
7:     if (block  $\in$  inputs) and (block  $\in$  outputs) and (blockc  $\neq$   $\emptyset$ ) do:
8:         Connect(block, blockc)
9:     if (blocki  $\notin$  connections) or (blockc  $\notin$  outputs) do:
10:        Connect(blocki, block)
11: return visualization
```

■

The presented graphical representation function successfully produces the diagram image for each randomly generated formal word or sentence. For a more descriptive and accurate representation of the digital gates, the identical methodology is followed using as backend the framework *SchemDraw* (Collin J. Delker, 2022) [6], a Python-built module provided by C. J. Delker. Both options were widely considered during experimentation, with similar results. Detailed examples of both can be found in the Appendix, section A.3.4.

### 3.4.4 Generative Framework Summary

Combining the data augmentation techniques with the diagram random generation methodology described above, we end up with a novel and spherical framework for the generation of annotated datasets on the task of automated pseudocode extraction process of digital diagram images. Based the details of the presented techniques, the generation framework is characterized by the following hyperparameters:

- recursion threshold, with a default value of 0.7
- maximum number of recursion occurrences, to control the span of the diagrams, with a default value of 10
- terminal symbol subset  $T_{random}$ , defaulting to the union of Latin lowercase letters and their *length-of-two* combinations
- the size of the overall diagrams (in all formats) produced
- the number of composed diagrams, in proportion to the overall size, defaulting to 13.5%
- the number of merged diagrams, in proportion to the overall size, defaulting to 13.5%
- the rotation rate, i.e., the chance of an image to be rotated, defaulting to 0.1
- the stochastic-ness and degrees of rotation domain, defaulting to uniform sampling in  $[0,360]$

Finally, although restricted to digital diagrams of logical gates for the time being, we want to note the ability of the framework to extend to other types of diagrams by two

diodes: either enriching and expanding the self-defined knowledge base, or incorporating external mathematical systems, like SymPy, that include the desired information on block functions.

## 3.5 Inference through Image Captioning

Due to the formal modelling that took place in (3.3), for each generated block, and subsequently a diagram image, we already have a pseudocode describing it. Therefore, we can now convert our initial problem from a deterministic production of the pseudocode of a diagram to an image captioning task. Below, we analyze the two main branches of our effort to produce inference through image captioning and the results achieved to date.

### 3.5.1 Model Architecture

As of November 2022, the established approach on image captioning considers attention-based encoder-decoder models [43] (Xu et al, 2015)[59]. As this is not the main concept of the presented work, we will not provide a detailed survey of the works in the field but will suffice to state that other architectures with the CNN-LSTM (Soh, Moses, 2016)[60] were also considered but, in our case, the results were more satisfying with the encoder-decoder in [59].

In a nutshell, the model consists of two parts: the encoder and the decoder. The encoder is usually a deep convolutional layer that is trained in an image classification task. By omitting its inference layer, usually a Softmax classification layer, we are able to receive the latent representation of an image through its last hidden layer. This latent representation can then be used as the vectorized input for the decoder part, which in the case of image captioning is an attention-based classifier that generates words according to

an already provided (and usually preprocessed) vocabulary. Seen as a trend since 2018, it is considered more efficient to use publicly available pre-trained convolutional models as the encoder, whose weights during the training of the encoder-decoder model remain constant (frozen). This means that only the decoder is trained on the provided training dataset. In our current effort, we are using a pretrained *EfficientNetB0* (Tan & Le, 2020)[62] model as the encoder, accessed through TensorFlow [61] archives. Following the established procedures in attention-based Transformers, the CNN Encoder is succeeded by an attention-based Encoder with one multi-head attention layer and a Layer Normalization (Ba et al, 2016)[63]. Finally, we also incorporate the positional embeddings as they are required for capturing positional information during attention. The architecture of the model used, as well as its configuration, is provided in the Appendix, A.3.5.

### 3.5.2 Results

In this section we present the details of the image captioning dataset. We investigated different combinations of sizes of the dataset with unique formats of the annotations and ended up using a dataset of approximately 30k images. Each image is encoded in dimensions of (299,299,3). We also tried both visualizing frameworks (see 3.4.3); they exhibit minimal differences, with SchemDraw 's specific gate representation benefiting the model by approximately 0.2 better accuracy score. The annotation format of each image is also simplified extensively; we use a one-line format, where every command of the original pseudocode is separated by a semi-column, similar to the C / C++ paradigm. The training-evaluation split of the dataset is also kept constant to an 80/20 ratio. Finally, we also incorporate typical image augmentation techniques: horizontal flipping, rotation by 0.2 (fraction of  $2\pi$ ), contrasting by 0.3 (i.e., chosen randomly in range of [1.0 –



0.3, 1.0 + 0.3]). Training is carried out for 50 epochs. Figure 9 contain the loss & accuracy monitoring graphs.

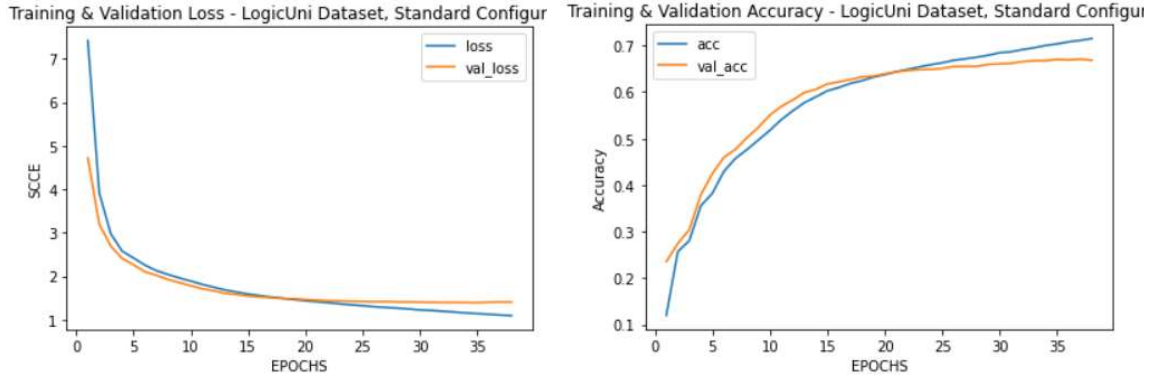


Figure 9: experimental setup (4) – Loss & Accuracy, SchemDraw, 50e

### 3.6 Hierarchy of Digital Circuit Elements & Automated Pseudocode Extraction

In the previous chapters we presented in detail the formal modelling of the digital diagram images, the creation of a complete, generative framework for the generation of annotated datasets of the digital diagram images, as well as the mapping of the pseudocode extraction task to an image captioning problem, accompanied by a detailed analysis of the models used, the setup and the improvement steps taken. During that effort, the achieved accuracy was rooted around seventy percent, with the main identified cause being the static dimensions of the pre-processed images in the encoder step causing a resolution drop for complex diagram of three or more blocks. In response to this, we propose a different methodology, based on a family of scanning techniques that isolate different partitions of the original image and thus overcome the resolution drop. The main parts of the methodology are the following:

i. The construction of a classification dataset with images of *single* elements frequently found digital diagram images, and a classification model that infer on such images.

ii. The adaptation of both the image captioning dataset and model on an extended collection of single elements.

iii. The incorporation of three different scanning techniques for the encapsulation of digital diagram elements and the reconstruction of pseudocode after the application of the two supervised learning models on the created partitions.

In the next sub-chapters, we analyze in depth the three main parts in the same order and discuss the suggested workflow.

## **3.6.1 Classification**

### **3.6.1.1 Modifications**

The classification setup is based on the generative framework that was introduced during the previous chapters, with several significant modifications. We deactivate the possibility of recursive element addition, to restrict each image to one single element. Additionally, we extend the list of supported elements in the framework with (18) composite elements frequently found in digital diagram images, to enrich the framework's inclusiveness. The comprehensive list of these elements can be found in the Appendix, section A.3.6.

These composite elements are incorporated both in their complex form of composed digital gates, as well as labeled simple box blocks. The latter representation is also utilized for the seven basic gates.

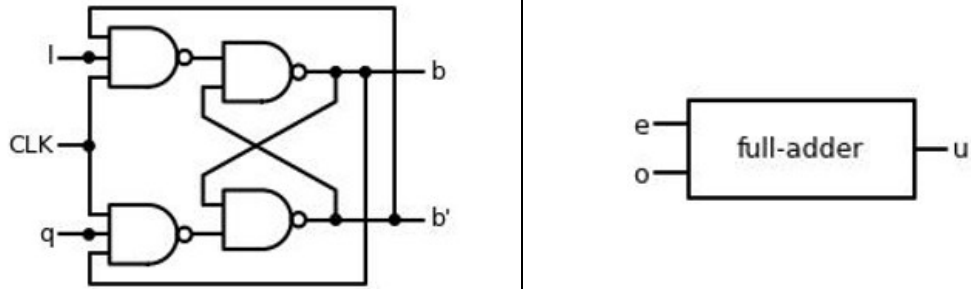


Figure 10: Example of gates-based and box-based graphical representation

To allow for a classification task, we also create, in parallel to the pseudocode captioning, a classification label for each image. This is derived with ease from the formal language representation that was previously introduced in 3.2. For composite elements of the same family, the specific type of the element is omitted, and the general family of elements is used. E.g., a *Master – Slave Flip – Flop* and a *D – type Flip – Flop* will be both labeled and classified as plain *Flip – Flop*.

Lastly, to emulate the conditions during the scanning of a complex image, the ability of depicting an element without its input(s) and output(s) labels is also introduced, which is used in a stochastic manner during generation. In that case, the generated pseudocode caption uses incremental  $ukn_{\{i\}}$  elements to represent these attributes.

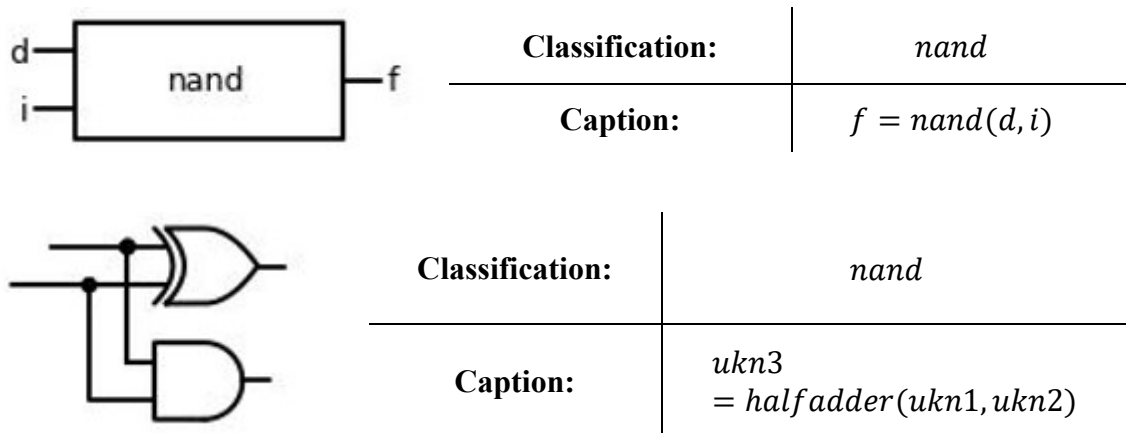


Figure 11: classification label and caption sample

### 3.6.1.2 False Class

Ideally, the classification task should also be able to perceive when an image correctly depicts an element and when it does not, to provide the means for an appropriate partition of a complex image into. One workaround would be to use the predicted probabilities per class for the Softmax layer of our classification model (described below), but this would require the specification of a threshold for the accepted predictions and further fine-tuning. Instead, we choose to introduce noise into the dataset by arbitrarily and independently cropping the four sides of the generated images by a certain percentage. Images that are cropped by a percentage of 10% to fifty 40% of the original dimension, are classified to a distinct False class. This introduction allows the model to identify and accept partitions including complete elements, and in the meantime, decline partitions that cut off essential segments. The lower threshold of 10% cropping results from the default behavior of the visual frameworks used ((Ellson et al, 2003) [57], (Collin J. Delker, 2022) [58]), as for the smaller percentage no essential information is removed. The upper threshold of 40% is set to avoid dimensional indeterminacy.

The chances for a False class instance during the generative process is equivalent to each other class, following a uniform distribution. The overall size of the dataset rises to 7000 samples, following a (70% training – 10% evaluation – 20% testing) split. The percentage of each class in each split of the actual dataset can be seen in Figure 3.44 below.

	Training – Samples	Training – Percentage	Validation – Samples	Validation – Percentage	Testing – Samples	Testing – Percentage
and	195	0.039796	4	0.005714	15	0.010714
False	410	0.083673	630	0.900000	960	0.685714

flip-flop	1344	0.274286	21	0.030000	144	0.102857
full-adder	431	0.087959	11	0.015714	45	0.032143
half-adder	679	0.138571	16	0.022857	63	0.045000
multiplexer	371	0.075714	2	0.002857	38	0.027143
nand	203	0.041429	3	0.004286	18	0.012857
nor	210	0.042857	1	0.001429	22	0.015714
not	460	0.093878	5	0.007143	35	0.025000
or	174	0.035510	2	0.002857	21	0.015000
xnor	211	0.043061	3	0.004286	18	0.012857
xor	212	0.043265	2	0.002857	21	0.015000

Table 5: block class distribution

### 3.6.1.3 Classifier

Two different models were tested for the classification task. The first model is a standard CNN architecture structured and trained from scratch, including three pairs of Convolutional and Max-Pooling layers and two dense layers, resulting in 5,075,611 parameters, all of them trainable. The second model is a pre-trained Tensorflow VGG19 (Karen & Zisserman, 2014)[68], including 20,880,459 parameters, but only 856,075 trainable, which correspond to the final Softmax layer.

The pre-trained VGG19 model achieves 100% testing accuracy, outperforming by far the standard CNN, so this is what we ended up incorporating into our pipeline. For clarity, we display the training and validation accuracy on the constructed dataset. Validation stands higher initially due to dropout.

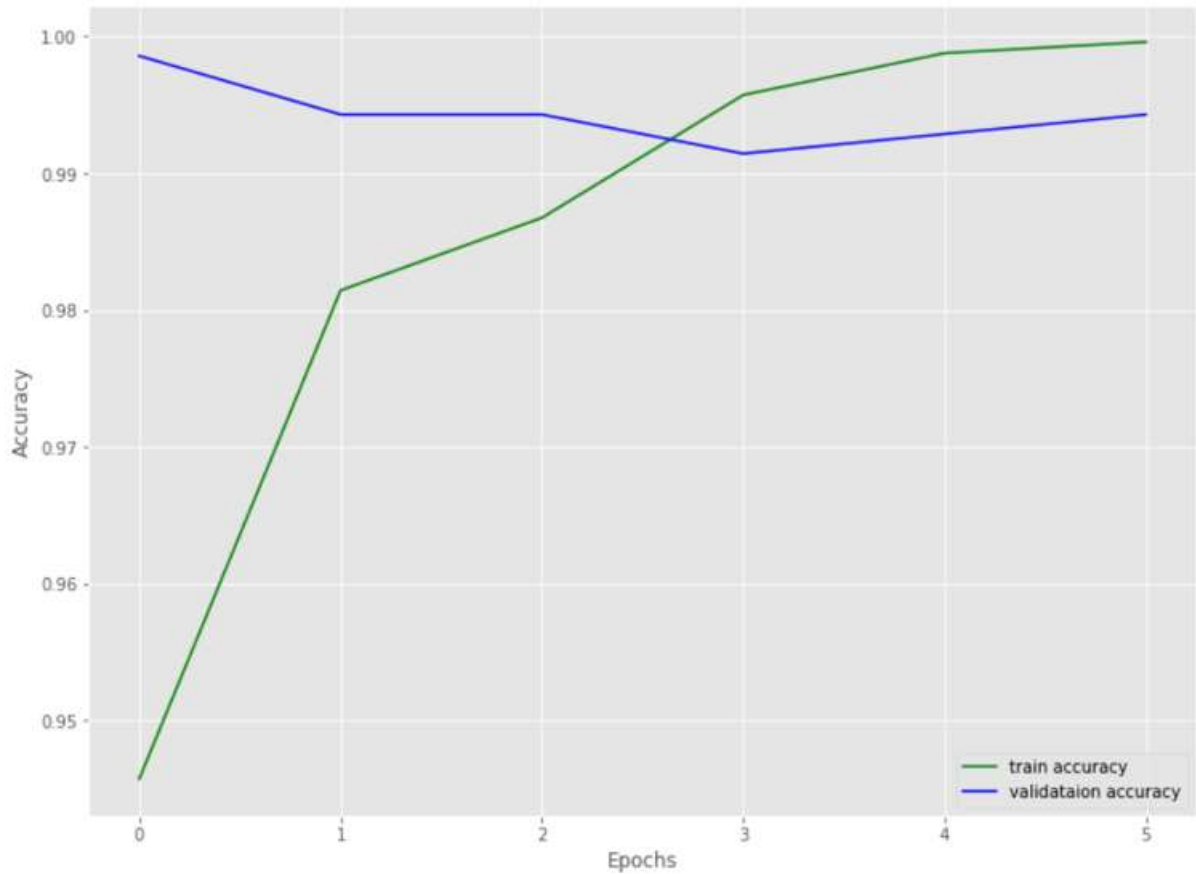


Figure 12: Classification train & validation accuracy

### 3.6.1.4 Single-element Image Captioning

The main mean of inference continues to be the image captioning model. We use the same encoder-decoder transformer architecture, as described previously in the chapter of image captioning, but retrained on an adjusted version of the image captioning dataset. Apart from the cropped images, every other restriction and enhancement mentioned in the classification dataset (i.e.: singular elements, dual representation, extended classes) is also incorporated for the image captioning task. Training and validation accuracy and loss are provided below:

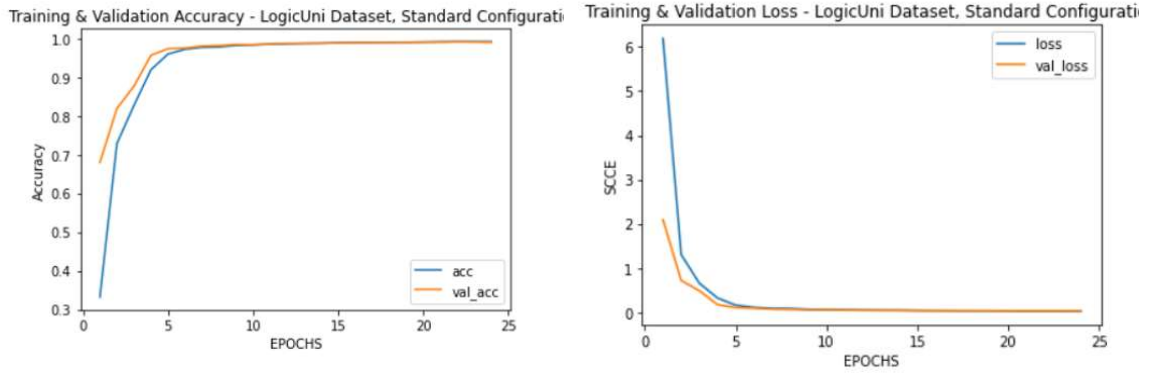


Figure 13: Image captioning train & validation monitoring

### 3.6.2 Scanning Patterns

To improve the efficiency of the overall framework, we propose the use of scanning mechanisms of the digital diagram image to create partitions of the image. The core idea relies on the assumption that the partitions include the essential information of the diagram but present a less challenging case for the image captioning task, thus guarantee a higher accuracy.

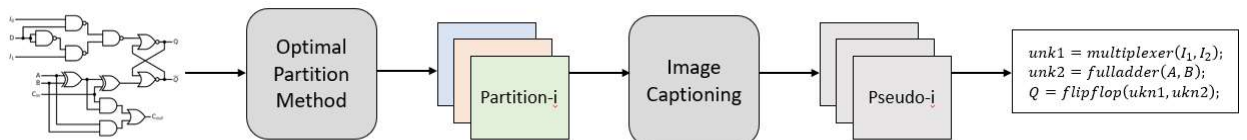


Figure 14: General scanning patterns methodology

The suggested workflow consists of an optimal partitioning of the image into segments, the filtering of the segments through the introduced classification framework, the pseudocode captioning of each image through the introduced captioning framework and finally the assembling of the individual pseudocodes to reconstruct the overall pseudocode that describes the image at hand.

We have already setup the classification and image captioning task through the previously introduced individual frameworks. Therefore, the most challenging part of the presented workflow is finding the optimal partitioning of the image, so that its segments' captioning provides unambiguously the pseudocode of the diagram. Below, we elaborate on four separate approaches to the problem and discuss the advantages and disadvantages the present.

### **3.6.2.1 Grid Search**

The first approach utilizes a grid-like pattern using a pre-specified kernel of rectangle shape to create a set of partitions. Each partition is then filtered through the classification framework, omitting those that classify as *False* (see *False* class in 3.6.1.2). The valid partitions are then forwarded to the image captioning framework, obtaining thus the pseudocode caption for each partition. The generated captions are then assembled to form the overall pseudocode description of the diagram.

To provide the most promising order of actions and avoid ambiguity or indeterminacy, the scanning is performed from the upper part of the image to the lower and from left to right, following hence the flow of the diagram.

The main drawback of the grid search is its dependency on the kernel shape and the stride size. Its simplicity and straightforwardness allows it to be quite efficient for an ideal configuration but fails otherwise. Additionally, complex diagrams with a variety of elements tend to require a more dynamic set of kernels; the static nature of grid search in its simple form overlook essential information. A workaround to these observations is the specification of the kernel shape as a percentage of the image's resolution. Starting with a



large size, we initially scan the image using a kernel size of 70% of its width and 40% of its height, an empirical estimation that aims in generality but can adapt accordingly. According to the return results, we can choose to either reduce or expand the kernel's size in any direction and repeat the process, with the default behavior to be its reduction by 10% and 5% respectively. If no improved results are provided after two iterations, the process stops, but this can also be tweaked by providing the necessary arguments. As for the stride, a lower size is always more effective but increases the runtime significantly. By default, it is set to 15 both vertically and horizontally.

To disambiguate results of overlapping partitions, a hierarchy of classes is set, which is displayed in the Appendix, section A.3.7, together with a detailed case of grid partitioning in section A.3.8. The higher hierarchy level prevails.

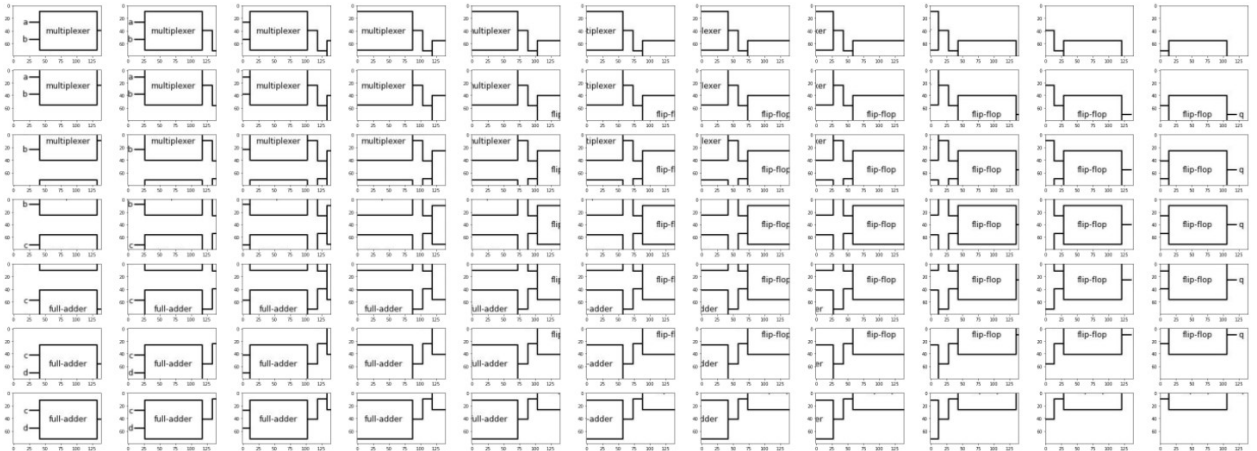


Figure 15: Grid Image Partitioning

### 3.6.2.2 Hough Circle Transformation

As we saw, grid-based approach due to its static nature may require multiple iterations to be sufficient, which rises the time complexity significantly. We address this drawback by exploiting the image's structural features to points of reference for the kernels

to be created at. The presented approach is based on a combination of the *Hough Circle Transformation* and a contour area scanning to identify both circular and rectangular elements and use their center coordinates as the center of dynamically created partitions of rectangular shape. This rids the approach of stride usage, as the image is scanned in neighborhoods, while covering elements of diverse sizes through the application of either *random* or *expanding* kernels, which characterizes the two modes of the approach.

Given a specific image, the application for Hough Circle Transformation help locate the circular centers.

In aid to the discovery of rectangular elements that Hough Circle Transformation does not cover, the calculation of **contour areas** helps to identify potential blocks, which then leads to the derivation of the valid contours' centers and the inclusion of the newly found center points to the Hough Circle Transformation centers.

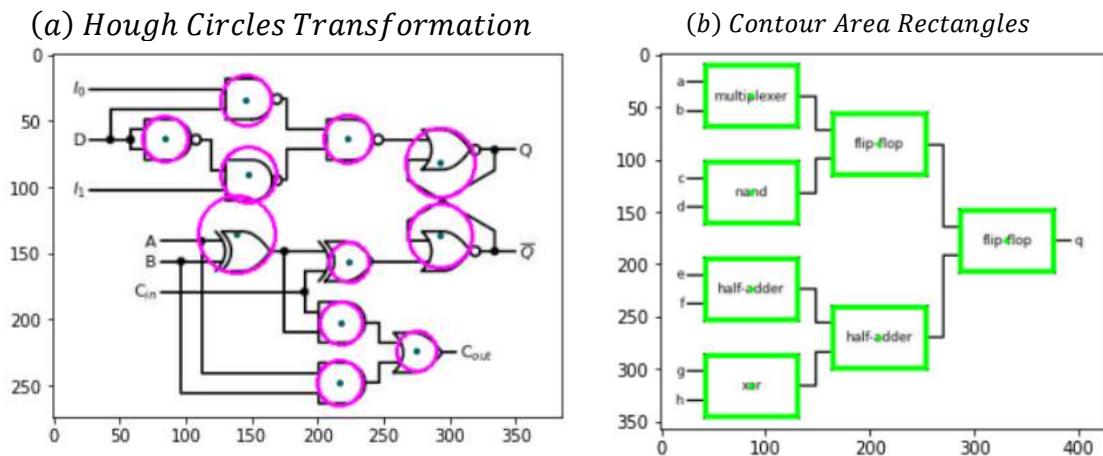


Figure 16: Center discovery methods

During the rectangle discovery, the most crucial parameters are **the acceptance limits of the calculated contour areas**. Based on the noticed ratios of gates and blocks size to image resolution, we empirically set the limits of an accepted contour area upper

limit as 60% and the lower limit as 2.5% of the total image area. In figure 3.55, 138 contour areas were computed out of which seven were valid.

We incorporate the inclusion of these points in both the random and expanding modes, resulting in more accurate and inclusive results.

The discovered centers act then as centers for the creation of rectangular partitions of the image. We test two different modes of creating partitions, one where each partition is created completely at random, and another one where the kernel that define the partition gradually expands. The first mode tested involves the random creation of partitions and is presented below:

#### **A. Random Kernels**

---

#### **Algorithm 3. Hough Circle Transformation – Random Kernels**

---

```
12: define image
13: define centers
14: define internal iterations
15: define out-of-bounds restrictions according to the position of the identified circle


---


16: while (centers  $\neq \emptyset$ ) do
17:   for every iteration in internal iterations do
18:     randomly define a partition according to the restrictions
19:     classify the partition
20:     if partition classifies against the False class do
21:       save the partition
```

- 
- 22: cast a majority vote on the partition's predicted classes according to accumulated probability
  - 23: remove circle centers included in the partition according to their distance from its contours
  - 24: predict caption for each chosen partition
  - 25: assemble the overall pseudocode
  - 26: **return** pseudocode

Figure 17: Hough-based random kernels algorithm

Figures 3.57 displays the initial to final states. Figures 3.58 and 3.59 show the prediction probability majority vote and the center exclusion mechanism respectively.

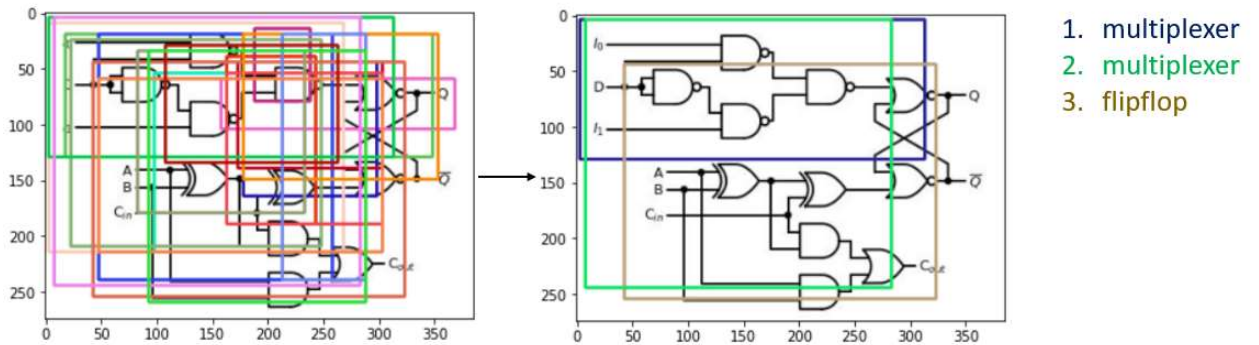


Figure 18: Application of random kernels and final partitions

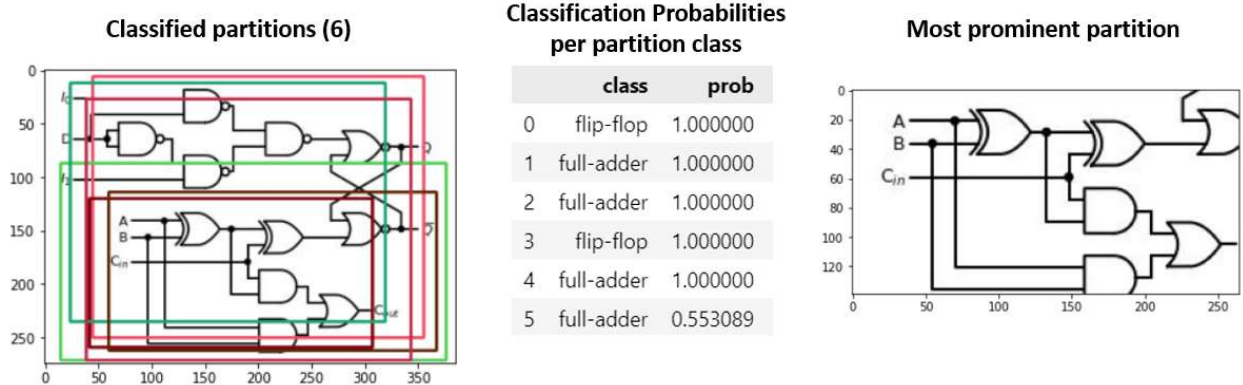


Figure 19: Prediction probabilities majority vote. Classifies to full-adder and predicts caption  
`< start > unk3, unk4 = full_adder ( v , o ) < end >`

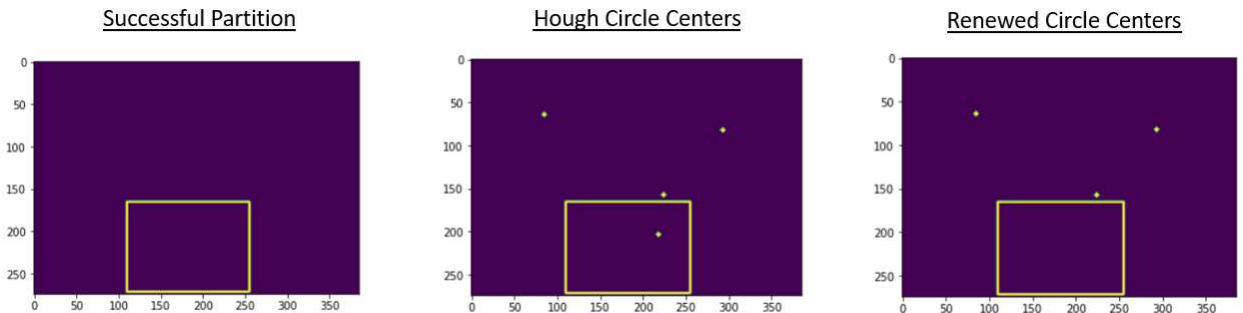


Figure 20: Exclusion of covered centers

The main drawback of this approach is that randomness is not always beneficial, as it fails to capture certain elements. Additionally, we cannot ignore that this is an iterative stochastic system – it depends heavily on the number experiments performed per found center. More iterations tend to produce more accurate results – mainly due to the nature of the problem and the hierarchical filtering – but render the method more expensive.

## B. Expanding Kernels

As a way to minimize randomness and produce more stable results, we alter the randomly chosen kernels to an expanding set of kernels, but in exchange, four expansion

steps, one per kernel side, are required to be set. The kernel spirals out until it reaches the set bounds of the image and no more incremental steps can be performed. For multiple observations, prioritize elements with higher hierarchy in the form of **weighted probability accumulation per class**, as due to small kernels, single gates are more frequent. The rest of the method remains the same.

---

**Algorithm 4. Hough Circle Transformation – Expanding Kernels**

---

```
1: define image
2: define centers
3: define expansion steps
4: define kernels of predefined shape per center
5: define out-of-bounds restrictions according to the position of the identified circle

6: while (centers  $\neq \emptyset$ ) do
7:     if a kernel can be expanded in-bounds do
8:         expand kernel according to steps
9:         classify the partition
10:        if partition classifies against the False class do
11:            save the partition
12:        cast a weighted majority vote on the partition's predicted classes
            according to classification accumulated probability and the class hierarchies
13:        remove circle centers included in the partition according to their distance
            from its contours
```

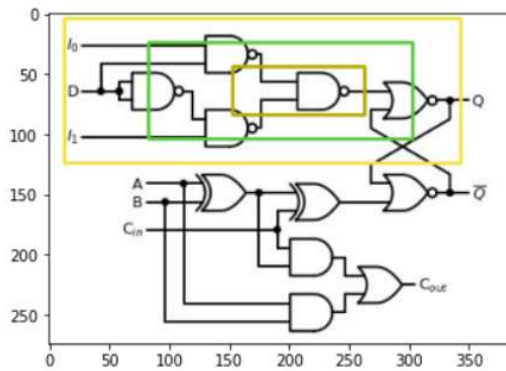
14: predict caption for each chosen partition

15: assemble the overall pseudocode

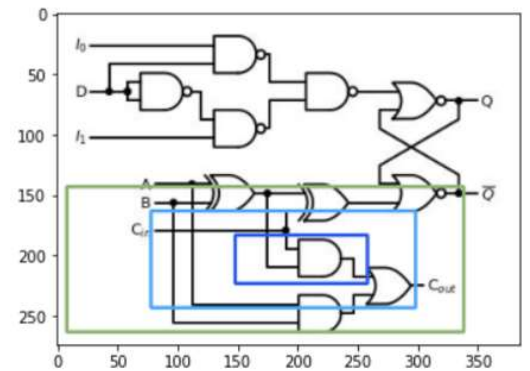
16: **return** pseudocode

Figure 21: Hough-based expanding kernels algorithm

Figure 22 displays the hierarchical priority. Figure 23 shows the weighted majority vote and caption prediction. Figure 24 shows the most prominent partition in rectangle shapes.

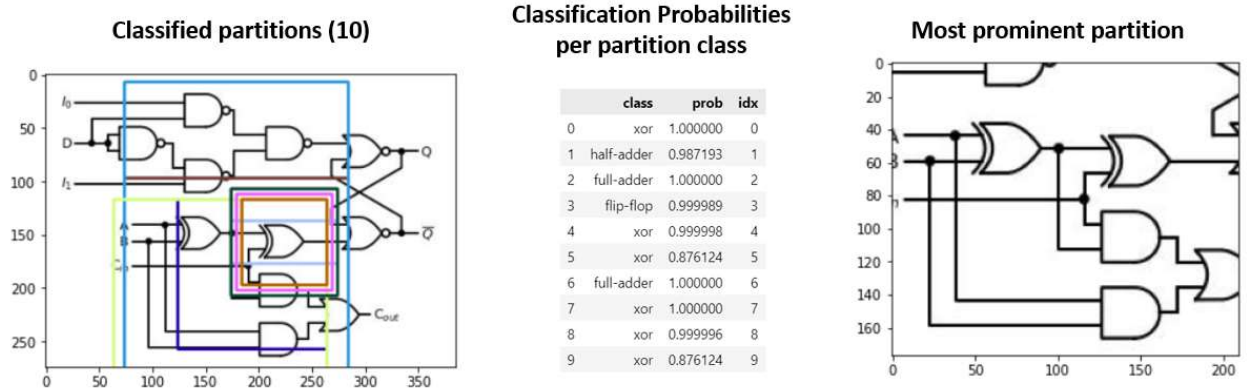


Multiplexer  $\geq$  Multiplexer  $>$  Nand



Fulladder  $>$  And  $>$  False

Figure 22: expanding kernels and block hierarchy



Classifies as **full-adder** and predicts the caption: < start > z , y = full adder ( y ) < end >

Figure 23: majority vote of most prominent partition

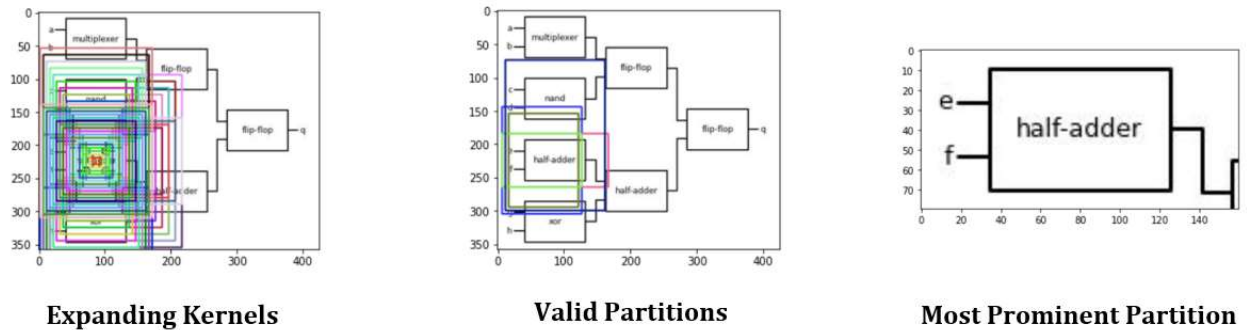


Figure 24: initial, valid 7 most prominent partitions

### 3.6.2.3 Genetic Algorithm

The main drawback of the random and expanding kernel approaches lies in their dependency on the Hough Circle Transformation and contour area calculation techniques for the discover of point of interest. To overcome potential failure on the discovery of such centers, we propose an alternative take on the scanning patterns by reducing to a global optimization problem. Considering the partitions of the original image as they were previously introduced, we construct an objective function for the potential solutions, that is proportional to the hierarchy of suggested partitions and inversely proportional to the



number of suggested partitions. We also consider the density of non-background pixels per partition as a regularization term to avoid vague partitions with high hierarchy. The resulting optimization task is:

$$\operatorname{argmax}_l \sum_{i \in p(l)} h(i) - w_n * n(l) - w_d * d(i) \quad (3.6.1)$$

Where  $l$  is the potential solution,  $p(l)$  are the partitions in  $l$ ,  $n(l)$  number of partitions in  $l$ ,  $d(i)$  is the pixel density of partition  $i$ , while  $h(i)$  is the hierarchy of the partition  $i$ . The weight  $w_n$  on the number of partitions acts as hyperparameter to our system to monitor the importance of the number of partitions. Same stands for each partition's non-background pixel density. It correlates with the use of partition filtering mechanisms and is discussed further below.

To solve the task, we utilize an optimization scheme inspired by genetic algorithms, that acts by encoding for each potential solution the partitions and their number as genes.

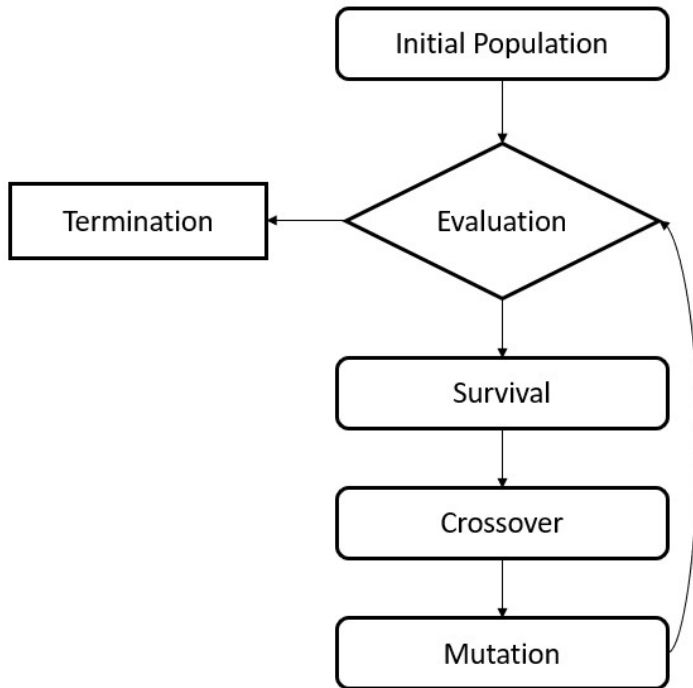
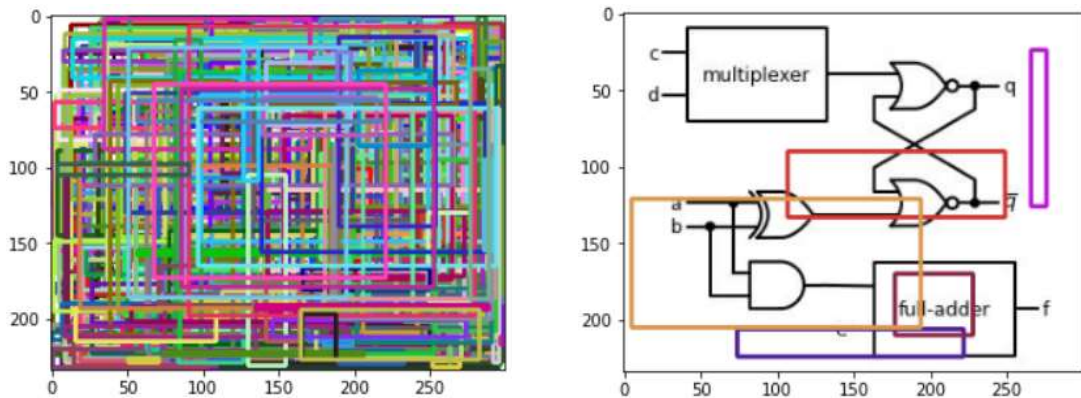


Figure 25: Genetic algorithm flowchart

Below, we present each of genetic operators and the stages followed during the genetic algorithm.

### A. Initial Population

Each individual comprises a set of partitions of the original image and represents a potential solution for the partitioning problem. The genes consist of a) the number of partitions contained, and b) the partitions themselves. The partitions can be performed with each of the three methodologies that have been already shown, i.e., variant-shaped grid partitioning, random & expanding kernels on the Hough Transformation-based and contour-based discovered centers. To address the dependencies observation made at the start, we experimented with the creation of partitions through a rectangle kernel of random size and position, not driven by any a priori knowledge. Each created partition holds an invalid label, unless it is successfully classified to a non-False class through the classification framework that was introduced in 3.6.1, similarly to the three previous approaches. For the experiments that follow, each individual is created through 1000 random partitions, while the size of the initial population is set to 50 individuals.



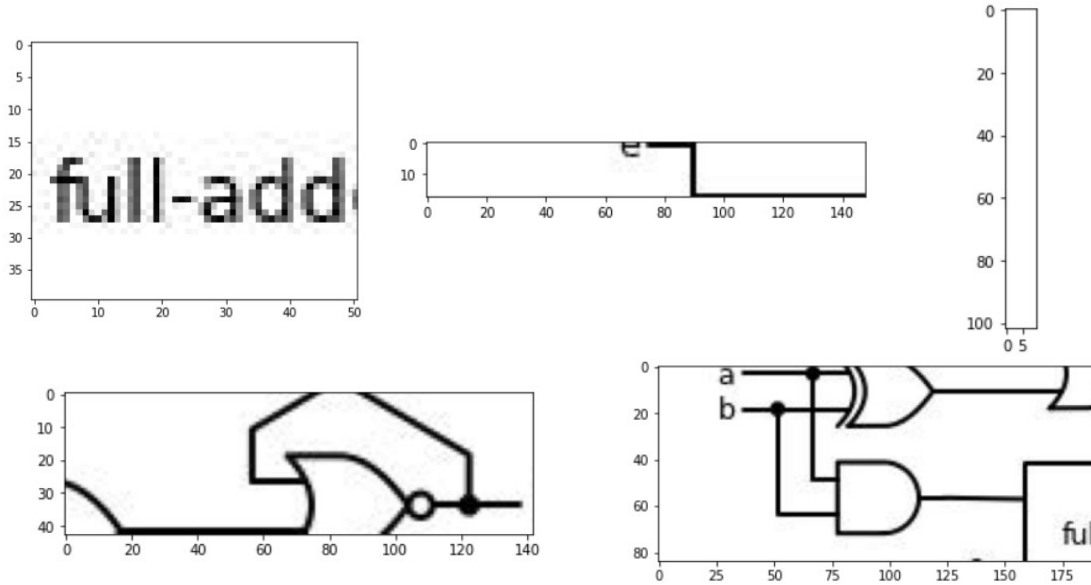


Figure 26: Creation & filtering of random partitions

## B. Evaluation

Due to the repetitive creation process, it is possible that certain partitions will overlap, introducing redundancy. To avoid such circumstances, we interchangeably use two filtering methods.

During the first one, each valid partition is passed to the image captioning framework and its caption is compared to the others, under one individual's scope. In case of identical results or extreme similarity based on their Levenshtein distance the partition is discarded.

Prediction	Target	Levenshtein
$p = flip\_flop(g, g)$	$h, e = half\_adder(e)$	0.6582278481012658
$h, e = half\_adder(e)$	$p = flip\_flop(g, g)$	0.5316455696202531
$e = half\_adder(e, g)$	$e = half\_adder(e, e)$	0.975
$e = half\_adder(e, e)$	$e = half\_adder(e, g)$	0.975

Table 6: caption-based filtering

The second filtering method uses the kernel contours to search for intersections in between two partitions. In the event of an intersection larger than 50% of the partition's

overall area, the partition is omitted. In case both partitions are covered by 50% from the intersection, the one with the least area in total is omitted. The accepted percentage for the discard is an introduced hyperparameter, which for now is used as 50%, as it seems to cover most of the cases where redundancy occurs.

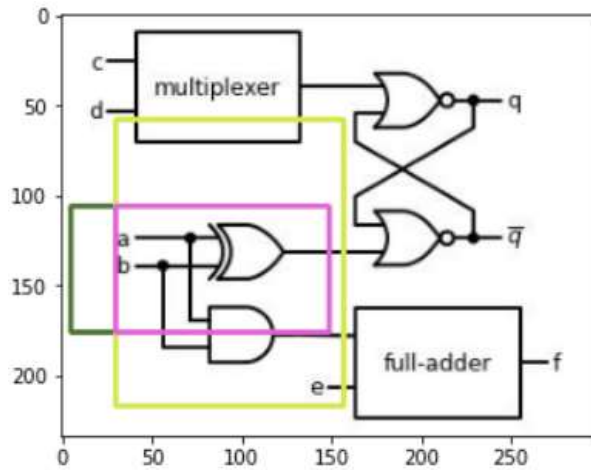


Figure 27: Intersection-based filtering

After removing any potentially redundant partitions, the objective function expressed in (3.6.2.3) is used to calculate the fitness score for each individual.

### C. Survival

We follow an explicit survival phase, where the fittest 40% of the initial population (50% for each consecutive generation) survives to the next generation. To add stochasticity, we also incorporate 10% of the remaining population as randomly selected individuals, without advising any fitness criteria.

This creates the first fragment of the new generation, which corresponds to 50% of the initial population. The rest consists of offspring created during the crossover operation.

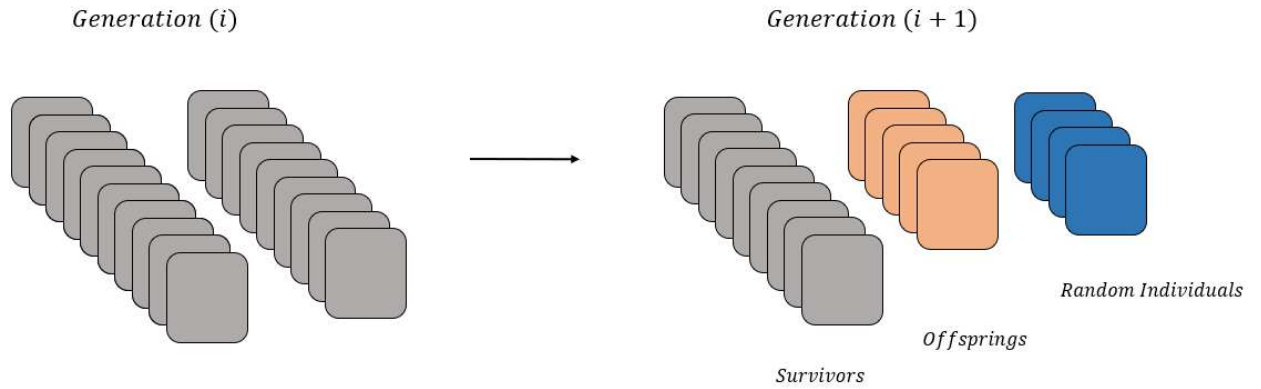


Figure 28: Next generation composition

#### D. Crossover

The crossover process involves the use of two parental individuals, uniformly sampled without replacement from the surviving population. Each gene is then selected equiprobably between each parent chromosome. The first gene that is defined is the number of partitions of the offspring. Afterwards, each partition is uniformly sampled from the set of the collected partitions of the two parents, until the selected number of partitions is met. Crossover takes place until the number of individuals created reaches 40% of the initial population.

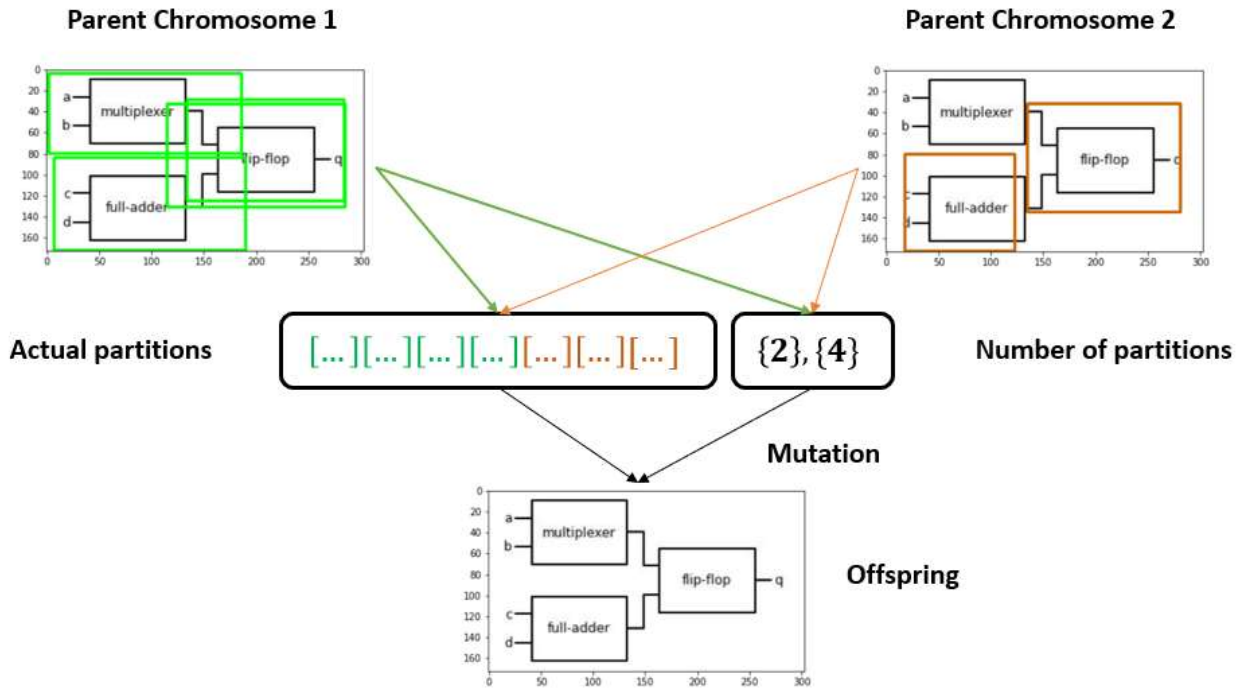


Figure 29: Crossover scheme

## E. Mutation

Mutation takes place during the creation of each offspring. Two different forms of mutation are used, one for each encoding of genes we incorporate, while the mutation rate remains for both as 10%. During the number of partitions selection, a uniform sampling is performed in the real interval  $[0,1]$ . If mutation rate exceeds the sampling, the number of partitions is randomly increased or decreased  $\pm 1$ . The second form of mutation involves around the selected partitions. Again, the same uniform sampling is carried out. If it falls behind the mutation rate, the partition at hand is replaced by a random valid partition that is generated on the fly. Both of these operations provide randomness to the method, increasing its search space significantly.

## F. Termination

Crossover and mutation are followed by a re-evaluation of the whole generation to charter the crossover byproducts. This step concludes the creation of the next generation.

The terminal criteria of the algorithm is the completion of a certain number of generations. The fittest individual during the last generation constitutes the proposed solution to the optimization problem stated in 3.6.2.3.

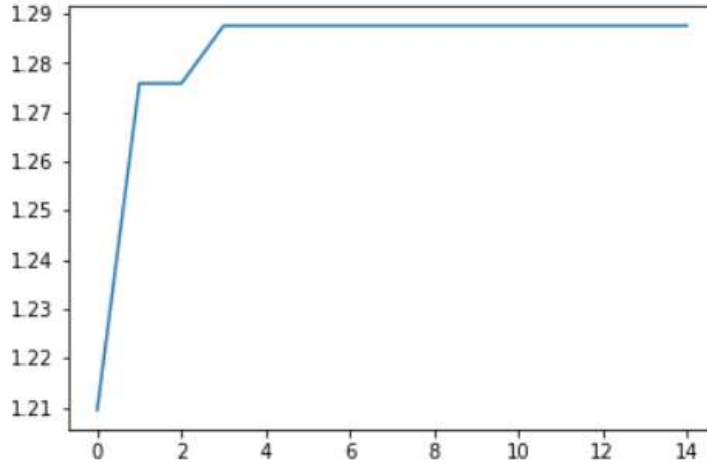


Figure 30: Monitoring of the fittest individual per generation, in a 15-generation run

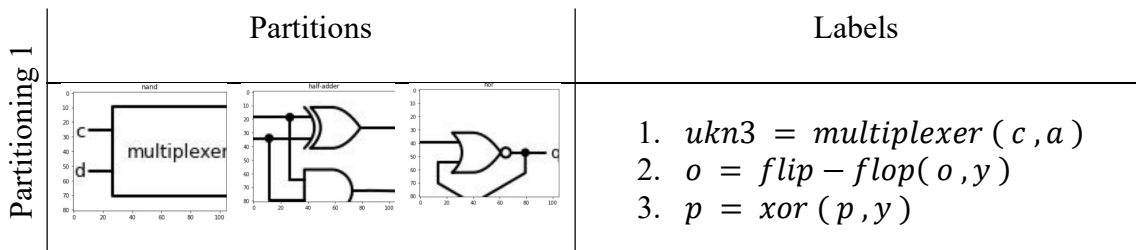
### 3.6.2.4 Modified Ensemble Method

In sections 3.6.2.1 – 3.6.2.3, we proposed three different families of methods for the solution of the automated pseudocode generation problem through image partitioning and analyzed their advantages and potential weaknesses. In this section, we explore the merging of these approaches and the effect it has in the overall inference. Inspired by the stacking paradigm of ensemble methods, we assume the three introduced approaches as classifiers and utilize the genetic algorithm in 3.6.2.3 as a level-1 meta-classifier for the overall inference. A similar use case of the genetic algorithm as classifier can be seen in (Sicora & Al-laymoun, 2014) [67].

Each involved partitioning method's outcomes are considered as classification outcomes and correspond to a pre-defined fraction of the initial population. E.g., 30%

percent of the initial population may be derived from a grid partitioning with various kernels, 30% percent from Hough circles and contour centers with expanding kernels, 20% with random kernels and the final 20% with the completely random partitions. The splitting proportion itself constitutes a hyperparameter for ensemble method and requires experimentation / fine-tuning. We propose the presented setup as it includes but restrains highly abstract partitioning, while it mainly leans on more data driven choices, like the expanding kernels and grid-like scans.

In the same manner, the ensemble method may consist of more controlled level-0 predictors and meta-classifiers. Below, we consider the case where the classifiers comprise predictions solely from grid scans using varying kernels. Both the kernel and the two-dimensional stride are specified on fractions of the width and height of the original image. Empirically, the stride is set in between 5 – 10% of the corresponding dimensions to avoid extended running times and frame appropriately the diagram’s elements. The example below performs a grid scan on the percentage space  $[0.35, 0.45, 0.55, 0.65, 0.75]$  for both dimensions. We present samples of the 25 predictions. The overlap and caption filters introduced during the genetic algorithm are also incorporated.





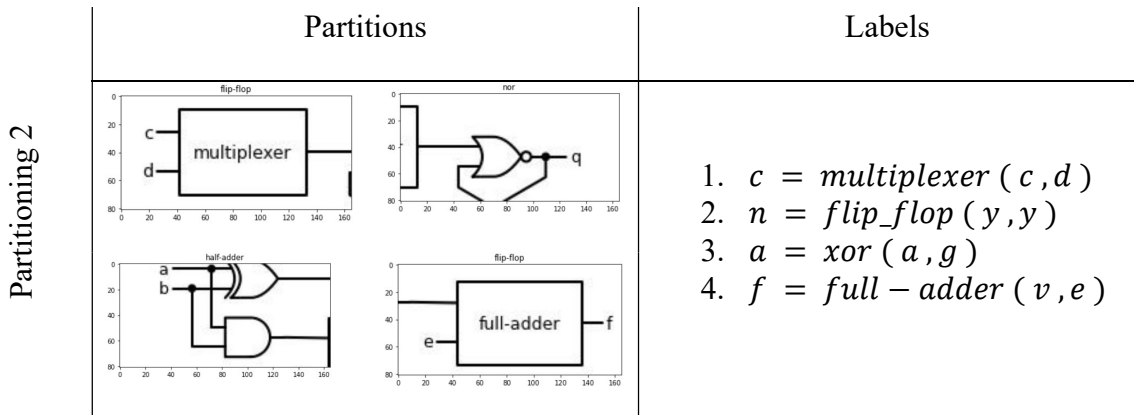
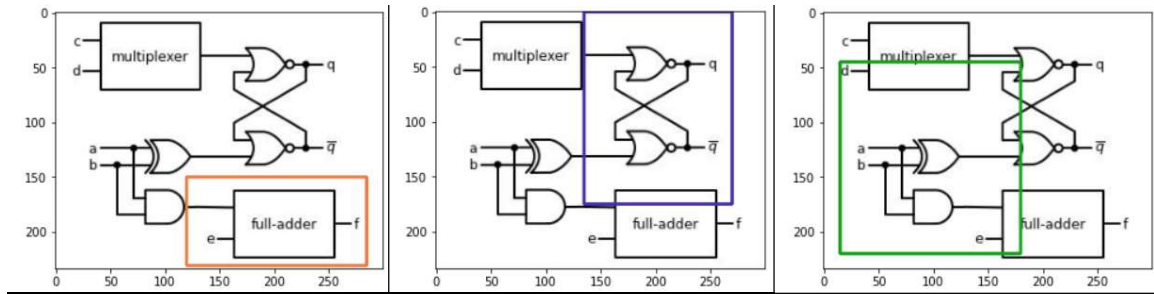


Figure 31: Ensemble method partitioning

Three of the 25 partition sets are able to capture all the diagram elements and produce the core pseudocode. The first spotted error concerns the variable names of the input and output elements of each block. As the partitions are rarely ideal, the model may be led to misclassify variable names that are missing or of similar silhouette, as seen in figure 3.75 (3). We resolve such occasions by comparing with other partitions captioned with the same block name, and use the majority of occurrences, whenever this is possible. There are also other post-processing solutions that involve approaches we do not incorporate yet in our framework, e.g., **the use of simple rules to identify the misclassification, like the same input and output variable name, and the use of OCR methods to limit the scope of acceptable variable names.**

The second issue spotted is the ambiguity created by the varying kernel scans. E.g., for the same region shown in figures 3.75 (1) and (2), a smaller partition is identified as a “xor” gate, while a larger partition capture the composite element “flip – flop”. We sort this issue out by re-applying the overlapping filter introduced in the genetic algorithm, now to the collection of prediction sets, instead of applying it on each prediction set separately. In our previous example, it limits the number of partitions from 25 to 3.



$$f = full\_adder(v, e)$$

$$r = flip\_flop(p, p)$$

$$y, e = half\_adder(a, e)$$

Figure 32: result after intersection filtering

The post-processing techniques introduced here, as the ones discussed above, are able to fine-tune the results of the ensemble approaches to a sufficient extent, therefore provide a solid approximation of the diagram's pseudocode. The automation and generalization properties of the approach are obvious, but some of the processing layers set new hyperparameters that require fine-tuning and cause the complexity of the system to rise. In our current setup of the grid ensemble method with various kernels, the parameter that remains to be fine-tuned is the stride of the kernels (see figure 3.77). Different stride sizes result in a different number of overall accepted partitions, and this can be controversial. There is always the solution of a set of values according to the dimensions of the image, but this would increase drastically the time complexity. The use of Hough Transformations or other contour discovery methods also do not seem appropriate, as they fail to perceive the complex elements of a diagram as entities. The situation resembles of unsupervised methods, similar to how the number of neighbors should be declared in KNN. We leave this parameter under discussion, while we set a default of 5% of the image's corresponding dimension.

Image Dimensions		
(234, 300)		
Stride Size	Partitions	Final Partitions
(5, 5)	43	3

(15,15)	32	3
(45, 45)	22	3
(70, 70)	16	2

Table 7: Number of partitions and stride size relation

Lastly, the processed predictions can either be presented in this state, or they can be forwarded to a meta-classifier, completing the ensemble method. At this point of our research, we were not able to expand the scope of our experiments towards this direction, as the lack of an annotated dataset on complex digital diagrams prohibits the use of supervised-learning classifiers. The generative framework introduced earlier in 3.6 is strictly restrained to single elements, while the generative framework introduced in 3.4 is capable of generating complex diagram images but is again restrained solely on digital gates; thus, neither can generate an appropriate annotated dataset for our purpose. We plan though to address the matter in future research efforts.

### 3.6.2.4 Experimental Results

We test the efficacy of the four presented approaches based on two scaling experimental phases, using default configuration of hyperparameters. We conclude by mentioning the uniqueness of each method and the importance parameter fine-tuning per case.

#### A. Single Entities' Samples

The first experimental phase consists of consecutive runs on the single entities' caption dataset, described in 5.1. For roundness, we incorporate two string metrics for evaluation of the results. We use the Levenshtein distance (Vladimir I. Levenshtein, 1965) [69] as the main metric to compare the generated captions with the ground truth, while we complement the analysis with the Sørensen-Dice coefficient (Lee R. Dice, 1945) [70] to

counterbalance the unwanted edit penalty that is occasionally applied by the Levenshtein distance. Each metric is used for different sample collections.

We slightly adapt the procedure followed for each method, so that it corresponds to its specifics. The method based on Hough circle transformation and contour areas with random kernels, presented in 3.6.2.2 (A) and abbreviated as HTCR, is applied on 20 random samples, performing 5 iterations on each sample, returning an average of the 100 performed predictions. The variation with the expanding kernels, seen in 3.6.2.2 (B) and abbreviated as HTCE, omits the 5 repetitions per sample, returning the average of the 20 single runs. Despite its non-deterministic nature, the genetic algorithm-based method with random kernels from 3.6.2.3, abbreviated as GAR, runs without repetitions on the same sample; instead, it eliminates occasional unpredicted behavior through large initial population sizes (50). Both GAR and the ensemble-based method (EGP) from 3.6.2.4 are tested on a smaller subset (5) due to their exorbitant requirement of building individual cases. Figure 5.6 contains the experimental results for each method.

	<i>GAR</i>	<i>HTCR</i>	<i>HTCE</i>	<i>EGP</i>
<i>Edit</i>	0.3866	0.9187	0.6682	0.2646
<i>Dice</i>	0.5556	0.8552	0.600	0.2941

*Table 8: Single-entities dataset experimental results*

We observe that, on the single entities’ task, GAR and EGP score low, mainly due to their inability to capture an appropriate partition, as the global optimization approach used may get stuck at local optima. Most of the inaccuracies occur when the partition partially captures is falsely placed yet classified and retained. The overlapping filtering method also contributes to this phenomenon, as occasionally larger but misplaced

partitions overcome others, smaller but ideally placed. Thus, the content of a box block may be falsely interpreted, or the gate of single composite element captured in a relatively large partition may keep the method from obtaining the bigger image. Capturing multiple items that do not belong to an existing entity simultaneously also causes the model to fail, as it is trained on single entities; this includes one of our future goals but requires the creation of generative framework anew. These observations apply also to the HCTE, but on a lower frequency. We obtain the best results for HTCR at 0.9187 Levenshtein score and 0.8552 Dice coefficient, as it rely mainly on the filters, is not prone to local optima and its plain mechanics allows it for a great number of random partitions without a severe efficiency penalty.

## B. Complex Samples

During the second experimental phase, we test the four approaches on individual complex images, containing multiple, interconnected entities. The images are manually designed through the proposed graphical framework in 3.4, cover both representations, and exhibit varying complexity.

It was also observed that, while blocks and elements where frequently predicted accurately, the Dice and Edit metrics applied a penalty due to the character order mismatch from the provided label. While this is in general a desired property, it does not reflect in our instances, as the commands can be reorder with ease, according to the presence of variables. To overcome this, we complementary include as third metric a fuzzy variation of the Edit distance, as described in [71]. In a nutshell, each string outcome is tokenized and then alphabetically ordered and reassembled. The remaining setup follows as in the single entities' task. The results can be seen in figure 3.80.

	<i>GAR</i>	<i>HTCR</i>	<i>HTCE</i>	<i>EGP</i>		<i>GAR</i>	<i>HTCR</i>	<i>HTCE</i>	<i>EGP</i>
<i>Edit</i>	0.333	0.6423	0.7532	0.3451	<i>Edit</i>	0.3106	0.4298	0.3476	0.4127
<i>Dice</i>	0.4351	0.3156	0.3684	0.2526	<i>Dice</i>	0.4351	0.2263	0.1913	0.2286
<i>Fuzzy</i>	0.5870	0.6360	0.6800	0.7000	<i>Fuzzy</i>	0.4700	0.4580	0.5100	0.4700

	<i>GAR</i>	<i>HTCR</i>	<i>HTCE</i>	<i>EGP</i>
<i>Edit</i>	0.2059	0.6078	0.5059	0.5481
<i>Dice</i>	0.3478	0.3728	0.4286	0.3913
<i>Fuzzy</i>	0.4923	0.4820	0.5600	0.4000

Table 9: Results on complex samples

While HTCR and HTCE are now ill-performing around the values of  $\sim 0.2$  for both Levenshtein distance and Dice coefficient, GAR and EGP are able to maintain their performance. In essence, the methods continue to exhibit the same behavior, thus the complex diagram structure allows GAR and EGP to discover multiple elements and differentiate between the partitions in a more sophisticated manner. Instead HTCR and HTCE seem to discard overlapped findings due to the straightforward partitioning, therefore lacking in performance compared to the other two in context of complex diagrams.

### C. Parameter Configuration

In every experiment performed, we utilized the models using the default configuration of parameters, as described in their respective sections above. We already proved that the methods work sufficiently well in a single entities setup and provide promising leads in case of complex diagrams. Still, the parameters play a crucial role in

every method's case. E.g., using medium to small kernels with small strides in the modified ensemble method significantly help to identify multiple blocks in complex images, while the setup prohibits the discovery of large single entities, especially in case of box representation. See section A.3.9 in the Appendix for a detailed instance. In addition, parameters like the kernel sizes and the stride set correlate with the intersection threshold used during the overlap filtering method from 3.6.2.3 (B), both in EGP and HTCE. Considering all the above, although we showcase the default behavior of methods, we suggest the studying of an extended search space of parameters and their combination per method, for a rounded set of results.

We conclude with a side-to-side comparison of the four proposed methodologies on their predictions on the image from figure 3.65. We use the default configuration for each method, while the GAP is run for 20 generations. Results can be seen in figure 3.85 below.

<b>Label</b>	<b>EGP</b>
$unk1 = multiplexer(c, d)$ $unk2 = half\_adder(a, b)$ $q = flip\_flop(unk1, unk2)$ $f = full\_adder(unk3, e)$	$k = flip\_flop(p, g)$ $d = multiplexer(c, d)$ $g = flip\_flop(y, y)$ $a = xor(a, g)$
<b>HCRE</b>	<b>HCRR</b>
$i = multiplexer(m, w)$ $b = flip\_flop(m, v)$ $d = flip\_flop(y, h)$ $m = half\_adder(m, g)$ $e = half\_adder(e, g)$	$j = flip\_flop(j, q)$ $o = flip\_flop(j, h)$ $q = flip\_flop(y, q)$ $a = flip\_flop(a, a)$ $e = flip\_flop(o, o)$
<b>GAR20</b>	<b>GPH</b>
$g = flip\_flop(p, x)$	$c = multiplexer(c, d)$ $b = flip\_flop(y, g)$ $n = flip\_flop(y, y)$ $q = xor(a, a)$ $q = flip\_flop(a, g)$ $q = xor(a, j)$ $p = xor(a, g)$ $a = xor(a, g)$ $f = full\_adder(v, e)$

Table 10: Comparative experimental results

In general, we get the most accurate and controlled predictions with EGP. HCRE also follows on well but the lack of a final overall filtering makes it vulnerable to ambiguity. GPH is also promising but its simplicity leads it to “local minima” situations, stuck to several XOR gates and ignoring the outer complex element. HCRR and GAR are the only purely randomized methods, which we can see that the post-processing filters do not really benefit. They actually discover most of the diagram’s elements, but small sized partitions or cut-off input-output variable names prompts the intersection and caption filters to remove them. Their results can be extensively improved with more iterations per chromosome or center respectively, but then the time complexity rises. Similarly, more post-processing actions like popularity votes in the candidates or the those stated



previously may also prove beneficial for the rest of methods, but we preferred to present the raw outcomes as is.

As a final touch, we take advantage of the formal scheme presented in 3.3 to convert the predictions to their formal representation. This enables several utilities that were presented during this section, e.g., the *LaTeX* templatings for a better visual representation. An example of the HCRE results in formal representations can be seen in table 11:

### HCRE

---

multiplexer:  $IN(m, w)|DS(\text{None})|OUT(i)$   
 flip – flop:  $IN(m, v)|DS(\text{None})|OUT(b)$   
 flip – flop:  $IN(y, h)|DS(\text{None})|OUT(d)$   
 half – adder:  $IN(m, g)|DS(\text{None})|OUT(m)$   
 half – adder:  $IN(e, g)|DS(\text{None})|OUT(e)$

*Table 11: conversion to formal representation*

As it is accustomed, we display for reference the average of running time in seconds for each method. GAR appears specifically time consuming as we also include the creation of chromosomes, but this action can be performed separately, thus minimize the required time to approximately four minutes. The experiments were performed in a system using 16 GB RAM, and an Intel Core i7-10750H CPU at 2.60GHz, 2592 MHz with 6 Cores and 12 Logical Processors, while the two neural models run through Tensorflow CUDA compatibility on a NVIDIA GeForce RTX 2060.

	<b><i>GAR</i></b>	<b><i>HTCR</i></b>	<b><i>HTCE</i></b>	<b><i>EGP</i></b>
<i>sec</i>	2700/240	480	580	155

*Table 12: Running time per method*

## 3.7 Conclusion

In section 3, we addressed the problem of automated pseudocode generation from digital diagram images as found in technical documents and proposed a complete and thorough methodology for the automated generation of a pseudocode under certain conditions, without any a priori or contextual knowledge. We began in section 3.1 by defining the conditions and the objectives and by referring to the previous efforts on the subject. In 3.2 we revisited the philosophy of the existing efforts and proposed a new methodology that is based on the functional modelling of the diagram blocks. Following the revisiting proposition, in 3.3 we introduced in detail the functional modelling scheme of the diagram blocks and produced pseudocode, based on a specifically designed context-free formal grammar. Through the formal modelling scheme, we created a generative framework that allows for the creation of pseudocode-annotated diagram images from scratch and showcased the internal representations and mechanisms. In 3.4 we utilize the framework to create the annotated dataset and therefore map the initial problem into the well-known supervised learning task of image captioning. We then present analytically in 3.5 all the followed steps and progress insights, reaching an initial accuracy of  $\sim 70\%$ . Due to the image resolution restraints and other bottlenecks found in the image captioning task, we propose in 3.6 a workaround involving recursive and non-recursive scanning patterns and their analysis, a modified version of the generative framework and two novel image classification and image captioning datasets on digital diagram images. Additionally, we expand the existing scope to a more dynamic setup and propose a varying family of approaches. These efforts results in promising experimental results and sets the path to a new family of approaches on the field of automated pseudocode generation.

In our future steps on the subject, we aim to improve on both the complex-case image captioning effort and image partitioning approach. We already work on introducing external tools for the correct disambiguation of the produced partitions in the ensemble approach in 3.6.2.4, as well incorporating the wider scope in the complex image captioning scheme of 3.4, something we are aspired it will benefit the overall efficiency.

## 4. Analysis & Synthesis of Mathematical Formulas

## 4.1 Introduction

The following section comprises the second branch of a unified effort on the automated, deep, and semantic understanding of technical documents and their components. Presented as a subtask of the automated document analysis, which as a domain of artificial intelligence and data mining has been receiving increasing attention over the late years, this branch's main aim involves around the modality of mathematical formulas, as they are found in technical documents. Mathematical formulas constitute a multifaceted module, being the subject of various tasks that have received from plenty to the minimum of attention over the literature, with one of the most popular tasks being their discovery in a document (Zanibbi et al, 2002) [90] and their successful parsing (Fateman et al, 1996) [89].

Within the section that follows, we focus our efforts on the semantic understanding of the mathematical formulas. Being an abstract notion, by semantic understanding we mean any substantial information that can be extracted on the utility and the validity of a given mathematical formula. Of course, given a mathematical formula from a technical document, there is openly available, a priori mathematical knowledge that can be utilized, using common reasoning tools or existing frameworks. What is not always so obvious or easy to get, is how the given formula synergizes with other mathematical entities; in other words, how can we use a given valid mathematical expression, describing a certain phenomenon, to expand our knowledge or our inference capabilities on the topic at hand or its extensions. Striving to take the subject a step further, the presented methodology utilizes any given mathematical formula as an object with structural features and aims in extracting any potentially valid metaknowledge that could prove beneficial. Specifically,

we combine existing common knowledge and techniques with our own contributions, in order to create a modelling and managing framework for given mathematical formulas, under the veil of context-free formal grammars. The created managing framework allows for the composition and decomposition of the mathematical formulas, resulting in new synthetic mathematical objects, derivatives of designated plausible formulas but of unknown validity and utility. Our aspiration is that, through the processes of elimination or contradiction, these synthetic elements may contribute to the acquisition of useful metaknowledge on several scientific subjects, obtained in a sound manner by using common mathematical knowledge and existing formulas that describe state-of-the-art results. The sub-sections that follow in 4.2 describe the context-free formal grammar that constitutes our modelling framework and the main analyzing methodology for a given mathematical formula. Through the process, we highlight the restrictions that arise, while in 4.3 we offer outsourcing solutions and deep learning adaptations that counter several limitations. We conclude by performing an extended proof of concept on the overall synthesis framework in 4.2.11 and 4.3.5 and a discussion on the introduction of task and its future in 4.4.

## **4.2 Synthesis Methodology**

### **4.2.1 Formal Modelling of Synthesis Framework**

Before introducing the methodology of formulas' synthesis, we establish a formal mathematical modelling of the subject formulas based on a context-free formal grammar. Apart from providing coherence and structure to the subjects of the framework, the basic sets of the grammar, and specifically the production rules, introduce the basic principles

and the core mechanisms of the framework, while its broad and dynamic form offers expandability and adaptiveness.

The context-free grammar adopts the standard form of formal grammars (Chomsky, 1956)[72] (Chomsky, 1957)[73]:

$$G = \{N, T, S, P\} \quad (4.2.1)$$

Aiming to model any mathematical term that could be employed by the synthesis methodology, the fundamental sets of the grammar are defined in a dynamic and broad setup:

- $T$ : includes all the mathematical symbols and terms representing variables, constants, operators and special notation of punctuation, and their valid combinations, whose implementation is supported by *LaTeX* (Lamport, 1986)[56]. A detailed list that surpasses the mathematical scope can be seen in the work of (Pakin, 2021)[74]. Excludes only the terms that may by chance be identical to a term belonging to the non-terminal set  $N$ . It results to an infinite but enumerable set, as there exist 1-1 mapping to the set of natural numbers  $\mathbb{N}$ .

- $N$ : based on identification by regular expressions, it consists of all the elements of the terminal set, attached with the prefix “*nts\_*”, with its definition displayed below. Although it would be ignorant to make such a claim, the goal of the notation is to avoid any overlapping with existing mathematical terms, so that no restrictions are imposed on the terminal set  $T$ , and inductively to the analysis framework. Derived by  $T$ ,  $N$  is infinite but enumerable.

$$N = \{nts\_ \}. T = \{z: z = xy | x \in \{nts\_ \}; y \in T\} \quad (4.2.2)$$

-  $P$ : conceptually similar to  $T$ , it includes all the mathematical operations whose implementation is supported by *LaTeX*. The operations are occasionally discretized into unary and multi-parametric, as two different implementation methods are offered, discussed further in the methodology's presentation.

-  $S$ : defined as the prefix of the non-terminal symbols.  $S$  is a subset of  $N$ , as  $\{nts\_ \} = \{nts\_ \}. \{\varepsilon\}$  and by definition  $\{\varepsilon\} \in T$ .

$$S = \{nts\_ \}$$

The context-free formal grammar constitutes the foundation of the analysis' framework and defines its dynamic boundaries. During each application of the framework, a formal language derived from the grammar is used, that narrows the scope of the framework on the task at hand. Examples of the framework's implementation and adaptiveness are given in detail in the following sub-section. Notations are simplified to avoid redundancy. Binding operators correspond to the production rules  $P_l$  of the language that are used for the formulas' synthesis, while the subjects belong to the language's terminal set  $T_l$ . The schema below is given as an example to display the relations between the assumed formal language and the framework's operating sets. The set  $BO$  corresponds to the binding functions, while the set  $SB$  corresponds to the subjects of the framework. As a clarification, the multiplication symbol  $*$  in  $BO$  is a simplification of the multiplication



operation and implemented as a function, while the same symbol in  $SB$  represents the symbol itself as a character. Same stands for the operation  $\frac{d}{dt}$ .

$$SB = \{m, v, x\} \subset T_l = \left\{m, v, x, d, t, *, /, \frac{dm}{dt}, \frac{dv}{dt}, \frac{dx}{dt}, \frac{dt}{dt}\right\} \subset T \quad (4.2.3)$$

$$BO = \left\{*, \frac{d}{dt}\right\} \subset P_l = \left\{*, /, \frac{d}{dt}\right\} \subset P \quad (4.2.4)$$

## 4.2.2 Syntactic Perspective

The proposed methodology is comprised of two separate analytical perspectives for each formula: the syntactic analysis and the semantic analysis. Under the syntactic perspective, we adopt the common mathematical notion of viewing every mathematical formula as a structured entity, belonging to the presented formal grammar. Adopting the grammar's general form, the composing elements for each formula consist of every mathematical symbol that represents a variable or a constant, bound by the mathematical and relational operators, all of which are included in the grammar's terminal set. As it is accustomed, the manipulation of the terminal symbols is performed through the predefined production rules, discussed above, which also the main mean of ensuring that the grammar's syntactic rules are kept true. As for the final representation, expanding or simplifying computations are supported through the presented framework.

### Target

$$H(F) = - \sum_{i=1}^n P(x_i) \log P(x_i)$$

### Expression

$$\begin{aligned}
S &\Rightarrow G - C \Rightarrow 0 - C \Rightarrow -\sum_{i=G}^C F \Rightarrow -\sum_{i=1}^C F \Rightarrow -\sum_{i=1}^n F \Rightarrow \\
&\Rightarrow -\sum_{i=1}^n G * V \Rightarrow -\sum_{i=1}^n P(x_i)V \Rightarrow -\sum_{i=1}^n P(x_i) \log G \Rightarrow -\sum_{i=1}^n P(x_i) \log P(x_i)
\end{aligned}$$

### **Sequence of applied production rules**

$$S \Rightarrow (52) \Rightarrow (1) \Rightarrow (70) \Rightarrow (2) \Rightarrow (24) \Rightarrow (63) \Rightarrow (26) \Rightarrow (75) \Rightarrow (26)$$

*Figure 33: Formal modeling and production of mathematical formulas*

### **4.2.3 Semantic Perspective**

While the syntactic perspective is strict and specific, based on common mathematical reasoning, the semantic part of the analysis is quite abstract. A formula generated through the above production rules is by default of unknown significance or validity; in most of the cases, it might be syntactically sound, but its semantics and meaning are unvalidated. In contrast, a mathematical formula found in a technical document is usually accompanied by complementing textual information on its utility and its characteristics. Quite often, it is linked to visual elements, e.g., figures and graphs, that also contribute to its semantic knowledge. Even in the unlikely event where no complementary information is provided, the operations found in the formula or its normalized representation can help define its semantic features, based on a priori knowledge, i.e., the sum of multiple terms in decreasing or increasing grade order most likely signifies a polynomial.

Based on these observations, the methodology presented below takes advantage of an a priori knowledge on existing, validated mathematical formulas, in order to seek metaknowledge that could potentially prove useful at understanding or describing phenomena. Although in our experiments we mainly focus on structured physics domains, like the Newtonian laws of motion and their derivatives, the methodology is **domain-agnostic** and could be adapted on any topic, provided the corresponding setup preparation. Essentially, the formal grammar is utilized as a generative framework to create new mathematical formulas, based on two extracted subsets from the terminal set and production rules set respectively. This results on an expanding layered graph  $G_{res}$  plotting the interactions between formulas through common mathematical operations, where the nodes represent mathematical expressions, while the edges correspond to mathematical operations.

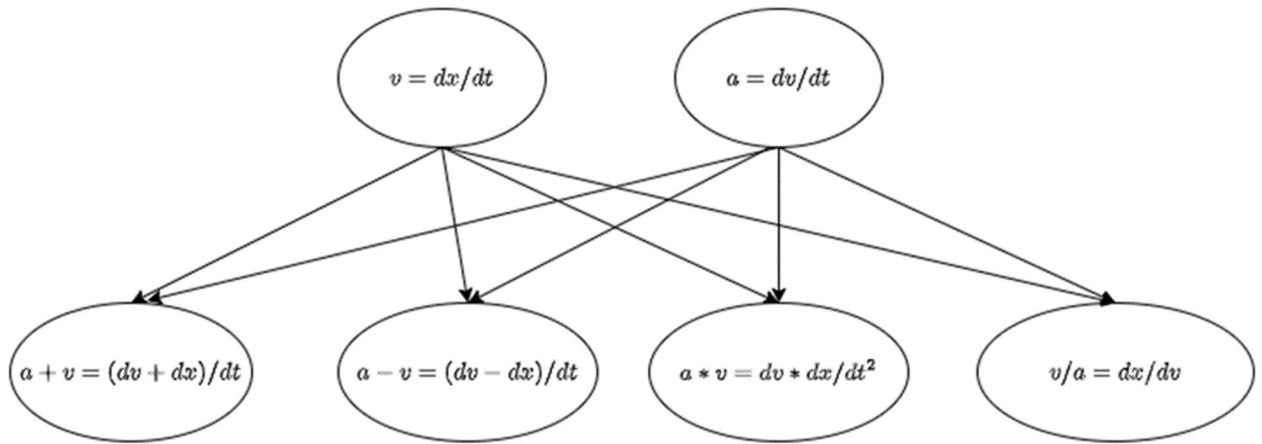


Figure 34: Example of minimal  $G_{res}$  graph

## 4.2.4 Synthesis Formulation

**Definition 4.1.** Given a set of binding operations,  $\mathbf{O}$ , and a set of subject formulas,  $\mathbf{S}$ , the operation of mathematical formula synthesis is defined as:

$$\text{Synthesis}(\mathbf{O}, \mathbf{S}) := \left\{ b(s_0, s_1, \dots, s_k) \mid \begin{array}{l} s_1, \dots, s_k \in \mathbf{S}, b \in \mathbf{O}, k \in \{0, 1, \dots, m_b\} \\ \text{and } k \leq |\mathbf{S}| \end{array} \right\} \quad (4.2.5)$$

where:

$m_b \in \mathbb{N}$ : maximum number of accepted arguments for operation  $b$

■

Domain definition is omitted, as it depends on the undefined set of binding operations,  $O$ .

Synthesis, as defined above, acts as the main operation in the synthesis framework, by constructing each level in the  $G_{res}$  graph. In the next subsections, we dive into a discussion, where we try to capture any risen issue or incompatibility than may occur.

## 4.2.5 Indeterminacy Checks & Imposed Conditions

The validity of each byproduct, i.e., each node above the first level, is unknown, but can partially be validated by performing deterministic mathematical checks that seek common indeterminacies. We tie out the potential indeterminable cases according to the mathematical operations that bind the formulas and may pose restrictions – e.g., during a division  $\mathbf{a}/\mathbf{b}$ , the denominator  $\mathbf{b}$  should always satisfy the relation  $\mathbf{b} \neq \mathbf{0}$ ; during a matrix multiplication  $\mathbf{A} \times \mathbf{B}$  of matrices  $\mathbf{A}_{m_1 \times n_1}$  and  $\mathbf{B}_{m_2 \times n_2}$ , the relation  $n_1 = m_2$  should always hold. These restrictions are checked and then generated as complementary conditions for a formula to be valid. Thus, we end up characterizing each unknown formula with validity – a categorical feature, which responds to three values: valid, conditional, and invalid. The latter case of invalidity corresponds to the occasion where at least one of the generated

conditional relations cannot be satisfied. During the creation of a new formula, each preexisting condition of the operand formulas is inherited by the byproduct, alongside with any newly emerged condition.

In addition to the restrictions imposed by the utilized mathematical operations, we also consider any restriction that might be forced or assumed by the operating environment. E.g., consider the Newtonian laws of motion, which can also be considered as the foundation for classical mechanics. Omitting the analysis of rotational movement, the dominant variables met are those of *time* ( $t$ ), *mass* ( $m$ ), *position* and *movement* ( $x$ ), *velocity* ( $v$ ), *acceleration* ( $a$ ), *momentum* ( $p$ ), *work* ( $W$ ) and *kinetic energy* ( $E$ ). One of the main restrictions imposed by the operating environment is the distinction between scalars and vectors. Newtonian physics define  $x, v, a$  and  $p$  as vector entities, usually denoted as  $\vec{x}, \hat{x}$  or  $\boldsymbol{x}$ , and the entities of  $t, m, W$  and  $E$  as scalars. Consequently, strict vector or scalar operators (e.g., outer product, see also *hierarchy of calculations*) are unable to be applied on entities differently distinct. Another common source of imposed restrictions is the defined domain of values for each entity. Under the presented scope,  $t$  and  $m$  are always non-negative, a fact that also leads to  $E = \frac{1}{2}mv^2$  being strictly non-negative, as  $v^2$  follows the same behavior. This generates the conditions:

$$[m \geq 0, t \geq 0, E \geq 0] \tag{4.2.6}$$

Typically, the Newtonian analysis of motion is performed under the scope of  $\mathbb{R}$ , therefore complex or imaginary elements are not considered, resulting in expressions similar to  $\sqrt{-m}$  being rejected as invalid.

Another imposed condition specific to the current domain, which falls in a unique category on its own, is the law of inertia; it states that a body of mass  $m$  remains at rest or in a uniform motion unless a force is applied upon it. Furthermore, the mass  $m$  of a body is considered constant to time. The outcomes of operations that are non-variable (e.g.,  $\frac{dm}{dt} = 0$ ) are not considered during the next level synthesis, as they were observed not to propagate any meaningful information.

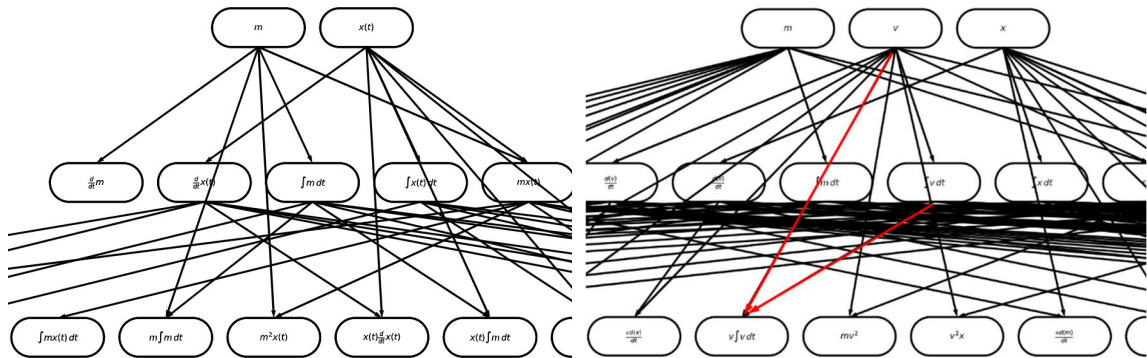


Figure 35: G-res with (right) and without (left) intra-level synthesis

Based on the discussion above, we provide the following definition of byproduct invalidity.

**Definition 4.2.** A synthesis byproduct,  $y$ , is considered invalid if at least one of the following statements holds:

- (i)  $y \in \Phi$ , where  $\Phi$  is the set of constants
- (ii)  $y$  is accompanied by contradicting conditions

(iii) The synthesis operation of  $y$  and its arguments are incompatible, due to strict domain of operation.

■

To provide an analytical example of invalid expression, we consider the following setup:

**Subject**

1.  $y = \sqrt{10 - x}$
2.  $z = \sqrt{x - 10}$

**Binding operators**

1. Division (/)

(0):  $y = \sqrt{10 - x}$ , (1):  $z = \sqrt{x - 10}$

	<b>R1</b>	<b>R2</b>	<b>Synthesis</b>	<b>Validity</b>	<b>Conditions</b>
<b>Division</b>	$y = \sqrt{10 - x}$	$z = \sqrt{x - 10}$	$\frac{y}{z} = \frac{\sqrt{10 - x}}{\sqrt{x - 10}}$	Invalid	$[dt \neq 0,$ $10 - x \geq 0,$ $x - 10 \leq 0]$

(0):  $z = \sqrt{x - 10}$ , (1):  $y = \sqrt{10 - x}$

	<b>R1</b>	<b>R2</b>	<b>Synthesis</b>	<b>Validity</b>	<b>Conditions</b>
<b>Division</b>	$z = \sqrt{x - 10}$	$y = \sqrt{10 - x}$	$\frac{z}{y} = \frac{\sqrt{x - 10}}{\sqrt{10 - x}}$	Invalid	$[dt \neq 0,$ $10 - x \geq 0,$ $x - 10 \leq 0]$

Table 13: Example of invalidity occurrence

Due to the contradicting conditions that are imposed by the square root function (while performing the corresponding indeterminacy check, we omit the complex space  $C$

and its derivatives), the synthesis result produced by division is rendered invalid. Invalidity was immediately spotted as this is an artificial setup constructed for this purpose. In our chosen setup, invalidity tends to be uncommon, occurring deeper in the analysis graph  $G_{res}$ . More examples can be found below, in the *Hierarchy of Calculations* chapter.

## 4.2.6 Termination Criteria

Finally, we address the issue of termination criteria. While the synthesis process is by definition deterministic, its termination occurs only if there are contradicting conditional relations for all formula leaves of the latest tree level, rendering them invalid. This dependency on the binding operators in many cases may result in infinite iterations. Thus, we use a form of early stopping, terminating the iterations if, per say, for two subsequent levels, there are no findings valid or semantically meaningful in the current setup.

**Definition 4.3.** *The synthesis process terminates when one of the following cases are satisfied:*

- (i) *A predefined depth of synthesis layers has been reached.*
- (ii) *No semantically valid synthesis byproduct has been formed in two subsequent levels.*

■



## 4.2.7 Space Complexity

Below, we provide a basic theorem and derived corollaries on the space complexity of each level of the synthesis analysis.

As the general case, we consider intra-level operations in  $G_{res}$  available, and the binding operators are not assumed to satisfy the commutative property. In the case where all the available non-unary binding operators are commutative, the number of byproducts is computed through the combination of subjects,  $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ , instead of using their number of permutations,  $P(n, k) = \frac{n!}{(n-k)!}$ .

**Theorem 4.1.** *Let  $S$  be the set of initial subjects, with  $m_0 = |S|$ . Let also  $\mathbf{O}_t \subseteq \mathbf{O}$  be the set of operators accepting  $t$  arguments, with  $b_t = |\mathbf{O}_t|$  and  $t \in \{1, 2, \dots, k\} \subset \mathbb{N}$ ,  $k \leq m_0$ . The number of synthesis outcomes that are constructed through the synthesis operation in (4.2.5) is equal to:*

$$m_1 = \sum_{t=1}^k b_t * P(m_0, 1) - r_1 \quad (4.2.7)$$

Where:

$$P(n, k) = \frac{n!}{(n-k)!} \quad (4.2.8)$$

*Is the set of permutations per  $k$  in a set of  $n$  elements. The term  $r_1$  corresponds to a regularization term towards towards redundant byproducts; either due to the*

commutative property being effective, constant byproduct, or incompatibility between binding operator and subjects.

**Proof 4.1.** For each  $o \in O_t$ , the number of argument permutations  $o$  can operate on is by definition:

$$P(m_0, t) = \frac{m_0!}{(m_0 - t)!} \quad (4.2.9)$$

As there are  $b_t$  such operators, the total number of byproducts will be:

$$b_t * P(m_0, t) = b_t * \frac{m_0!}{(m_0 - t)!} \quad (4.2.10)$$

The sizes of available argument sets is in the worst case:  $\{1, 2, \dots, k\} \subset \mathbb{N}$ , with  $k \leq m_0$ . Therefore, the overall byproducts incorporate the outcomes of each operator and arguments combination, minus the redundant or declined elements, whose number is expressed as  $r_1$ . Thereafter, the final enumeration of byproducts at level-1 of  $G_{res}$  is:

$$\begin{aligned} m_1 &= (b_1 * P(m_0, 1) + \dots + b_k * P(m_0, k)) - r_1 \\ &= \sum_{t=1}^k b_t * P(m_0, t) - r_1 \end{aligned} \quad (4.2.11)$$

■

**Corollary 4.1.1.** *The space complexity of  $m_1$  is bounded by:*

$$O\left(\sum_{t=1}^k b_t * P(m_0, t)\right) \quad (4.2.12)$$

**Proof 4.1.1.** Emerges naturally as the worst case scenario for number of nodes in level-1 of  $G_{res}$  occurs when  $r_1 = 0$ .

■

**Corollary 4.1.2.** *The number of nodes in the  $i$ -th level of  $G_{res}$  is:*

$$m_i = \sum_{t=1}^k b_t * P\left(\sum_{j=0}^{i-1} m_j, 1\right) - \sum_{j=1}^{i-1} m_j - r_i \quad (4.2.13)$$

**Proof 4.1.2.** The proof reasoning is identical to the one in in *Theorem 4.1*. In the general case, where intra-level operations are enabled, the available synthesis subjects include the nodes of every level of  $G_{res}$  prior to level- $i$ , which sum up to:  $m_0 + \dots m_{i-1}$ .

To avoid redundant nodes, elements that exist in previous levels are omitted. Therefore, the regularization term becomes:  $m_0 + \dots m_{i-1} + r_i$ . The term  $r_i$  corresponds to  $r_1$  mentioned above, for the level- $i$ . The indicator  $k$  also adapts to be:  $k \leq m_0 + \dots m_{i-1}$

The overall enumeration of elements in level- $i$  becomes:

$$\begin{aligned}
m_i &= \sum_{t=1}^k b_t * P(m_0 + \dots m_{i-1}, 1) - (m_0 + \dots m_{i-1} + r_i) = \\
&= \sum_{t=1}^k b_t * P\left(\sum_{j=0}^{i-1} m_j, 1\right) - \sum_{j=1}^{i-1} m_j - r_i
\end{aligned}
\tag{4.2.14}$$

■

**Corollary 4.1.3.** *The space complexity of  $m_i$  is bounded by:*

$$O\left(\sum_{t=1}^k b_t * P\left(\sum_{j=0}^{i-1} m_j, 1\right)\right)
\tag{4.2.15}$$

**Proof 4.1.3.** Identical to *Corollary 1.2*.

■

## 4.2.8 Hierarchy of Calculations

A problem that arises with the chained operations is the incompatibility of arguments when applying operations that do not satisfy the assumed properties. A representative example is matrix multiplication. As seen in the semantic perspective's introduction, for  $\mathbf{A} \times \mathbf{B}$  of matrices  $\mathbf{A}_{m_1 \times n_1}$  and  $\mathbf{B}_{m_2 \times n_2}$ , the relation  $\mathbf{n}_1 = \mathbf{m}_2$  should hold by definition. In the event that the relation holds,  $\mathbf{B} \times \mathbf{A}$  cannot be valid unless  $\mathbf{n}_2 = \mathbf{m}_1$ , i.e., the commutative property does not hold, both under the validity's perspective, as well as the calculation's result. In our current setup, the application of such an operation imposes another restriction, but given fixed dimensionality of matrices, this would result in incompatibility.

Another common case is found in operations that map the result to a different plane. E.g., in the vector domain, the common operations of inner product ( $\mathbf{a} \cdot \mathbf{b} \mid \langle \mathbf{a}, \mathbf{b} \rangle$ ) and outer product ( $\mathbf{a} \otimes \mathbf{b}$ ) both receive vector arguments, but the former outputs to  $\mathbb{R}$ , while the later outputs to the vector space. Depending on the other components of the synthesis expansion, the order of execution when using such binding operators may greatly affect the emerging cases of invalidity; e.g.,  $\langle \vec{\mathbf{a}}, \vec{\mathbf{b}} \rangle$  outputs a scalar and its further synthesis using inner or outer product results results in invalidity.

That above cases highlight that, in a partial search of potential valid expressions, the order of calculations that results to the longest path should be carefully selected, so as to develop the synthesis to its full potential. Usually, this corresponds to the longest path in the corresponding knowledge graph, from roots to leaf. This is not a case that arises in the proposed method though, as every desired combination of expressions is considered due to its grid-search base. The main reason we adopt such an exhaustive mentality derives from the need to inspect any possible byproduct of synthesis, as no other validity signs or semantic indications are used.

An explanatory graph that showcases the completeness of the methodology can be seen in the Appendix, section A.4.2

## 4.2.9 Methodology

Combined together, the presented notions structure the methodology for the exploratory synthesis of mathematical formulas, which can be seen in figure 36 in the form of pseudocode and in figure 37 as a flowchart.

---

**Algorithm 5** Formula Synthesis Algorithm

---

27: **define** subject formulas  $S$

28: **define** binding operators  $B$

29: **define** Indeterminacy Checks ( $IC$ ) derived from the binding operators

30: **define** the maximum number of analysis levels  $ml$

---

31:  $byproducts \leftarrow \{\emptyset\}$

32:  $conditions \leftarrow \{\emptyset\}$

33:  $status \leftarrow \{\emptyset\}$

34:  $level \leftarrow 1$

35: **for** every subject  $s$  in  $S$  **do**

36:      $validity, restrictions \leftarrow IC(s)$

37:     append restrictions into conditions

38:     append  $validity$  into  $status$

39: **while**  $(\exists x: x \in status \wedge x \neq invalid) \wedge (level < ml)$  **do**

40:      $status \leftarrow \{\emptyset\}$

41:      $byproducts_{level} \leftarrow \{\emptyset\}$

42:     **for** every binding operator  $op$  in  $B$  **do**

43:         **for** every permutation  $p_{arg}$  of subjects of a permissible argument list  
          length **do**

44:              $synthesis \leftarrow op(* args)$

45:              $validity, restrictions \leftarrow IC(synthesis)$

---

46:               append *synthesis* into *byproducts<sub>level</sub>*  
47:               append *restrictions* into *conditions*  
48:               append *validity* into *status*  
49:               append *byproducts<sub>level</sub>* into *byproducts*  
50:    *level*  $\leftarrow$  *level* + 1  
51: **return** *byproducts, conditions, status*

*Figure 36: Pseudocode describing the synthesis methodology*

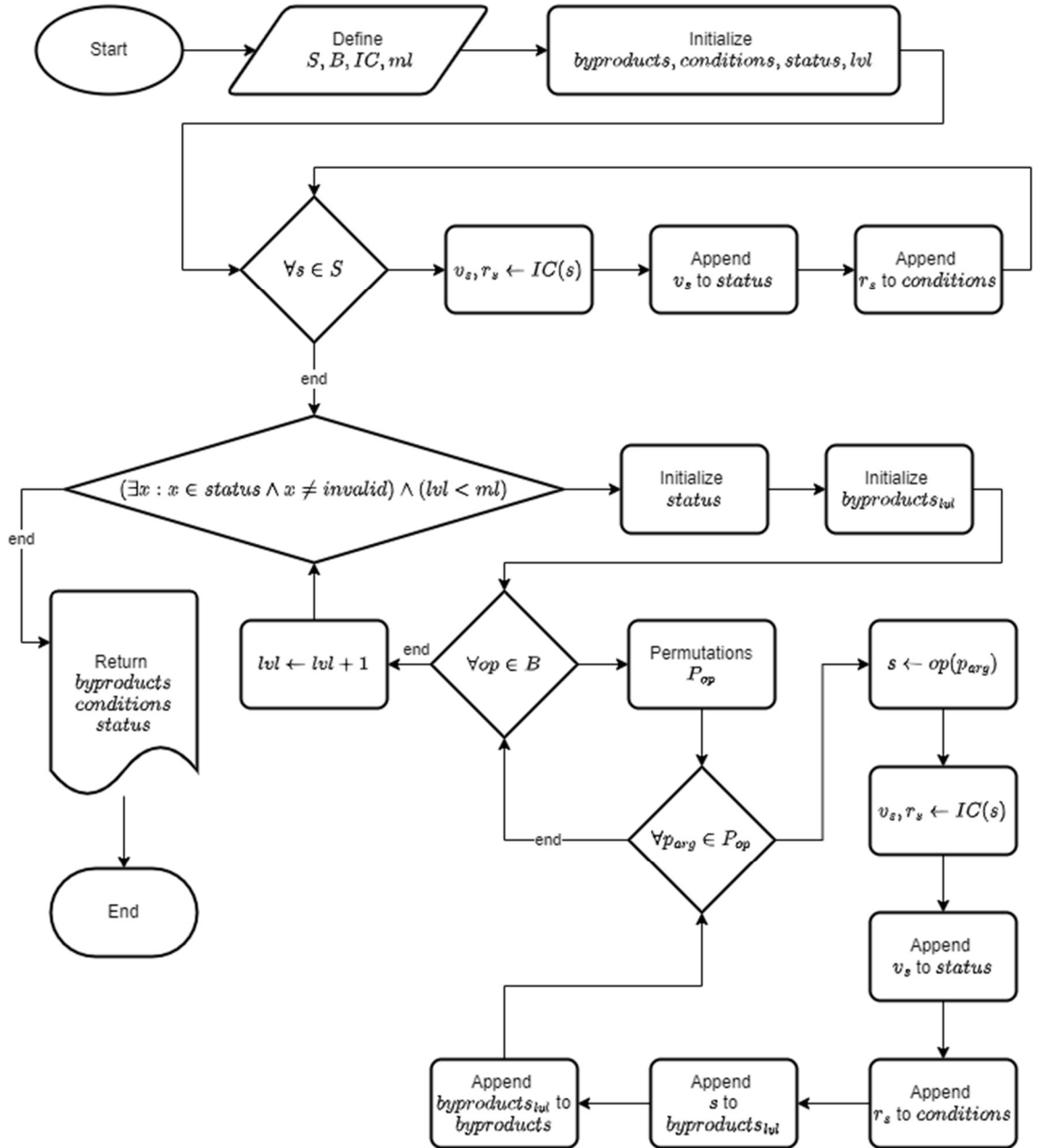


Figure 37: Flowchart of the main process

The methodology requires the definition of the described setup, which sums up to the set of subjects  $S$ , the set of binding operators  $B$ , the indeterminacy checks' mechanism  $IC$ , and the maximum number of analysis levels,  $ml$ . The latter works as a termination criterion, but the methodology is always subject to early termination, in case all the leaf



nodes on the latest level of analysis are syntactically or semantically invalid. Both cases are captured by the condition in *line 13* of the pseudocode, and the corresponding conditional block in the flowchart. Concerning the terms used in the flowchart,  $P_{op}$  symbolizes the set of every permutation  $p_{arg}$  of permissible argument list length per synthesis operator,  $op \in B$ . Similarly,  $s$  corresponds to the product of the synthesis operation, while  $v_s$  and  $r_s$  are the outcomes of  $IC$ , regarding the validity and the restrictions of  $s$ .

#### **4.2.10 Intra-level Synthesis**

A method implemented as described in figures 36 and 37 utilizes only the leaf nodes of each analysis level. This is done to confer more clarity and explainability to the provided visual examples. The incorporation of non-leaf nodes during the synthesis can easily be achieved by enriching the subjects per analysis level right before the synthesis. For a global perspective, the enrichment should include all unique previous byproducts and the initial subjects. For a partial analysis, the enrichment could also be confined to certain levels. The set examined during the termination criterion should still be kept as the leaf nodes.

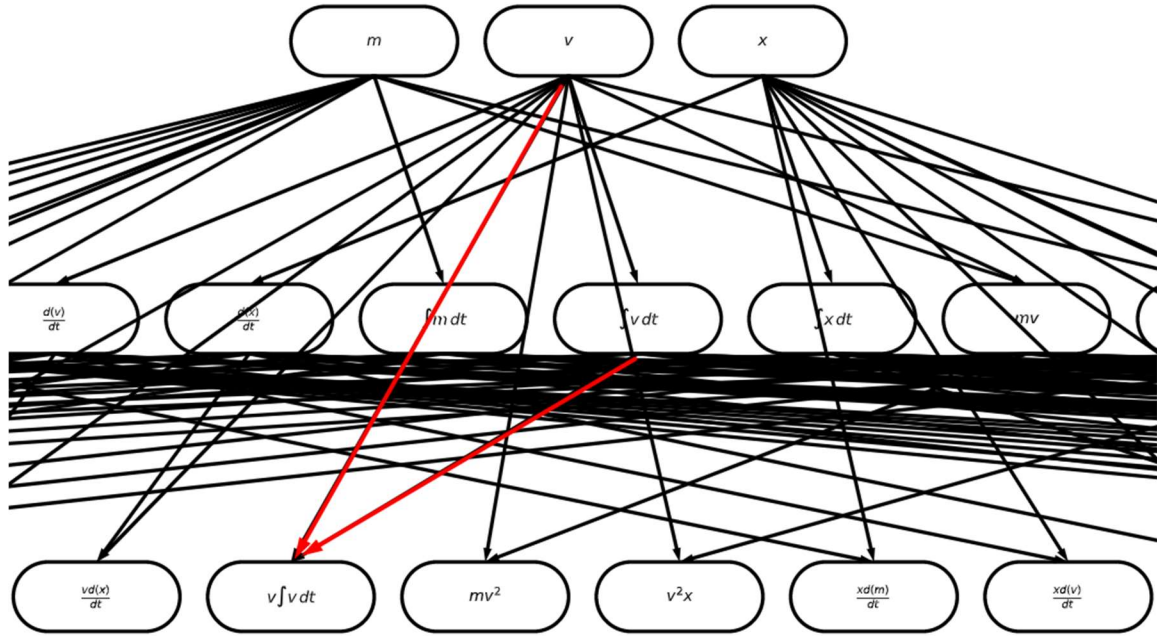


Figure 38: Intra-level synthesis

### 4.2.11 Framework's Proof of Concept

To establish its efficacy, the methodology is applied on the set of subjects  $\{m, x\}$ , utilizing *multiplication* as a binary synthesis operator, as well as two unary synthesis operators, *differentiation through time* and *indefinite integration through time*. Based on Sympy's infrastructure (Meurer et al, 2017)[53],  $x$  is expressed as function of time ( $t$ ), while the  $m$  remains a symbol, invaring to time.

#### Binding Functions:

1. Multiplication (\*)
2. Differentiation through time  $\left(\frac{d}{dt}\right)$
3. Integration through time  $(\int dt)$

#### Subjects / Level 0:

$$\{m, x(t)\}$$

The level 0 of the synthesis analysis is identical to the set of subjects. The level 1 and level 2 outcomes, together with the  $G_{res}$  graph, are provided below:

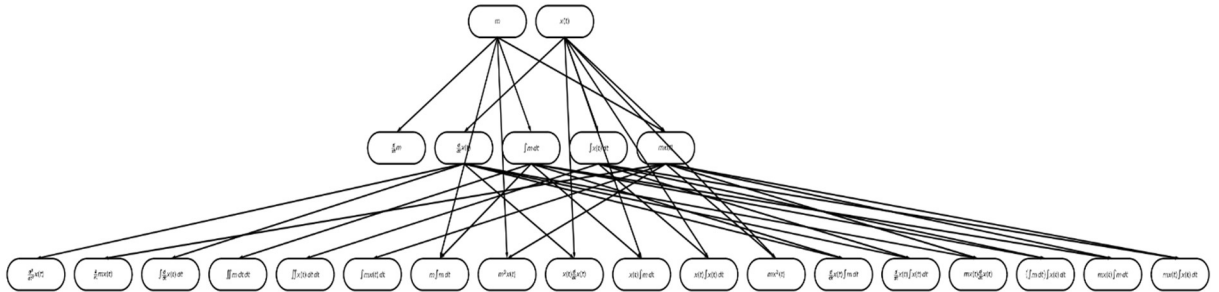
Level 1

$$\left\{ \frac{d}{dt} m, \frac{d}{dt} x(t), \int m dt, \int x(t) dt, mx(t) \right\}$$

Level 2

$$\left\{ \begin{array}{l} \frac{d^2}{dt^2} x(t), \frac{d}{dt} mx(t), \int m \frac{d}{dt} x(t) dt, \int \int m dt dt, \int \int x(t) dt dt, \int mx(t) dt, m \int m dt, \\ m^2 x(t), x(t) \frac{d}{dt} x(t), x(t) \int m dt, x(t) \int x(t) dt, mx^2(t), \frac{d}{dt} x(t) \int m dt, \\ \frac{d}{dt} x(t) \int x(t) dt, mx(t) \frac{d}{dt} x(t), \left( \int m dt \right) \int x(t) dt, mx(t) \int m dt, mx(t) \int x(t) dt \end{array} \right\}$$

Resulting Graph



At this point, it is possible to incorporate a form of the aforementioned a priori knowledge, and identify matchings with the outcomes of each level. Specifically, given the a priori knowledge  $APK$ , while keeping the synthesis outcomes unoptimized, we get the following matchings:

$$APK = \left\{ x: x(t); v: \frac{d}{dt} x(t); p: mv; a: \frac{d^2}{dt^2} x(t); a: \frac{d}{dt} v; E: \frac{mv^2}{2}; \right\}$$

Level 1 / Substitutions

$$\left\{ \frac{d}{dt} m, v, \int m dt, \int x(t) dt, mx(t) \right\}$$

Level 2 / Substitutions

$$\left\{ \begin{array}{l} \mathbf{a}, \int v dt, \int \int m dt dt, \int \int x dt dt, \int mx dt, \mathbf{p}, m \int m dt, \\ m^2 x, vx, x \int m dt, x \int x dt, mx^2, v \int m dt, v \int x(t) dt, \\ px, \left( \int m dt \right) \int x dt, mx \int m dt, mx \int x(t) dt \end{array} \right\}$$

It is noticeable that, in two levels of analysis, the expressions of acceleration, velocity and momentum are identified; either standalone, or nested into an expression. Synthesizing a level deeper, we also get occurrences of the accumulated force's expression ( $F$ ). As the number of synthesis instances grows, it is hard to be universally displayed, neither as a set nor in the  $G_{res}$ . Therefore, we simply exhibit the leaf to root paths that contributed to the synthesis of the expression, in figure 39. Finally, we expand to the use of binding operators with multiple argument, as shown in figure 40.

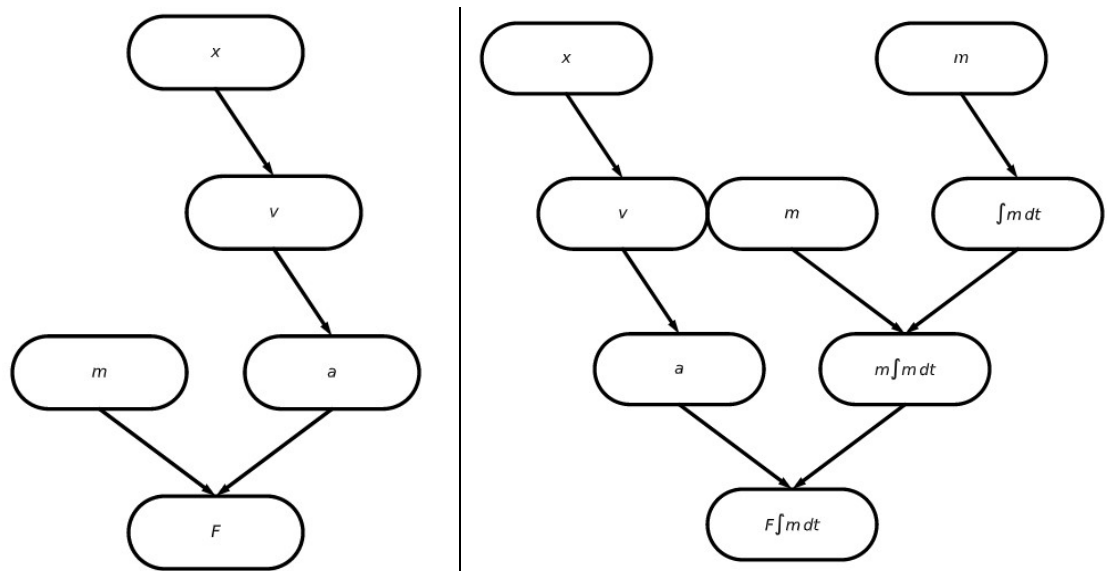


Figure 39: Occurrences of  $F$  relation after substitution in the 3rd level of synthesis

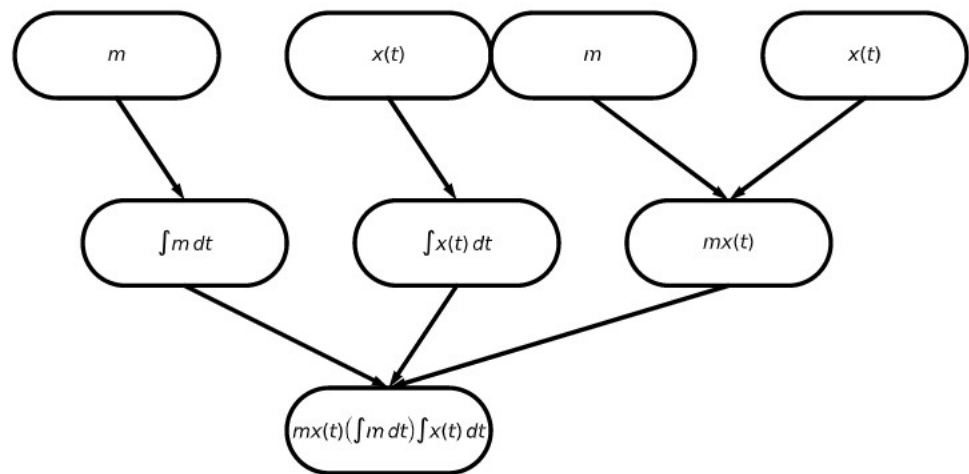


Figure 40: Level-2 outcome of binding operation (\*) with multiple arguments

We conclude the proof of concept walkthrough by discussing on the transition to realistic cases of mathematical formulas synthesis. One of the first obstructions met in a realistic example is the selection of binding operators. While there is provided a substantial arsenal of synthesis operations in mathematics or generally scientific domains, due to the volume of redundant outcomes and the computation cost, the use of semi-random large sets of operators is prohibited. Therefore, the question that arises is which operations indicate

the potential for a fruitful synthesis. Prior, the assumed setup consisted of two simplistic main subjects,  $\{m, x(t)\}$ , binded by one binary (multiple) and two unary synthesis operators  $\{\frac{d}{dt}, \int dt, *\}$ . In the case of Newtonian Motion, it is common knowledge that derivation and integration though time, in addition to multiplication of physical quantities, derive the majority of the established expressions. It could be observed through the patterns of the subject expressions, but for such actions to be universally applied, justification and formal establishing are required.

Another bottleneck met is the dependency on the a priori knowledge used to validate the synthesis results. When using subjects that fall into the state-of-the-art category, we hypothetically get closer to the creation of an expression describing an unstudied phenomenon, but have no explicit means of proving so. The use of a priori knowledge in the presented examples was sufficient for the proof of concept, but it was intentionally and manually selected, therefore it offers no means of generalization.

We believe that both of these issues can be approached by leveraging the modern robustness of probabilistic inference and the increasing amount of openly available data, and therefore concerning ideas and implementations are discussed and analyzed in related sections that follow through the rest of the dissertation. Tracing graphs of such synthetic cases are offered below, where the synthesis levels of analysis is designed using simply the formulas of kinetic energy and accumulated force in Newtonian motion.

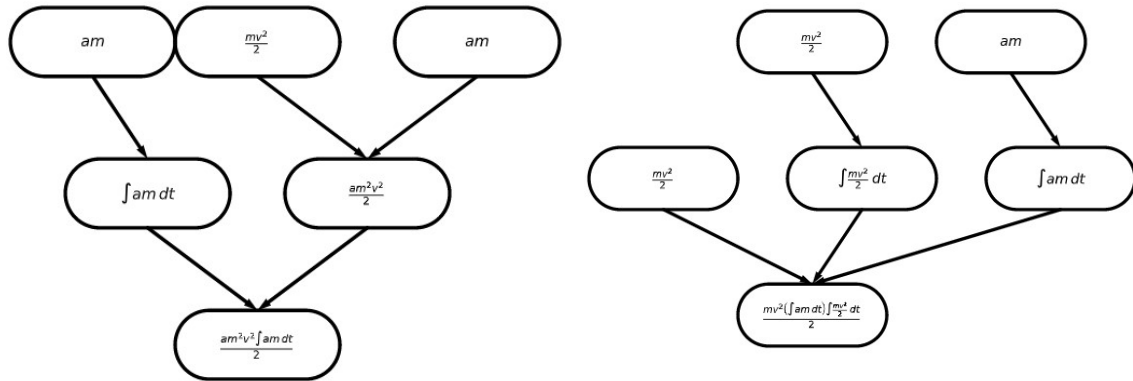


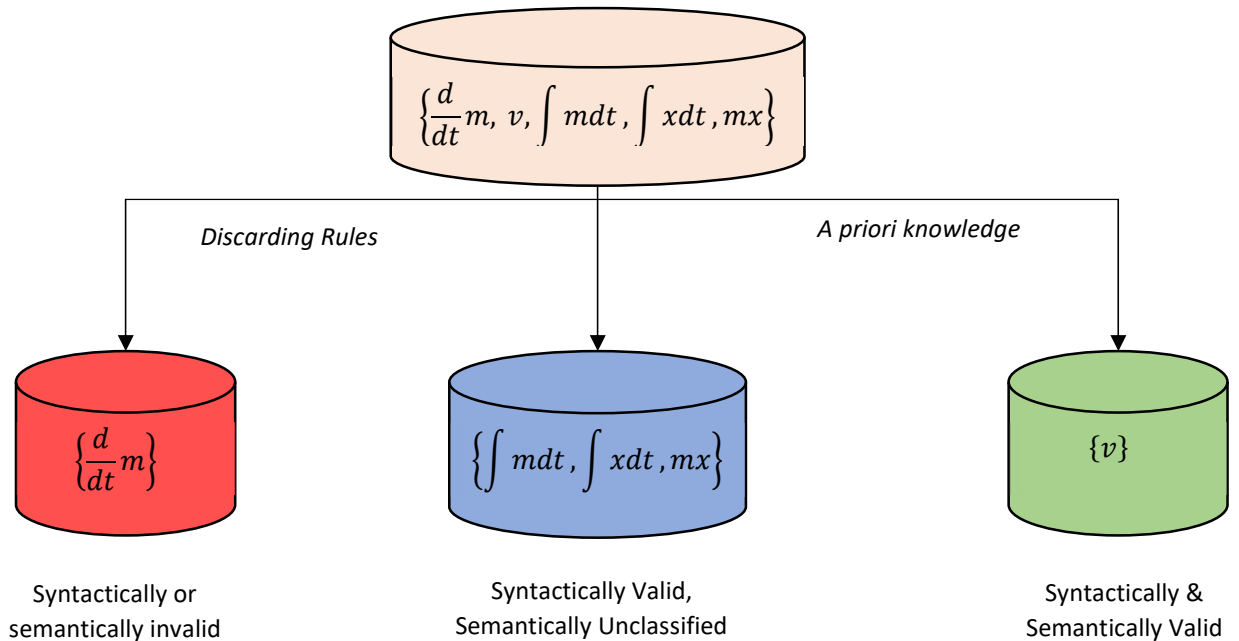
Figure 41: Tracing graphs of complex unclassified nodes

This walkthrough of the synthesis methodology, as presented above, manifests the proof of concept on the core idea introduced by the current work: the assumption that the systematic synthesis of mathematical expressions describing aspects, actions, behaviors or phenomena in a certain scientific field, may result in novel formulas that partially or completely describe phenomena derived or related to the ones assumed. Up to this point, we introduced the motives that drive this research effort, established a formal modelling scheme as our main framework, as well as described and displayed a complete synthesis methodology. With the last complete example on the field of Newtonian Motion – see also appendix section A.4.1 – , using a collection of a priori established expressions, we performed the proof of concept of the initiate, and concluded with a discussion on the bottlenecks that such a task imposes to the general case of state-of-the-art formula subjects. In the next part, we address the synthesis outcome manipulation without assuming any domain-specific a priori knowledge, based on reasoning and probabilistic inference.

## 4.3 A-Priori-Knowledge Acquisition

### 4.3.1 Knowledge Outsourcing

Figure 42: Validation Scheme



According to the aforementioned synthesis framework, the outcome of the synthesis process is a set of expressions, each of which falling to one of the following three categories: syntactically or semantically *invalid*, syntactically and semantically *valid*, and syntactically valid but semantically *unclassified*. The details of the syntactic evaluation of each expression is based on the definition of the existing setup and the imposed indeterminacy checks; its procedures can be automated, and its details were thoroughly presented in the previous section. The challenging aspect of the evaluation is rather the semantic meaning of each formula. While it was also discussed following the syntactic analysis, semantic interpretation of a formula was stated as completely dependent on the



knowledge database, which comprises the available *a priori* knowledge on the specific subject. E.g., it is the existing knowledge of the definition of stable velocity in Newtonian motion,  $v = \frac{dx}{dt}$ , that indicates its classification to the valid subset. Naturally, an *a priori* knowledge base is not provided for unseen expressions, thus, it is trivially non-existent in the case of *state of the art* subject expressions, the de facto case during a scientific document's analysis.

To address the absence of *a priori* knowledge, we turn towards existing collections of expressions and search for features or patterns that will indicate probabilistically the validity of a synthetic outcome. More specifically, the methodology takes advantage of well-established and organized datasets of mathematical formulas on a specific field (in this instance, Classical Mechanics & Newtonian Motion) to obtain the means of disambiguating the semantic validity of the synthesis outcomes. Ultimately, by leveraging outsourced mathematical formula data, we convert the problem at hand into an extrapolation of a machine learning; the reasoning and the techniques used are thoroughly presented in the next subsections. The process is by default uncertain and may resemble “guessing based on observations”, but it is an inevitable flaw, as by definition the described task cannot provide labeling or any form of validation of the results.

As a convention, the probabilistic values assigned on each outcome will represent from now on a *confidence score*, indicating its potential to be semantically valid.

### 4.3.2 Dataset

Dealing with a newly defined and specific task, there exist no predefined benchmark datasets that could be explicitly utilized; thus, a brief survey was performed on fields and tasks that deal with typed mathematical formulas. We chose to proceed with the *Feynman Symbolic Regression Database* dataset (Max Tegmark, MIT) [76], also cited and introduced by (Udrescu & Tegmark) [75]. The Feynman dataset consists of 120 labeled pairs for symbolic regression, where the input corresponds to samples from the value domain of 120 functions, while the label corresponds to the symbolic representation of the function itself. The functions constitute established formulas from various fields of physics and are derived from the *Feynman Lectures on Physics* collection (Feynman et al, 1963) [7][8][9]. As the task at hand does not involve any means of symbolic regression but rather deals explicitly with the symbolic formula, only the labels were extracted and used.

Another notable candidate dataset is the *im2latex-100k* (Kanervisto, 2016) [80], introduced in the work of (Deng, Kanervisto et al, 2016) [81]. It comprises 100 thousand pairs of images and LaTeX formulas, aiming at the task of identifying and producing the LaTeX string representation of formulas using their image format. Similarly to aforementioned dataset, only the labels are taken into consideration. We should also highlight that the use of content outside of the classical mechanics' domain is crucial, as it helps analyze the behavior of the methodology, when trained on a different subject than the one at hand, i.e., during the case of subject extrapolation (partial or complete).

### 4.3.3 Differentiation from Symbolic Regression

At this point we should note the similarities as well as the differentiations of the symbolic regression task and the proposed methodology. In the general case of symbolic regression, given a set of elements,  $(x_1, \dots, x_n, y)$ , paired with the observed resulting value  $y$  if these elements were provided to a hypothesized function, we assume this function as  $f$ , i.e.,  $f(x_1, \dots, x_n) = y$ ; then, examine a usually vast, but carefully restrained set of symbolic terms through regression inspired techniques, aiming to reconstruct the symbolic representation of function  $f$ . Examples of symbolic regression approaches that follow this, or similar assignment, can be seen in the highly acclaimed works of (Udrescu & Tegmark) [75], (Schmidt & Lipson, 2009) [82] and (Petersen et al, 2020) [83], among others. A symbolic regression benchmark database containing benchmarked approaches and datasets is maintained by William La Cava and the Computation Health Informatics Program at Harvard Medical School, found in [84].

In contrast, the synthesis technique used in this dissertation does not search a predefined space of symbolic terms. Our search space is dynamically created, by applying synthesis operations on the chosen subjects, which are usually minimal, accounting from two to five. Additionally, the current methodology does not leverage any labeled data, but rather focuses on finding patterns of the a priori knowledge elements, specifically on the synthesis operator usage. The end result is also different in nature, as the output of our system is a score assignment to each synthesis outcome, expressing their estimated “potential” value. The main differentiation though can be spotted on the desired outcome; symbolic regression, in its general form, seeks the symbolic representation of a function whose domain field, value range and semantics are partially known, while in our case, we

deal with symbolic representations whose semantic meaning is unknown and strive to discover their potential in being valid. Despite the differences, the common ground of utilizing symbolic representation of mathematical formulas and the search for an optimal synthesis of terms is indisputable; most likely, our task can get complemented by symbolic regression-inspired ideas, a case which we are currently examining.

### 4.3.4 Graph Representation of Formulas

Being SymPy objects, the outcomes of the synthesis methodology display a symbolic representation. This implies a textual form of operators and arguments – referred as *func* and *args* in SymPy’s documentation –, following a particular set of unrestrictive conventions – e.g., the expression  $\frac{x}{y-\lambda}$  is encoded as  $Mul(x, Pow(Add(y, Mul(-1, \lambda)), -1))$ ; assuming  $x, y, \lambda$  are registered as symbols and -1 as an integer, division is expressed as multiplication by a term at the power of -1 and subtraction as the addition with a term multiplied by -1. We can easily convert to a string, LaTeX or MathML (Ausbrooks et al, 2014) [41], but the expression directly corresponds to a tree graph:

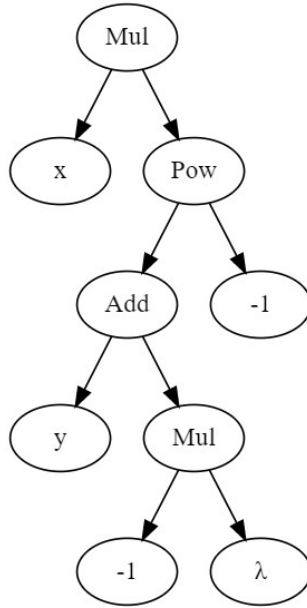


Figure 43: Tree representation of  $x/(y-\lambda)$ .

The tree as a representation provides a hierarchical structure that lifts the need of parentheses met in symbolic string formats and constitutes to a richer dimensionality input to learning techniques. A similar discussion can be found in the acclaimed work of (Lample & Scharon) [85], where the authors genuinely resort in expressing the tree structure as a sequence in prefix notation, e.g.,  $[* x^{\wedge} + y * -1 \lambda - 1]$ , which also avoids parentheses and is shorter in length, then use Attention-based Transformers as their inference techniques on their described task of reasoning on the calculations of the symbolic expressions.

In the presented methodology, we adopt the graph representation as is, and utilize modern architectures of Graph Neural Networks as our inference technique. To do so, for each formula in the acquired datasets we extract the set of nodes, the set of edges and the adjacency list of edges, encapsulating the connectivity and the topology of the graph, as well as rendering the data into a format compatible with conventional graph neural network architectures. For the time being, we keep the connectivity of the nodes unbiased, therefore no edge weights are used. Our default setup includes one node feature, which is the type

of the outer operation of the formula – e.g., the feature of the node representing  $Mul(x, Pow(Add(y, Mul(-1, \lambda)), -1))$  would correspond to  $[Mul]$ , the feature of the node representing  $Pow(Add(y, Mul(-1, \lambda)), -1)$  would correspond to  $[Pow]$ , etc. (see SymPy’s symbolic representation). The only graph feature incorporated is the type of the outermost operation of the symbolic representation of the whole formula. – e.g., the feature of the graph representing the formula  $Mul(x, Pow(Add(y, Mul(-1, \lambda)), -1))$  would correspond to  $[Mul]$ .

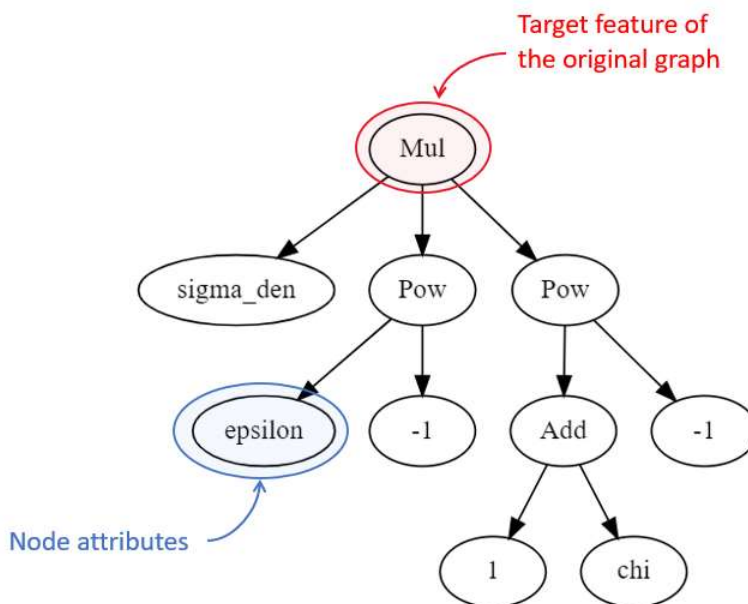


Figure 44: Graph attribute details

### 4.3.5 Description of the Graph Neural Network model

#### (Graph Prediction)

Operating on formula graphs as a whole, we utilize a graph neural network architecture containing a readout layer for a unified graph embedding, as seen in the work of (Xu et al, 2019) [86], also followed by the PyTorch Geometric related article on the

graph classification task through GNNs (Fey & Lenssen, 2019) [87]. The network comprises three graph-adapted convolutional layers, inspired by (Kipf & Welling, 2017) [88], enriched with skip-connections, and relu activation. The convolutions are followed by the averaging readout layer and a dropout normalization, ending in a dense layer that produces the predicted logits.

The number of classes is equal to the unique binding operators identified in the provided data, which for the Feynman dataset accounts for 10 classes. Batch size is 32. Node, edge and graph features, together with the edge weights are described in the Formula Graph Representation subsection above. Training is performed through the typical Adam variation of Gradient Descent (Kingma and Ba, 2014)[64], on regular cross-entropy loss.

### **4.3.6 Synthesis Prediction on Disjoint Graphs**

In subsection (X.), we stated our objective to be the assignment of a confidence score on the potential of all the synthesis results. To achieve this, firstly we make the process of confidence assignment more manageable, by addressing the task of confidence score assignment to every available binding operator, for a given set of formulas. Secondly, we utilize the described preprocessing scheme and GNN model on the Feynman dataset, which results in a robust classifier on the task.

The main idea involves around the decomposition of a given formula into its outermost binding operation and its respective arguments, which is a straight implication of reversing the SymPy's symbolic representation. The formulas' graphs that result from the decomposition are included into a single, disjoint graph, while the original outermost

binding operation is used as the graph’s feature and label, similarly to the description of the Formula Graph Representation section.

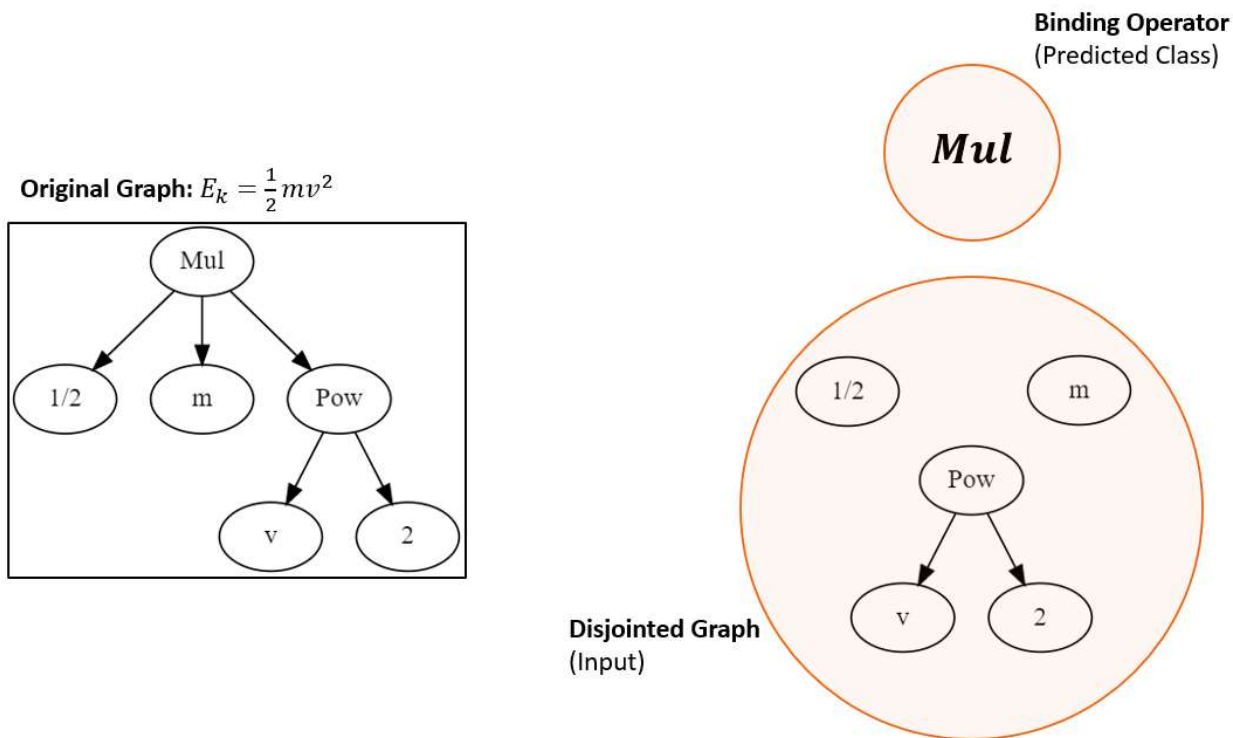


Figure 45: Extraction of disjoint graph and label

### 4.3.7 Results & Discussion

The GNN model was trained on the disjoint adaptation of the Feynman dataset, configured as described in the previous subsection, and trained for 200 epochs with early stopping. Specifically, 10 classes are derived from the Feynman dataset, the last of which bears the name *Function* and includes miscellaneous or non-major types of functions – e.g.,  $\arcsin()$  etc. It is a straight normalization task, with the only abnormality observed being the imbalance class distribution, as it can be seen in table 14, which the model seems to overcome successfully (excluding the *Add* class, which tends to be interpreted as *Mul*).



Train Accuracy			Test Accuracy		
0.9531			0.9063		
	Ground Truth	Predictions		Ground Truth	Predictions
Mul	56	56	Mul	28	31
Pow	3	6	Pow	3	1
Add	3		Add	1	
Function	2	2			

Table 14: Imbalanced class distribution & predictions analytics

To reach an estimation of confidence, predicted logits are normalized through a min-max normalization scheme to provide a value in the [0,1] domain for each class. An example of the estimated confidence for a random sample from the test set is provided in table 15:

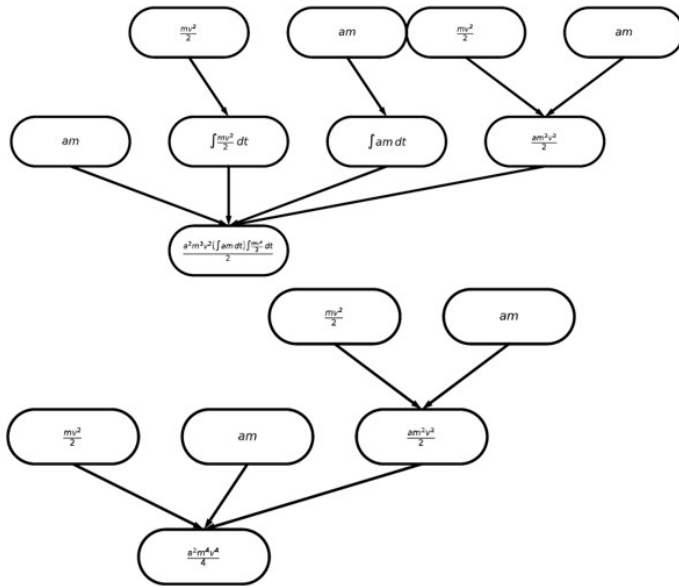
#### CLASS PROBABILITIES

<b>Add</b>	0.7782861590385437	<b>Exp</b>	0.05868188664317131
<b>Atom</b>	0.08954363316297531	<b>Log</b>	0.07360976189374924
<b>Mul</b>	1.0	<b>Sin</b>	0.05607523396611214
<b>Pow</b>	0.5108705759048462	<b>Tanh</b>	0.09291713684797287
<b>Cos</b>	0.07826695591211319	<b>Function</b>	0.0

Table 15: Confidence score through GNN logits

We get a 5-fold cross-validation accuracy of 0.9368 on the Feynman dataset.

We conclude the results section by providing a sample on the synthesis framework experimentation. Using the unsimplified expressions:  $\frac{a^2 * m^3 * v^2 (\int a * m dt) \int \frac{m}{2} dt}{2}$  and  $\frac{a^2 * m^4 * v^4}{4}$ , we get an initial confidence score for each of the binding operation classes in our model.



Class Probabilities	
Add	0.8571221232414246
Atom	0.24663984775543213
Mul	1.0
Pow	0.704578697681427
Cos	0.24490845203399658
Exp	0.22302459180355072
Log	0.2378426343202591
Sin	0.22760023176670074
Tanh	0.25284484028816223
Function	0.0

Figure 46: Formulas derived from the synthesis framework & their predicted class probabilities

As seen, the confidence scores are assigned to the binding operators’ set of the dataset, not of the framework. This observation implies that extrapolation might prove unfeasible for unrelated scientific domains. For closely related data though – e.g., the Feynman dataset and our framework’s proof of concept – extrapolation may be successfully achieved, despite the use of different operations. The main cause of this compatibility is the expressiveness we encounter in the mathematical domain, as mathematical operators tend to form derivation trees (we are all familiar with the abstract but generally true statement that all mathematical operations are derived by the 4 basic ones).

### 4.3.8 Self-evaluation through Validity Classification

One of the most prominent concerns of the presented framework is the absence of an evaluation mechanism. The absence is derived by the nature of the data used, as we ideally try to predict the most promising direction during the construction of new

mathematical terms, describing new scientific phenomena; therefore, it is implied by definition that there are no ways of evaluating the resulting syntheses. To address this fundamental issue, we devise an approach that is also centered on the acquired *a priori* knowledge, but aims in reassuring the validity of given formula graphs, which in this instance correspond to the graphs synthesized according to the disjoint graph classification output.

Similarly to the disjoint classification task, the validity reassurance is achieved through deep geometric learning techniques. The architecture of the referred graph neural network is slightly different to the aforementioned – see also [17][18] – but shares the same core of graph convolutional layers, based on [19]. It differentiates itself through two distinct modifications: firstly, it uses entire consistent graphs of single mathematical formulas, labeled as either valid or invalid, while, secondly, the output layer of the model is also modified to perform a binary classification task. Thus, instead of assigning confidence scores to the available binding operators, it aims to predict the validity of its input.

The formula graphs and the corresponding labels used for this task are obtained from both the elements of the Feynman dataset and their synthesis derivatives. Specifically, each of the dataset formulas is broken into subgraphs for each node that shares a connection with the root (which in the disjoint graph constituted the ground-truth synthesis operator). The formulas described by these subgraphs act as the subject terms of the synthesis framework. The binding functions, both unary or of multiple arguments, are set by default as the operations found in the original formula. From this point and onwards, we define and follow the upcoming conjecture concerning the synthesis derivatives:

**Conjecture 1.** *The result of a semantically valid formula with an altered root binding operator is considered a semantically invalid formula.*

■

There are two reasons that drive us to this conjecture. The first is the statistical observations on the resulting formulas during the proof of concept. We notice that, in the general case, a subset of synthetic results derived from the same subject terms, contains *at most* one formula that belongs to the a priori knowledge. However, we are in no position to guarantee that no false negatives will occur, as we possess neither an a priori knowledge, nor an unbiased validation mechanism to verify. What we can assume, though, based on the same statistical observations, is that the number of hypothesized false negatives is insignificant to the training process and will not bias the inference capabilities of the model.

Following the space complexity definitions from 4.2.7, the synthesis process outputs are monitored by the following relation:

$$m_1 = \sum_{t=1}^k b_t * P(m_0, t) - r_1 \quad (4.3.1)$$

Where  $m_1$  corresponds to the number of formulas produced,  $t$  to the number of arguments per binding operator,  $k$  to any set of arguments cap (otherwise set to the number of initial subjects  $m_0$ ),  $b_t$  to the binding operators compatible with  $t$  arguments and  $m_0$  to the number of initial subjects. The term  $r_1$  is a regularization term that removes redundant elements due to commutativity or constants. Through repeating efforts, it is observed that the disproportionality of classes reaches the levels of 9% for positive instances, compared

to 91% for negative instances. This class imbalance was noticed to suppress the effectiveness of the graph convolutional model, a phenomenon regularly spotted with modern GNNs and instances of real-world problems as discussed in (Zhao et al, 2021) [21], where the vast majority exhibits imbalanced classes. We counter the disproportionality by regulating the amount of invalid instances while keeping stable the valid instances to 96 samples and maximizing the total size; the synthetic nature of the data, together with the task formulation, allow for such tweaks in the data distribution, without corrupting the nature of our task (as it may have happened with a real-world dataset, or task). Best results were recorded for a 75% - 25% split, where our graph convolutional model reached a mean test accuracy of  $\sim 0.98$  on a 4-fold cross-validation.

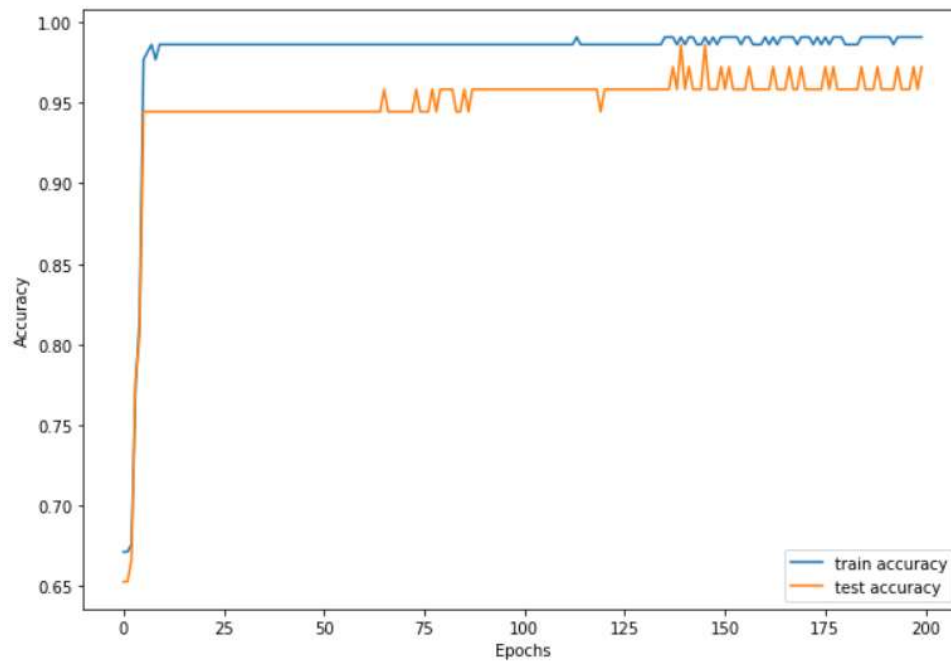


Figure 47: GNN train and test accuracy on the constructed dataset

The self-evaluation procedure concludes with the application of the GNN inference model on the most confident result of the disjoint graph classification outputs. Mathematically, the self evaluation task could be described as:

$$NN_{val}(\operatorname{argmax}(GNN_{op}(G_d))(G_d)) \quad (4.3.2)$$

Where  $GNN_{val}$  refers to the validation graph convolutional model,  $GNN_{op}$  refers to the synthesis operator graph convolutional model, and  $G_d$  is the disjoint graph of the terms under inspection. Its practical use can be seen in cases where the highest predicted confidence scores by  $GNN_{op}$  are close, and the  $GNN_{val}$  classifies highest scoring operator as invalid but the second in order operator as valid. This adds the acceptance margin for such a distinction as another hyperparameter to our method, which we tend to consider in case the confidence scores differ less than 10%, and is set as such intuitively. The overall pipeline of training, inference and self-evaluation can be seen in the workflow diagram in figure 48.

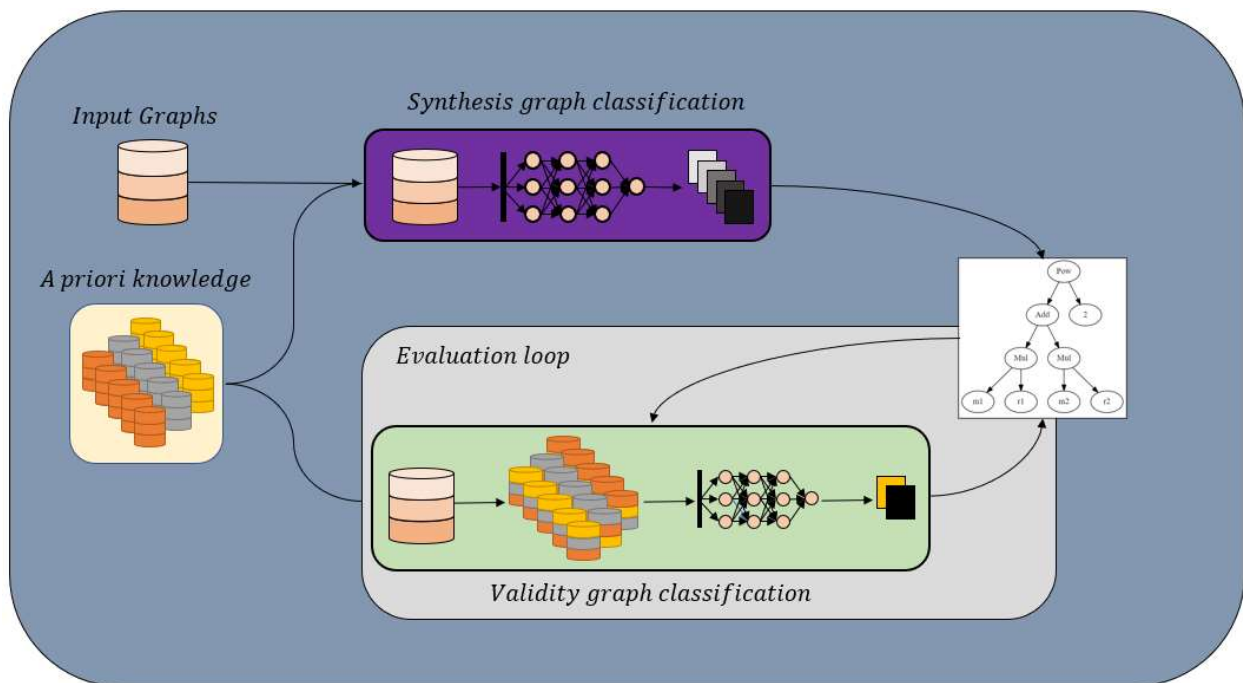


Figure 48: Training, inference & self-evaluation pipeline

## 4.4 Conclusion

In this chapter, we introduce the ambitious task of directed mathematical formula synthesis. We hypothesize the mathematical formulas found in technical documents as metadata, and explore ways of extracting meaningful knowledge on their potential as synthetic components, instead of synthesis byproducts. To do so, we introduce a complete synthesis framework, based on the graph representation of mathematical formulas, that allows for the systematic creation of syntactically correct but semantically unclassified formulas. During the framework introduction, we discuss the limitations that arise, the imposed restrictions, and the dependency on the subject domain’s a priori knowledge. In response to this, we propose a simplistic graph alteration that creates GNN-based graph classification instances, and therefore allows for the use of robust deep learning methods to fill the a priori knowledge gap in the state-of-the-art level. Motivated by the absence of

ground-truth labels, we also develop a self-evaluation mechanism, by leveraging the introduced synthesis framework to create valid and invalid instances. We then train a graph convolutional model identical to the aforementioned on the synthetic instances, that acts as a validity indicator on our initial predictions. We conclude with an extended proof of concept on both the synthesis framework and the graph classification task on the Feynman physics dataset.

Our main motivation is to ignite the interest on the subject and open the path for its development as a distinct task. The presented work is but a mere introduction on the directed formula synthesis. A promising aspect that we have not covered is the interpretability of valid synthesis byproducts. The semantics of the byproduct's components are already known, therefore, the confidence scores could provide an association between the components' sentiment and the implication of the predicted operation to describe the byproduct's interpretation. Additionally, there are numerous other approaches that could be used to provide a confidence score for the validity of synthetic formulas, as well as numerous issues that have not been addressed. Cross-domain synthesis, selection of binding operators, extrapolation and a clean form of evaluation comprise some of the upcoming challenges, to name a few. We are currently working on the development of task mappings and the effect of only partially correlated a priori knowledge bases. We are also experimenting with types of representation beyond the symbolic. During the proposal defense, we exhibited results of projecting symbolic mathematical objects to vector spaces and using unsupervised learning to capture basic mathematical properties. We believe that such an effort can both augment the existing framework's capabilities and offer a path to interpretability. We display our initial in token and expression level results underneath.



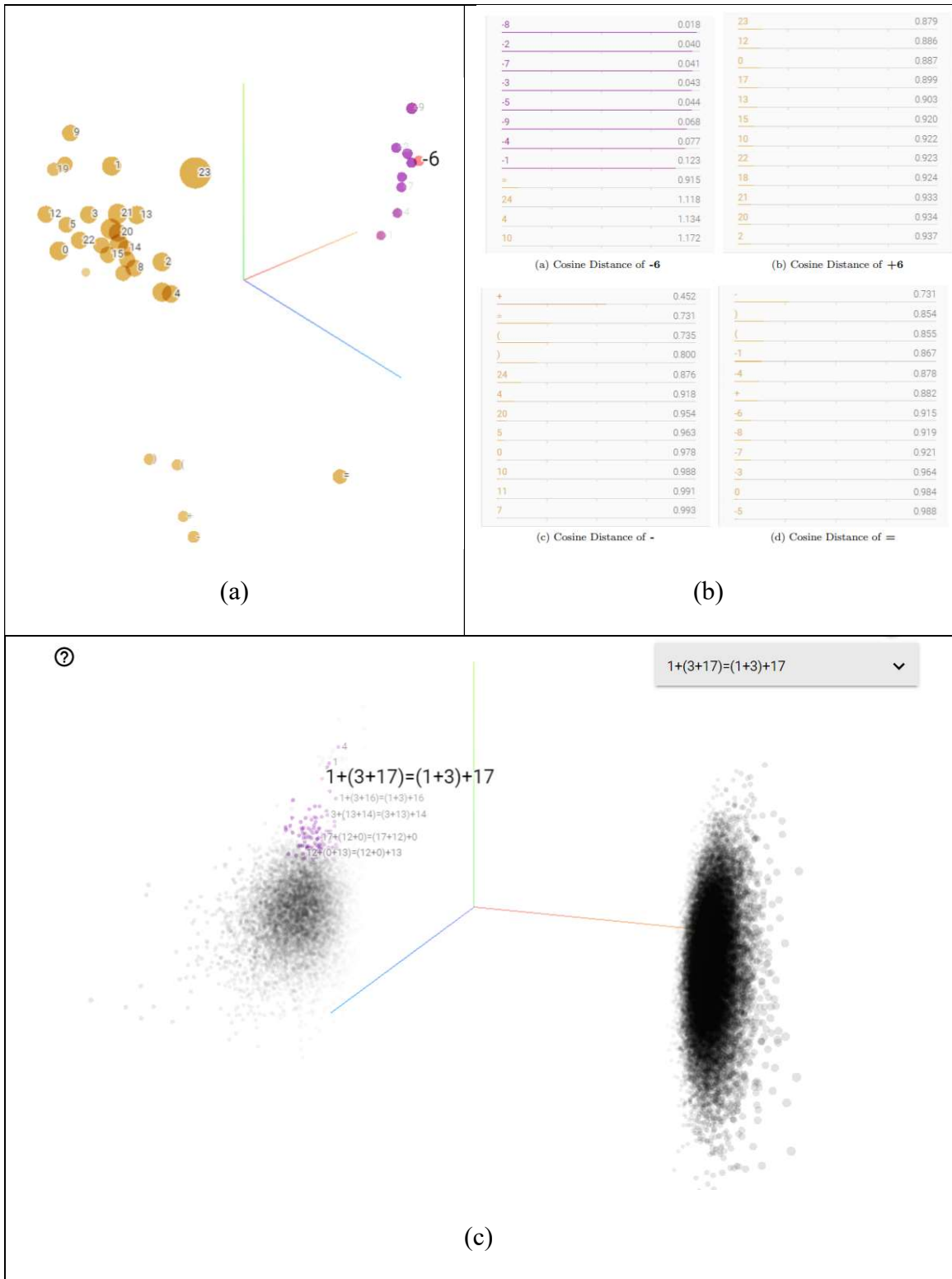


Figure 49: Vector space projections of mathematical tokens

## 5. Epilogue

## 5.1 Conclusion

We began our effort in chapter 1, by realizing the necessity of the automated understanding of technical documents and distinguishing two under-researched but beneficial modalities, the automated comprehension of mathematical formulas and the automated pseudocode extraction. We then proceed in chapter 2 to a thorough investigation of the related literature on both topics.

Based on literature overview observations, we begin our research activity on the automated pseudocode extraction task in chapter 3, by setting the basis of a functional modelling infrastructure for digital diagrams images found in technical documents. We identified the existing flaws and suggested a novel and complete framework that addresses the task. The core functionalities of our methodology are summed up to the decomposition and re-composition of digital circuits under a formal language perspective, generative capabilities on digital circuit images, the annotation mechanism, the graphical representation functions and the specialized data augmentation. Our main contributions on the subject are stated through the following achievements:

- i. mapping of the original task to the image captioning domain,
- ii. creation of two distinct novel datasets of digital circuit images, one on image captioning and one on type classification, as well as their generation mechanisms
- iii. a thorough analysis on the impact of the dataset's characteristics on the image captioning efficacy

iv. realization of existing challenges and four distinct novel methods for their overcoming, based on image partitioning

In parallel, we explore the topic of automated deep understanding of mathematical formulas in chapter 4. We focus on capturing the semantics of mathematical expressions and perform an evaluating survey on the subject's existing literature, which was published in 2021. We then introduce the task of analyzing state-of-the-art mathematical formulas as metadata, specifically by synthesizing existing elements and capitalizing on a priori knowledge. We begin similarly to the pseudocode extraction process, by defining a formal modelling scheme and creating a synthesis framework. We then proceed into using the graph representation of existing established formulas to discover novel mathematical expressions with the highest potential of describing domain aspects. The highlights of our work on mathematical formula synthesis and analysis are the following:

- i. creation of a formal modelling scheme,
- ii. address inconsistencies and adapting to basic mathematical reasoning,
- iii. robust generation of synthetic formula instances based on graph representation of expressions
- iv. simulation of a priori mathematical knowledge and synthesis operation confidence assignment through geometric learning
- v. proof of concept on the Newtonian motion domain
- vi. self-evaluation scheme

In both subjects, the presented methodologies combine traditional reasoning and modern learning, to achieve successful modelling, robust inference capabilities and

knowledge discovery. A bottleneck in both cases appears to be the absence of structured data. During the pseudocode extraction task, we create out of necessity our own synthetic data through the generative framework, but we cannot do the same with established mathematical formulas. The Feynman dataset is a great tool for the proof of concept, but for more scaled experiments, larger resources are needed. Our next immediate step is the creation of structured datasets from real and established mathematical resources, on multiple subjects. Apart from improving the inference capabilities of our existing models and expanding the a priori knowledge base in general, this will allow for experiments on cross-domain ablation studies. Finally, we embrace our discussion on synergistic approaches, by experimenting with cooperative methods between the two fields. Specifically, we are investigating the construction of a digital circuit inspired setup for the synthesis framework, the conversion of the predicted pseudocode of chapter 3 to strict mathematical terms, and its use as input to the synthesis framework.

In conclusion, our greatest aspiration through this work, is to encourage other researchers to engage with the deep understanding of technical documents and contribute with their own work on the several interesting aspects that arise on this domain. The pseudocode extraction task can be expanded from digital circuits to numerous other applications and provide practical and scalable solutions. The mathematical formula synthesis, on the other hand, exhibits an intriguing and ambitious initiate, with tremendous potential for new, directed discoveries. We believe that the presented methodologies can constitute the foundation for further development, both on their respective subjects, as well as other modalities on the deep technical document understanding.

# Publications

Nikolaos Gorgolis, Nikolaos Bourbakis; “Evaluating Methodologies on Deep Understanding of Mathematical Formulas in Technical Documents” in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, November 2021, doi: 10.1109/ICTAI52525.2021.00148

Michail S. Alexiou, Nikolaos Gorgolis, Sukarno Mertoguno, Nikolaos G. Bourbakis; “Deep Understanding of Technical Documents: An Enhancement on Diagrams Understanding”, August 2021 *International Journal of Artificial Intelligence Tools* 30(05):2150027, doi: 10.1142/S0218213021500275

Nikolaos Gorgolis, Nikolaos Bourbakis; “Approaching Pseudocode Captioning of Digital Diagram Images in Technical Documents”, **submitted** in *International Journal of Artificial Intelligence Tools*, 2022

Nikolaos Gorgolis, Nikolaos Bourbakis; “Mathematical Formulas as Metadata: A Synthesis Framework ”, **to be submitted**

# References

- [1] Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. (2001). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In Kremer, S. C.; Kolen, J. F. (eds.). *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press. [ISBN 0-7803-5369-2](#).
- [2] Rajat Raina; Anand Madhavan; Andrew Y. Ng (2009). "Large Scale Deep Unsupervised Learning using Graphics Processors", Computer Science Department, Stanford University, Stanford CA 94305 USA, ICML '09: *Proceedings of the 26th Annual International Conference on Machine Learning*, June 2009, pp. 873–880, <https://doi.org/10.1145/1553374.1553486>
- [3] Alex Krizhevsky; Ilya Sutskever; Geoffrey E. Hinton; "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems 25* (NIPS 2012), pp. 1097-1105
- [4] <https://research.google/people/jeff/>
- [5] arXiv research repository, [arXiv.org e-Print archive](https://arxiv.org)
- [6] arXiv submission rate statistic, [arXiv submission rate statistics | arXiv e-print repository](#)
- [7] arXiv monthly submissions, [Monthly Submissions \(arxiv.org\)](#)

- [8] Zeeshan Ahmed & Thomas Dandekar; “MSL: Facilitating automatic and physical analysis of published scientific literature in PDF format”, 2015, DOI: 10.12688/f1000research.7329.3
- [9] Dominique Antoine; Suzanne Collin; Karl Tombre; “Analysis of Technical Documents: The REDRAW System”, in *Structured Document Image Analysis*, 1992, pp. 385-402, DOI: 10.1007/978-3-642-77281-8\_18
- [10] S. Jiang, J. Hu, C. L. Magee and J. Luo, "Deep Learning for Technical Document Classification" in *IEEE Transactions on Engineering Management*, 2021, doi: 10.1109/TEM.2022.3152216
- [11] Carlos H. Caldas; Lucio Soibelman; “Automating hierarchical document classification for construction management information systems”, *Automation in Construction* 12 (2003) 395 – 406, doi:10.1016/S0926-5805(03)00004-9
- [12] L. Aristodemou and F. Tietze, “The state-of-the-art on Intellectual Property Analytics (IPA): A literature review on artificial intelligence, machine learning and deep learning methods for analyzing intellectual property (IP) data,” *World Pat. Inf.*, vol. 55, pp. 37–51, 2018.
- [13] N. Bourbakis, A. Psarologou, G. Rematska and A. Esposito, "A Human-Like SPN Methodology for Deep Understanding of Technical Documents," *2016 IEEE 28th*



*International Conference on Tools with Artificial Intelligence (ICTAI)*, San Jose, CA, 2016, pp. 772-778, doi: 10.1109/ICTAI.2016.0121.

[14] N. Bourbakis & S. Mertoguno; “A Holistic Approach for Automatic Deep Understanding and Protection of Technical Documents”, in *International Journal on Artificial Intelligence Tools*, Vol. 29, No. 6 (2020) 2050007 (39 pages), DOI: 10.1142/S0218213020500074

[15] N. Bourbakis; W. Meng, C. Zhang, Z. Wu, N. J. Salerno; S. Borek; “Retrieval of Multimedia Web Documents and Removal of Redundant Information”, *International Journal on Artificial Intelligence Tools*, Vol.8, No.1 (1999)

[16] N. Bourbakis; “Converting Diagrams, Formulas, Tables, Graphics, and Pictures into SPN and NL-text Sentences for Automatic Deep Understanding of Technical Documents”, *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017, pp. 247-254, doi: 10.1109/ICTAI.2017.00047

[17] D. Saxton; E. Grefenstette; F. Hill; P. Kohli; “Analyzing Mathematical Reasoning Abilities of Neural Models” in *International Conference on Learning Representations*, 2019, doi: <https://doi.org/10.48550/arXiv.1904.01557>

[18] B. Berman & R. J. Fateman; “Optical Character Recognition for Typeset Mathematics” in *International Symposium of Symbolic and Algebraic Computation*, August 1994, pp. 348–353, doi: <https://doi.org/10.1145/190347.190438>

[19] Y. Chen; T. Shimizu; K. Yamauchi; M. Okada; "Ambiguous problem investigation in off-line mathematical expression understanding," *SMC 2000 conference proceedings, IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving*

to *Systems, Humans, Organizations, and their Complex Interactions'*, cat. no.0, 2000, pp. 2917-2922 vol.4, doi: 10.1109/ICSMC.2000.884443.

[20] S. Toyota; S. Uchida; M. Suzuki; “Structural Analysis of Mathematical Formulas with Verification based on Formula Description Grammar”, *DAS'06: Proceedings of the 7th international conference on Document Analysis Systems*, February 2006, pp. 153–163, doi: [https://doi.org/10.1007/11669487\\_14](https://doi.org/10.1007/11669487_14)

[21] Y. Guo, L. Huang, C. Liu and X. Jiang, "An Automatic Mathematical Expression Understanding System," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, pp. 719-723, doi: 10.1109/ICDAR.2007.4377009

[22] X. Lin; L. Gao; Z. Tang; X. Hu; X. Lin; “Identification of Embedded Mathematical Formulas in PDF Documents Using SVM”, *Proceedings of SPIE - The International Society for Optical Engineering*, January 2012, pp. 8297:31 -, doi: 10.1117/12.912445

[23] A. Nazemi; I. Murray; D. McMeekin; “Mathematical Information Retrieval (MIR) from Scanned PDF Documents and MathML Conversion”, *December 2014 IPSJ Transactions on Computer Vision and Applications* 6, pp.132-142, DOI: 10.2197/ipsjtcva.6.132

[24] M. Mahdavi; M. Condon; K. Davila; R. Zanibbi; "LPGA: Line-of-Sight Parsing with Graph-Based Attention for Math Formula Recognition," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 647-654, doi: 10.1109/ICDAR.2019.00109.

- [25] Zanibbi, R.; Blostein, D.; “Recognition and retrieval of mathematical expressions”, *International Journal on Document Analysis and Recognition (IJ DAR)*, 331–357 (2012).  
<https://doi.org/10.1007/s10032-011-0174-4>
- [26] E. E. Kostalia, E. G. M. Petrakis and N. Bourbakis, "Evaluating Methods for the Parsing and Understanding of Mathematical Formulas in Technical Documents," *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, 2020, pp. 407-412, doi: 10.1109/ICTAI50040.2020.00070
- [27] Kaliszyk, C.; Urban, J.; Vyskočil, J.; “Automating Formalization by Statistical and Semantic Parsing of Mathematics”. In: Ayala-Rincón, M., Muñoz, C. (eds) *Interactive Theorem Proving: ITP 2017; Lecture Notes in Computer Science*, vol 10499. Springer, Cham. [https://doi.org/10.1007/978-3-319-66107-0\\_2](https://doi.org/10.1007/978-3-319-66107-0_2)
- [28] Kaliszyk, C.; Urban, J.; “Learning-Assisted Automated Reasoning with Flyspeck”, in *Journal of Automated Reasoning* 53, pp. 173–213 (2014), doi: <https://doi.org/10.1007/s10817-014-9303-3>
- [29] C. Kaliszyk, J. Urban and J. Vyskocil, "System Description: Statistical Parsing of Informalized Mizar Formulas," in *19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2017, pp. 169-172, doi: 10.1109/SYNASC.2017.00036.
- [30] C. Kaliszyk, J. Urban and J. Vyskocil, “Efficient Semantic Features for Automated Reasoning Over Large Theories”, *IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence*, July 2015, pp: 3084–3090

- [31] Kaliszyk, C., Urban, J., Vyskočil, J. (2015). “Learning to Parse on Aligned Corpora (Rough Diamond)”. In: Urban, C., Zhang, X. (eds), *Interactive Theorem Proving. ITP 2015*. Lecture Notes in Computer Science, vol 9236. Springer, Cham. Doi: [https://doi.org/10.1007/978-3-319-22102-1\\_15](https://doi.org/10.1007/978-3-319-22102-1_15)
- [32] Kaliszyk, C., Urban, J., Vyskočil, J., & Geuvers, J. H. (2014). “Developing corpus-based translation methods between informal and formal mathematics: project description”. In S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, & J. Urban (Eds.), *Intelligent Computer Mathematics (International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings)* (pp. 435-439). (Lecture Notes in Computer Science; Vol. 8543). Springer. [https://doi.org/10.1007/978-3-319-08434-3\\_34](https://doi.org/10.1007/978-3-319-08434-3_34)
- [33] HOL Light Theorem official homepage, <https://www.cl.cam.ac.uk/~jrh13/hol-light/>
- [34] HALES, T., ADAMS, M., BAUER, G., DANG, T., HARRISON, J., HOANG, L., . . . ZUMKELLER, R. (2017). “A FORMAL PROOF OF THE KEPLER CONJECTURE”, in *Forum of Mathematics, Pi*, 5, E2. doi:10.1017/fmp.2017.1
- [35] Jiang, Z., Gao, L., Yuan, K., Gao, Z., Tang, Z., & Liu, X. (2018, October). “Mathematics content understanding for cyberlearning via formula evolution map”. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, (pp. 37-46).
- [36] X. Lin; L. Gao; X. Hu; Z. Tang; Y. Xiao; X. Liu; “A Mathematical Retrieval System for Formulas in Layout Presentations”, *SIGIR '14: Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, July 2014, pp. 697–706, doi: <https://doi.org/10.1145/2600428.2609611>

- [37] M. Grigore; M. Wolska; M. Kohlhase; “Towards Context-based Disambiguation of Mathematical Expressions”, in *The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium on Computer Mathematics and Mathematical Aspects of Computer and Information Sciences*, December 2009
- [38] J. Davenport; “On Writing OpenMath Content Dictionaries”, in *ACM Special Interest Group on Symbolic & Algebraic Manipulation, Bulletin Volume 34 Issue 2*, June 2000, pp 12–15, doi: <https://doi.org/10.1145/362001.362012>
- [39] M. Q. Nghiem; G. Yoko; Y. Matsubayashi; “Automatic Approach to Understanding Mathematical Expressions Using MathML Parallel Markup Corpora” (*International Organized Session on Application Oriented Principles of Machine Learning and Data Mining*) (2012), doi: [https://doi.org/10.11517/pjsai.JSAI2012.0\\_1K2IOS1b6](https://doi.org/10.11517/pjsai.JSAI2012.0_1K2IOS1b6)
- [40] Nghiem, M. Q., Kristianto, G.Y., Topić, G., Aizawa, A. (2013). A Hybrid Approach for Semantic Enrichment of MathML Mathematical Expressions. In: Carette, J., Aspinall, D., Lange, C., Sojka, P., Windsteiger, W. (eds) *Intelligent Computer Mathematics. CICM 2013. Lecture Notes in Computer Science*, vol 7961. Springer, Berlin, Heidelberg, doi: [https://doi.org/10.1007/978-3-642-39320-4\\_18](https://doi.org/10.1007/978-3-642-39320-4_18)
- [41] Ausbrooks et al, “Mathematical Markup Language (MathML)”, Version 3.0 (Second Edition), *W3C*, 2014
- [42] Sepp Hochreiter, Jürgen Schmidhuber; “Long Short-Term Memory”, in *Neural Computation*, 1997; 9 (8), pp: 1735–1780. doi: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [43] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762.

- [44] M. Allamanis; P. Chanthirasegaran; P. Kohli; X. Sutton; “Learning Continuous Semantic Representations of Symbolic Expressions”, *ICML'17: Proceedings of the 34th International Conference on Machine Learning - Volume 70* August 2017, pp. 80–88, arXiv:1611.01423
- [45] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. “Semantic Compositionality through Recursive Matrix-Vector Spaces”. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201–1211, Jeju Island, Korea. Association for Computational Linguistics, <https://aclanthology.org/D12-1110>
- [46] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics, <https://aclanthology.org/D13-1170>
- [47] Wojciech Zaremba, Karol Kurach, and Rob Fergus. 2014. “Learning to discover efficient mathematical identities”. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'14)*. MIT Press, Cambridge, MA, USA, 1278–1286.
- [48] Zaremba & Sutskever, “Learning-to-Execute”, in *International Conference on Learning Representations*, 2015, arXiv:1410.4615

- [49] G. Rematska; N. Bourbakis; “A Stochastic Petri Net Reverse Engineering Methodology for Deep Understanding of Technical Documents”, Doctorate Dissertation, 2018, core-scholar: [https://corescholar.libraries.wright.edu/etd\\_all/1946/](https://corescholar.libraries.wright.edu/etd_all/1946/)
- [50] S. Shrihari; S. Lam; V. Govindaraju; R. Srihari; J. Hull; “Document Understanding: Research Directions”, 1992, Corpus ID: 60828228
- [51] Luo, J., Li, Z., Wang, J., & Lin, C. (2021). ChartOCR: Data Extraction from Charts Images via a Deep Hybrid Framework. 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), 1916-1924.
- [52] S. E. V. and P. Samuel, "Automatic Code Generation from UML State Chart Diagrams," in IEEE Access, vol. 7, pp. 8591-8608, 2019, doi: 10.1109/ACCESS.2018.2890791.
- [53] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondrej Certik, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Stepan Roucka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. SymPy: symbolic computing in python. PeerJ Computer Science, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL <https://doi.org/10.7717/peerj-cs.103>.
- [54] Wolfram-Research. Mathematica, version 12.0, 2019. Champaign, IL, 2019
- [55] MathWorks. MATLAB optimization toolbox (r2019a), 2019. The MathWorks, Natick, MA, USA

- [56] The LaTeX Project, Leslie Lamport, 2022, <https://www.latex-project.org/latex3/>
- [57] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, Gordon Woodhull. GraphViz and dynagraph – static and dynamic graph drawing tools, GRAPH DRAWING SOFTWARE, 2003, pp. 127-148
- [58] Collin J. Delker, SchemDraw, 2022, <https://schemdraw.readthedocs.io/en/latest/>
- [59] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio; “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, arXiv:1502.03044v3, doi: <https://doi.org/10.48550/arXiv.1502.03044>
- [60] Soh, Moses. “Learning Cnn Lstm Architectures for Image Caption Generation.” (2016).
- [61] Tensorflow documentation of the EfficientNet architectures, [https://www.tensorflow.org/api\\_docs/python/tf/keras/applications/efficientnet/](https://www.tensorflow.org/api_docs/python/tf/keras/applications/efficientnet/)
- [62] Mingxing Tan & Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019, Revised 2020, <https://arxiv.org/pdf/1905.11946.pdf>
- [63] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, Layer Normalization, published in arXiv-preprint arXiv:1607.06450, 2016, <https://arxiv.org/pdf/1607.06450.pdf>
- [64] Diederik P. Kingma, Jimmy Ba; “Adam: A Method for Stochastic Optimization”, 2014, arXiv:1412.6980, doi: <https://doi.org/10.48550/arXiv.1412.6980>



- [65] Keras image captioning model, [https://keras.io/examples/vision/image\\_captioning/](https://keras.io/examples/vision/image_captioning/)
- [66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, in *Journal of Machine Learning Research* 2014, 15(56):1929–1958, 2014
- [67] Riyaz Sikora & O'la Hmoud Al-laymoun; “A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms”, *Journal of International Technology and Information Management*, Volume 23, 2014, p.1-12, ISSN: 1941-6679
- [68] Simonyan, Karen, and Andrew Zisserman; "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
- [69] В. И. Левенштейн (1965). Двоичные коды с исправлением выпадений, вставок и замещений символов [Binary codes capable of correcting deletions, insertions, and reversals]. Доклады Академии Наук СССР (in Russian). 163 (4): 845–848. Appeared in English as: Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet Physics Doklady*. 10 (8): 707–710. Bibcode:1966SPhD...10.707L
- [70] Dice, Lee R. (1945). "Measures of the Amount of Ecologic Association Between Species". *Ecology*. 26 (3): 297–302. doi:10.2307/1932409. JSTOR 1932409.
- [71] <https://github.com/seatgeek/fuzzywuzzy>
- [72] Chomsky, Noam (1956). *Three models for the description of language*. IRE Transactions on Information Theory. 2 (3): 113–124. doi:10.1109/TIT.1956.1056813.

- [73] Chomsky, Noam (1957). *Syntactic Structures*, The Hague/Paris: Mouton, ISBN 978-3-11-021832-9
- [74] Pakin, Scott (2021). The Comprehensive LATEX Symbol List, <http://www.ctan.org/>
- [75] Udrescu SM, Tegmark M. AI Feynman: A physics-inspired method for symbolic regression. *Sci Adv.* 2020 Apr 15;6(16):eaay2631. doi: 10.1126/sciadv.aay2631. PMID: 32426452; PMCID: PMC7159912
- [76] Max Tegmark, *Feynman Symbolic Regression Database*, Department of Physics, MIT, <https://space.mit.edu/home/tegmark/aifeynman.html>
- [77] R. Feynman, R. Leighton, M. Sands, *The Feynman Lectures on Physics: The New Millennium Edition: Mainly Mechanics, Radiation, and Heat*, vol. 1 (Basic Books, 1963); <https://books.google.com/books?id=d76DBQAAQBAJ>.
- [78] R. Feynman, R. Leighton, M. Sands, *The Feynman Lectures on Physics*, vol. 2 in *The Feynman Lectures on Physics* (Pearson/Addison-Wesley, 1963b); <https://books.google.com/books?id=AbuAAAAMAAJ>.
- [79] R. Feynman, R. Leighton, M. Sands, *The Feynman Lectures on Physics*, vol. 3 in *The Feynman Lectures on Physics* (Pearson/Addison-Wesley, 1963); <https://books.google.com/books?id=6XvAAAAMAAJ>.
- [80] Kanervisto, Anssi. (2016). im2latex-100k , arXiv:1609.04938 Zenodo. <https://doi.org/10.5281/zenodo.56198>
- [81] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th*

International Conference on Machine Learning - Volume 70 (ICML'17). JMLR.org, 980–989.

[82] Schmidt M, Lipson H. Distilling free-form natural laws from experimental data. *Science*. 2009 Apr 3;324(5923):81-5. doi: 10.1126/science.1165893. PMID: 19342586.

[83] Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients, from International Conference on Learning Representations, 2021, arXiv:1912.04871

[84] Cava-Lab, Harvard Medical School, <https://cavalab.org/srbench/>

[85] Guillaume Lample & Francois Charton; “Deep Learning for Symbolic Mathematics”, in *International Conference on Learning Representations*, 2020

[86] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka; “How Powerful are Graph Neural Networks?”, International Conference on Learning Representations (ICLR), 2019, arXiv:1810.00826

[87] Matthias Fey & Jan Eric Lenssen; “Fast Graph Representation Learning with PyTorch Geometric”, in *International Conference on Learning Representations (ICLR)*, 2019, arXiv:1903.02428

[88] Thomas N. Kipf, Max Welling; “Semi-Supervised Classification with Graph Convolutional Networks”, *International Conference on Learning Representations*, 2017, arXiv:1609.02907

[89] R. Fateman, T. Tokuyasu, B. Berman and N. Mitchell, "Optical character recognition and parsing of typeset mathematics," *Journal of Visual Communication and Image Representation*, vol. 7, no. 1, pp. 2-15, 1996

[90] R. Zanibbi, D. Blostein and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455-1467, 2002

# Appendix

## A.3.1 Pseudocode LaTeX Customize Templates

### I. Multiple Algorithms

---

**Algorithm 1:** Description of the block  $(I - block) - 256 - bytes - RAM...$

---

- 1 Function  $(I - block) - 256 - bytes - RAM ()$ ;  
**Input** :  $S_i - read, S_i - write, data, address, clock, reset$   
**Output:**  $S_i$
- 2 Express the functionality of the block;
- 3 Return  $S_i$  ;
- 4 end;

---

**Algorithm 2:** Description of the block  $(j - block) - 256 - bytes - RAM...$

---

- 1 Function  $(j - block) - 256 - bytes - RAM ()$ ;  
**Input** :  $S_j - read, S_j - write, data1, address1, clock, reset$   
**Output:**  $S_j$
- 2 Express the functionality of the block;
- 3 Return  $S_j$  ;
- 4 end;

---

**Algorithm 3:** Description of the block  $(t - block) - 256 - bytes - RAM...$

---

- 1 Function  $(t - block) - 256 - bytes - RAM ()$ ;  
**Input** :  $S_t - read, S_t - write, data2, address2, clock, reset$   
**Output:**  $S_t$
- 2 Express the functionality of the block;
- 3 Return  $S_t$  ;
- 4 end;

---

**Algorithm 4:** Description of the diagram...

---

- 1 Function  $MainDiagram ()$ ;  
**Input** :  $data1, S_i - write, S_i - read, S_j - read, address2, reset, data2, S_t - read, S_t - write, address1, address, S_j - write, data, clock$   
**Output:**  $S_t, S_j, S_i$
- 2  $S_i = (I - block) - 256 - bytes - RAM(S_i - read, S_i - write, data, address, clock, reset)$ ;
- 3  $S_j = (j - block) - 256 - bytes - RAM(S_j - read, S_j - write, data1, address1, clock, reset)$ ;
- 4  $S_t = (t - block) - 256 - bytes - RAM(S_t - read, S_t - write, data2, address2, clock, reset)$ ;
- 5 Return  $S_t, S_j, S_i$  ;
- 6 end;

---

Figure 50: LaTeX Custom Template (1)

### II. Single Algorithm

---

**Algorithm 1** *Figure 4*

**Input:**  $address2, reset, data, St - write, St - read, address1, address, clock, Sj - read, Si - write, data1, data2, Sj - write, Si - read$

**Output:**  $Sj, Si, St$

---

```
1: procedure ( $I - block$ ) - 256 - bytes - RAM( $Si - read, Si - write, data, address, clock, reset$ )
2:   UnknownBlock()
3:   return  $Si$ 
4: end procedure
```

---

```
1: procedure ( $j - block$ ) - 256 - bytes - RAM( $Sj - read, Sj - write, data1, address1, clock, reset$ )
2:   UnknownBlock()
3:   return  $Sj$ 
4: end procedure
```

---

```
1: procedure ( $t - block$ ) - 256 - bytes - RAM( $St - read, St - write, data2, address2, clock, reset$ )
2:   UnknownBlock()
3:   return  $St$ 
4: end procedure
```

---

```
1: procedure  $main()(x)$ 
2:    $Si \leftarrow (I - block) - 256 - bytes - RAM(Si - read, Si - write, data, address, clock, reset)$ 
3:    $Sj \leftarrow (j - block) - 256 - bytes - RAM(Sj - read, Sj - write, data1, address1, clock, reset)$ 
4:    $St \leftarrow (t - block) - 256 - bytes - RAM(St - read, St - write, data2, address2, clock, reset)$ 
5:   return  $Sj, Si, St$ 
6: end procedure
```

**end**

---

Figure 51: LaTeX Custom Template (2)

## A.3.2 Example Internal Representations

### Formal Representation

**NOR:** IN(nz, lu, fr, gi)|DS(negates an OR clause of inputs)|OUT(dm),

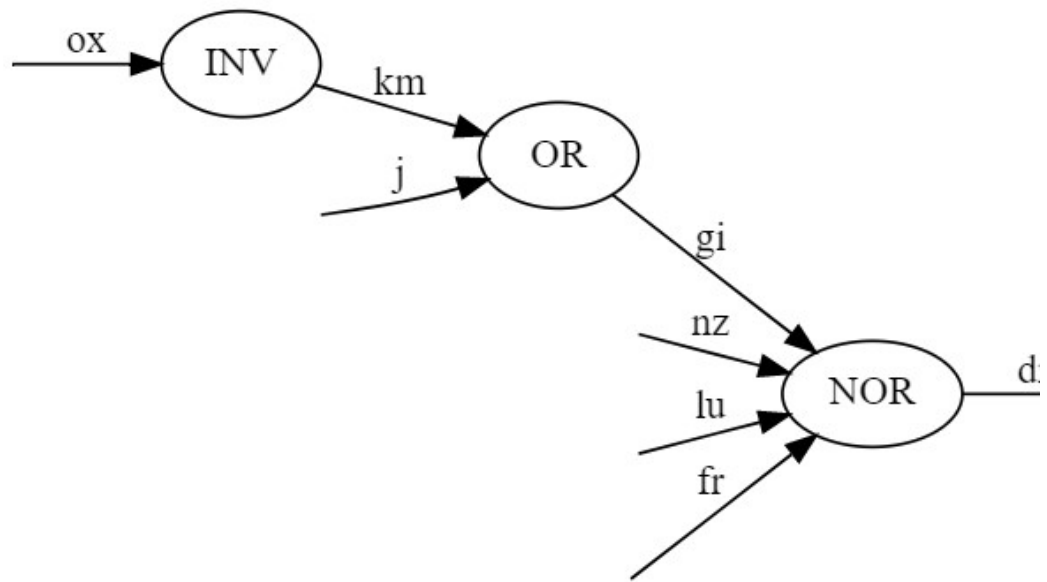
**OR:** IN(km, j)|DS(an OR clause of inputs)|OUT(gi)

**INV:** IN(ox)|DS(negates a single input)|OUT(km)

### CSV

	block	Num	Inputs	Outputs	Connections
0	NOR	1	nz	dm	NaN
1	NOR	2	lu	NaN	NaN
2	NOR	3	fr	NaN	NaN
3	NOR	4	gi	NaN	NaN
4	OR	1	km	gi	NOR
5	OR	2	j	NaN	NaN
6	INV	1	ox	km	OR

### Graphics



### Pseudocode



---

**Algorithm 1 Example**

---

**Input:**  $lu, ox, nz, fr, j$ **Output:**  $dm$ 

---

```
1: procedure INV( $ox$ )
2:    $km \leftarrow \neg ox$ 
3:   return  $km$ 
4: end procedure
```

---

```
1: procedure NOR( $nz, lu, fr, gi$ )
2:    $dm \leftarrow Nor(nz, lu, fr, gi)$ 
3:   return  $dm$ 
4: end procedure
```

---

```
1: procedure OR( $km, j$ )
2:    $gi \leftarrow j \vee km$ 
3:   return  $gi$ 
4: end procedure
```

---

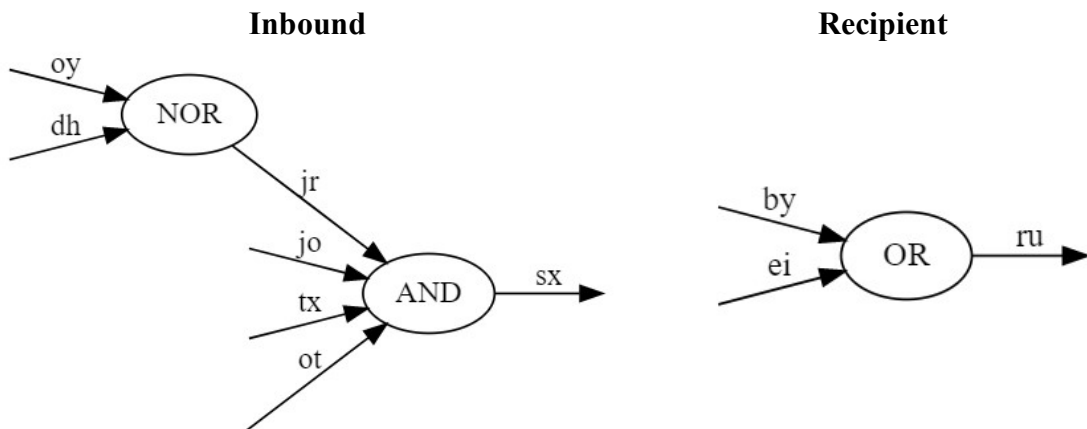
```
1: procedure main()( $x$ )
2:    $km \leftarrow INV(ox)$ 
3:    $dm \leftarrow NOR(nz, lu, fr, gi)$ 
4:    $gi \leftarrow OR(km, j)$ 
5:   return  $dm$ 
6: end procedure
end
```

---

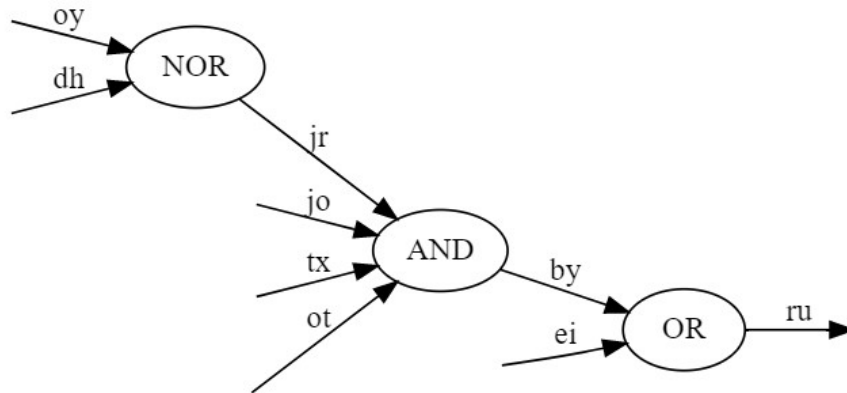
Figure 52: Example Internal Representations

### A.3.3 Data Augmentation Methods

#### I. Data Augmentation Methods – Composition



### Composed



### Inbound

DEFINE (Function) AND

Input: jo, tx, jr, ot

Main:  $sx = jo \& jr \& ot \& tx$

Output: sx

DEFINE (Function) NOR

Input: oy, dh

Main:  $jr = \text{Nor}(oy, dh)$

Output: jr

DEFINE (Function) main()

1.  $sx = \text{AND}(jo, tx, jr, ot)$

2.  $jr = \text{NOR}(oy, dh)$

### Recipient

DEFINE (Function) OR

Input: by, ei

Main:  $ru = by \mid ei$

Output: ru

DEFINE (Function) main()

1.  $ru = \text{OR}(by, ei)$

### Composed

DEFINE (Function) AND

Input: jo, tx, jr, ot

Main: by = jo & jr & ot & tx

Output: by

DEFINE (Function) NOR

Input: oy, dh

Main: jr = Nor(oy, dh)

Output: jr

DEFINE (Function) OR

Input: by, ei

Main: ru = by | ei

Output: ru

DEFINE (Function) main()

1. by = AND(jo, tx, jr, ot)

2. jr = NOR(oy, dh)

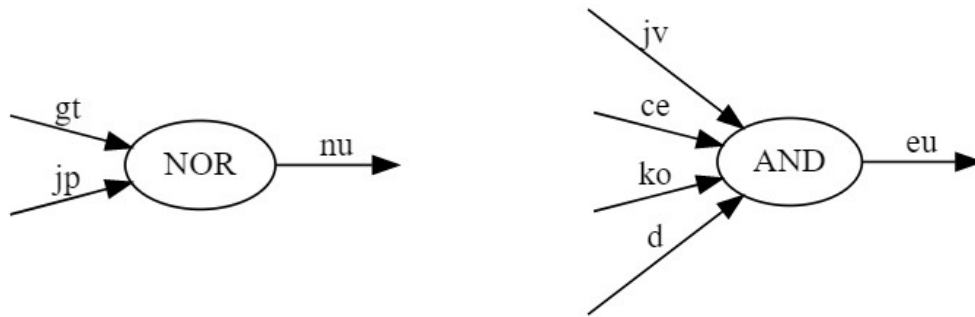
3. ru = OR(by, ei)

*Figure 53: Data Augmentation Methods – Composition*

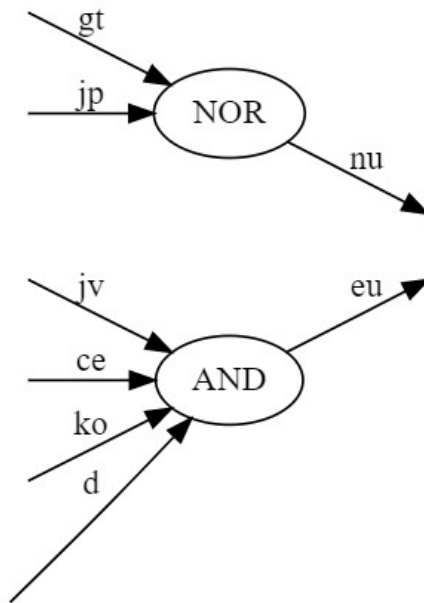
## II. Data Augmentation Methods – Merging

**Inbound**

**Recipient**



**Composed**



**Inbound**

DEFINE (Function) NOR

Input: gt, jp

Main: nu = Nor(gt, jp)

Output: nu

DEFINE (Function) main()

1. nu = NOR(gt, jp)

**Recipient**

DEFINE (Function) AND

Input: jv, ce, ko, d

Main: eu = ce & d & jv & ko

Output: eu

DEFINE (Function) main()

1. eu = AND(jv, ce, ko, d)

### **Merged**

DEFINE (Function) AND

Input: jv, ce, ko, d

Main: eu = ce & d & jv & ko

Output: eu

DEFINE (Function) NOR

Input: gt, jp

Main: nu = Nor(gt, jp)

Output: nu

DEFINE (Function) main()

1. eu = AND(jv, ce, ko, d)

2. nu = NOR(gt, jp)

*Figure 54: Data Augmentation Methods – Merging*

### **III. Data Augmentation Methods – Rotation**

**Original**

**Rotated**

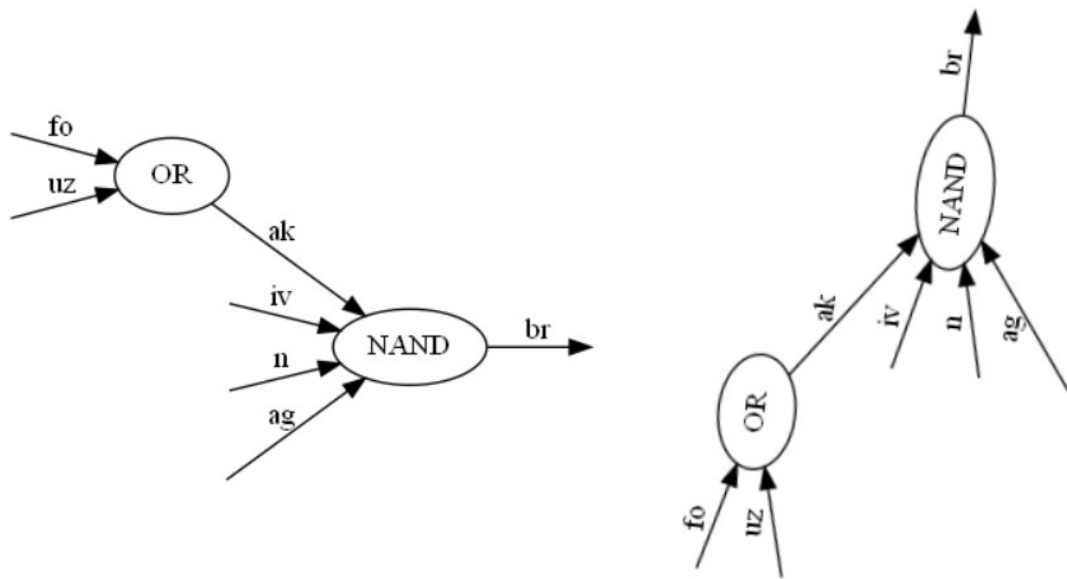


Figure 55: Data Augmentation Methods – Rotation

### A.3.4 Application on Newtonian Motion

#### I. GraphViz Example

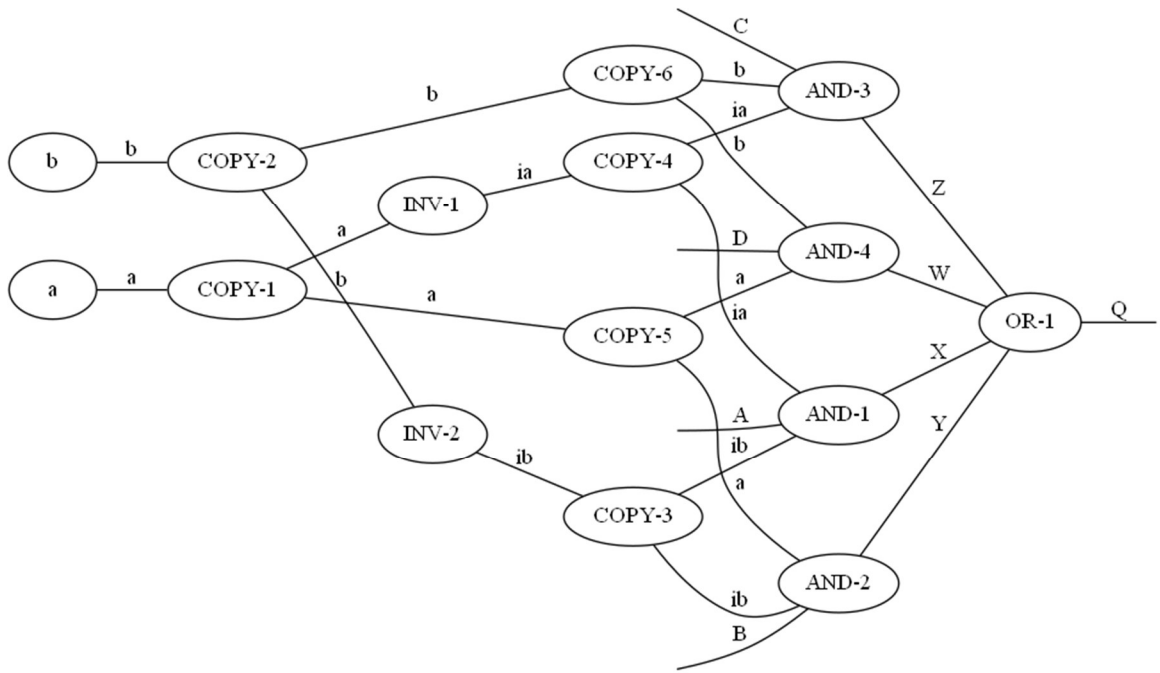


Figure 56: GraphViz Visualization Example

## II. SchemDraw

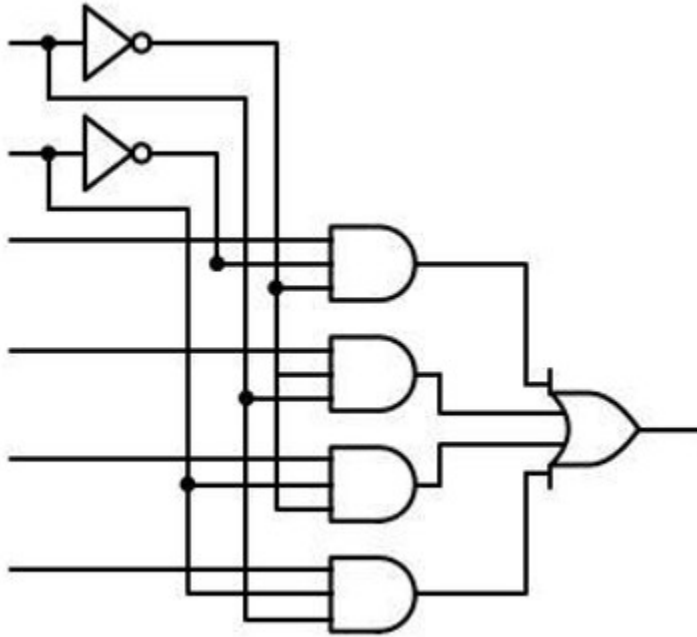


Figure 57: SchemDraw Visualization Example

### A.3.5 Application on Newtonian Motion

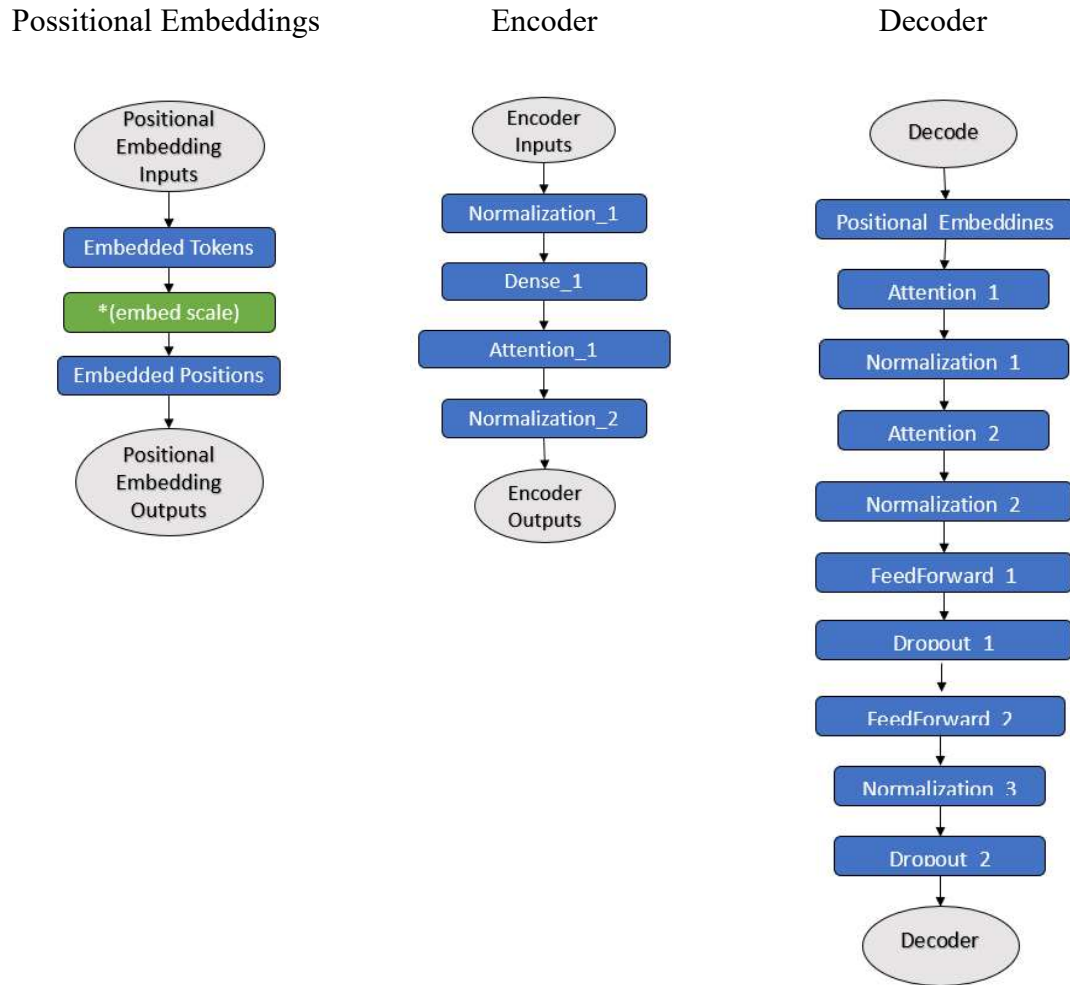


Figure 58: Image Captioning model architecture

### A.3.6 List of Composite Elements

BASIC GATES		COMPOSITE ELEMENTS	
NOT	(1)	JK FLIP – FLOP	(1)
AND	(2)	SR FLIP – FLOP NOR	(2)
OR	(3)	SR FLIP – FLOP NAND	(3)
NAND	(4)	SR FLIP – FLOP NOT NAND	(4)
NOR	(5)	SR FLIP – FLOP NAND NOT	(5)
XOR	(6)	SR FLIP – FLOP GATED	(6)
XNOR	(7)	D – TYPE FLIP – FLOP	(7)



D – TYPE FLIP – FLOP INV	(8)
MASTER – SLAVE FLIP – FLOP	(9)
2 – 1 MULTIPLEXER	(10)
4 – 1 MULTIPLEXER	(11)
HALF – ADDER STD	(12)
HALF – ADDER NOR	(13)
HALF – ADDER NAND	(14)
HALF – ADDER NAND INV	(15)
FULL – ADDER STD	(16)
FULL – ADDER NOR	(17)
FULL – ADDER NAND	(18)

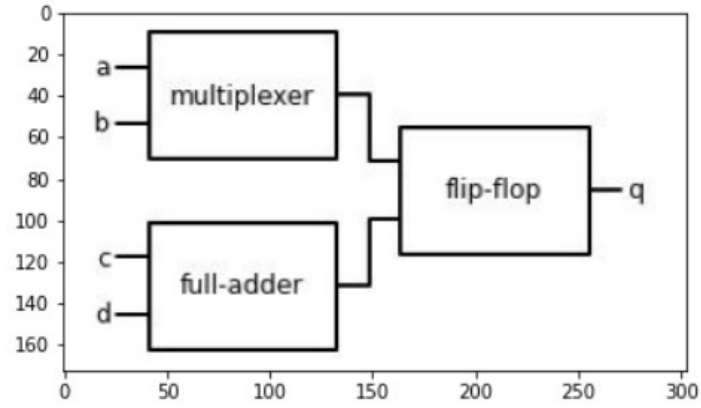
Table 16: Composite elements

### A.3.7 Block Hierarchy

<b>Block Hierarchy</b>	
<i>Multiplexer</i>	5
<i>Full – Adder</i>	4
<i>Half – Adder</i>	3
<i>Flip – Flop</i>	2
<i>Xnor</i>	1
<i>Nor</i>	1
<i>Xor</i>	1
<i>Or</i>	1
<i>Nand</i>	1
<i>And</i>	1
<i>Not</i>	1

Table 17: Composite Elements

### A.3.8 Detailed Grid Partitioning Example



Original Image

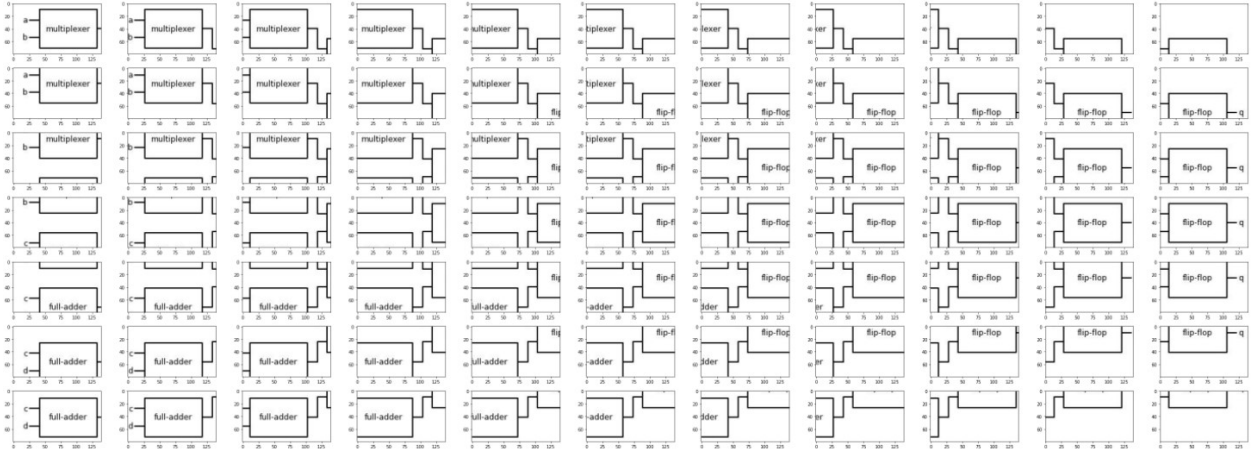
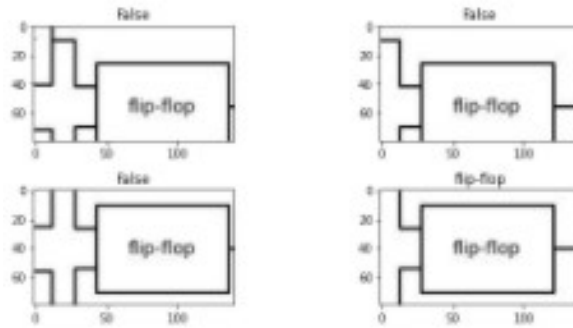
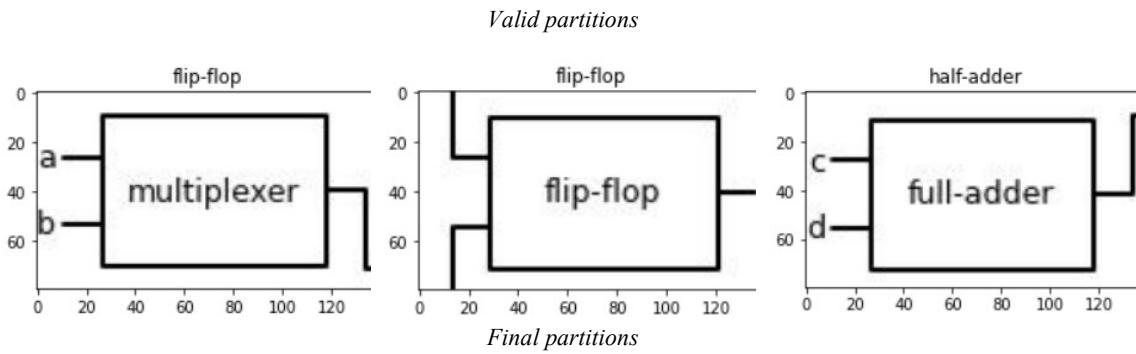
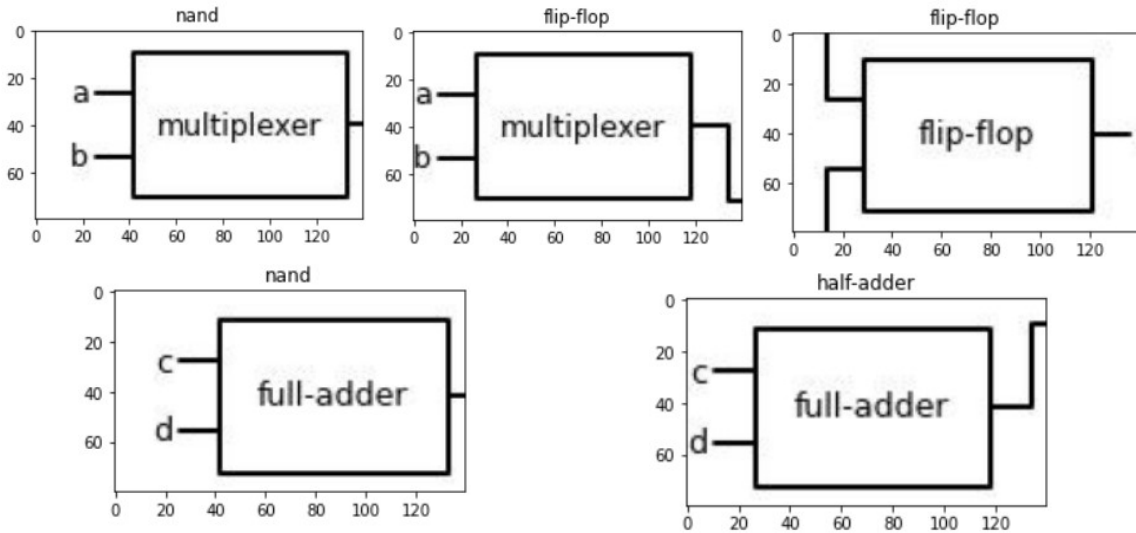


Image Partitioning



Partition classification



```
< start >
g = multiplexer ( o , d )
< end >
```

```
< start >
ukn3
= full - adder ( ukn1 , ukn2 )
< end >
```

```
< start >
c = full - adder ( c , d )
< end >
```

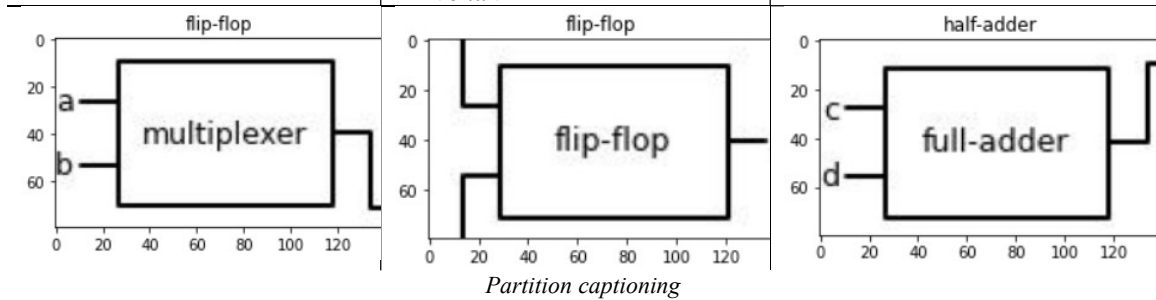


Figure 59: Grid partitioning Example

### A.3.9 Ensemble Configuration Changes

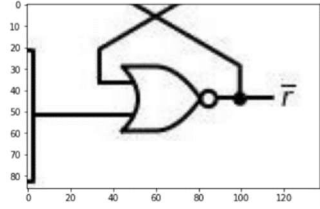
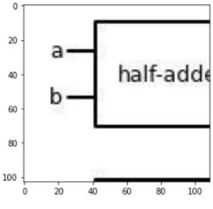
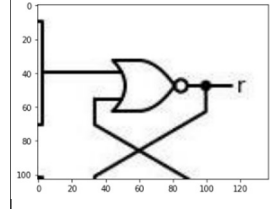
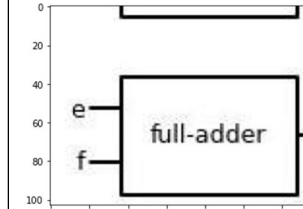
V step	H step	CHF	CVF
10	10	[0.2, 0.3, 0.4, 0.5, 0.6]	[0.2, 0.3, 0.4, 0.5, 0.6]
(c.1.1)	(c.1.2)	(c.1.3)	(c.1.4)
			
<b>Label</b>	unk1 = half_adder ( a , b ) ; unk2 = full_adder ( e , f ) ; r = half_adder ( unk1 , unk2 )		
<b>Prediction</b>	z = flip_flop ( j , 1 ) ; a = nand ( a , d ) ; r = flip_flop ( y , 1 ) ; m = full-adder ( a , f )		

Table 18: Ensemble Configuration Changes

## A.4.1 Application on Newtonian Motion

Below, we showcase the methodology by applying it on three popular formulas of the Newtonian Laws of Motion domain: instantaneous velocity, instantaneous acceleration, and momentum. These three expressions form the level-0 of the graph  $G_{res}$ , acting as building blocks for the construction of the next levels.

The a priori knowledge that we have is that these three expressions, apart from being syntactically valid, also semantically describe certain motion-related phenomena. To exhibit the inductive reasoning of the methodology, we reconstruct the level-0 of the  $G_{res}$ , by incorporating the corresponding subset of terminal symbols and production rules.

### Level-0: Subjects

1.  $m$
2.  $v$
3.  $x$

The binding operators used are multiplication and differentiation through time.

**Level-0: Binding operators**

1. Multiplication (\*)
2. Differentiation through time ( $\frac{d}{dt}$ )

The latter is a unary operator; thus, it can be managed in two ways: either it is applied on the subjects beforehand to augment the subset, or normally like the rest of the binding operators. The difference between the two applications is the level of analysis each byproduct occurs at - e.g., zero level with augmentation, first level with normal application. It also generates the condition that:  $\forall f$  in  $df/dt$ ,  $f$  is differentiable per  $t$ , which holds for every subject in the current setup, therefore we can omit it for simplicity, but it is retained in the specific example, in order to demonstrate the way indeterminacy checks work.

We also restrict to the analysis of pairs, although the operations used can surpass the duality and receive multiple arguments. Using the pre-stage augmentation for unary operators, the byproduct analysis for level-0 displays the following structure:

**Level-0: Subjects augmented**

<b>R</b>	<b>Validity</b>	<b>Conditions</b>
<i>m</i>	<i>True</i>	[]
<i>v</i>	<i>True</i>	[]
<i>x</i>	<i>True</i>	[]
<i>dm/dt</i>	<i>True</i>	[ <i>dt</i> ≠ 0]
<i>dv/dt</i>	<i>True</i>	[ <i>dt</i> ≠ 0]

$dx/dt$	<i>True</i>	$[dt \neq 0]$
---------	-------------	---------------

Table 19: Level-0 Augmented Subjects

**Level-0: Analysis of multiplication**

<b>R1</b>	<b>R2</b>	<b>Synthesis</b>	<b>Validity</b>	<b>Conditions</b>
$m$	$v$	$m * v$	<i>True</i>	$[]$
$m$	$x$	$m * x$	<i>True</i>	$[]$
$m$	$dm/dt$	$dm * m/dt$	<i>Conditional</i>	$[dt \neq 0]$
$m$	$dv/dt$	$dv * m/dt$	<i>Conditional</i>	$[dt \neq 0]$
$m$	$dx/dt$	$dx * m/dt$	<i>Conditional</i>	$[dt \neq 0]$
$v$	$x$	$v * x$	<i>True</i>	$[]$
$v$	$dm/dt$	$dm * v/dt$	<i>Conditional</i>	$[dt \neq 0]$
$v$	$dv/dt$	$dv * v/dt$	<i>Conditional</i>	$[dt \neq 0]$
$v$	$dx/dt$	$dx * v/dt$	<i>Conditional</i>	$[dt \neq 0]$
$x$	$dm/dt$	$dm * x/dt$	<i>Conditional</i>	$[dt \neq 0]$
$x$	$dv/dt$	$dv * x/dt$	<i>Conditional</i>	$[dt \neq 0]$
$x$	$dx/dt$	$dx * x/dt$	<i>Conditional</i>	$[dt \neq 0]$
$dm/dt$	$dv/dt$	$dm * dv/dt^2$	<i>True</i>	$[]$
$dm/dt$	$dx/dt$	$dm * dx/dt^2$	<i>True</i>	$[]$

$dv/dt$	$dx/dt$	$dv * dx/dt^2$	<i>True</i>	$\square$
---------	---------	----------------	-------------	-----------

Table 20: Level-0 analysis of multiplication

The results comprise several outcomes with conditional validity. The conditions sum up to time difference being other than zero, which in differentiation holds, but we kept the outcome to showcase the automatically applied division check. The table above shows that every outcome using multiplication as binding operator is syntactically valid. For the majority of them, the semantic interpretation remains unknown, but for the sake of the example, we use our a priori domain knowledge to distill the three preferred expressions and construct the  $G_{res_0}$  – to showcase the equivalence between the two unary operators, we use the application of unary operators as normal binding operator for the graph – ,

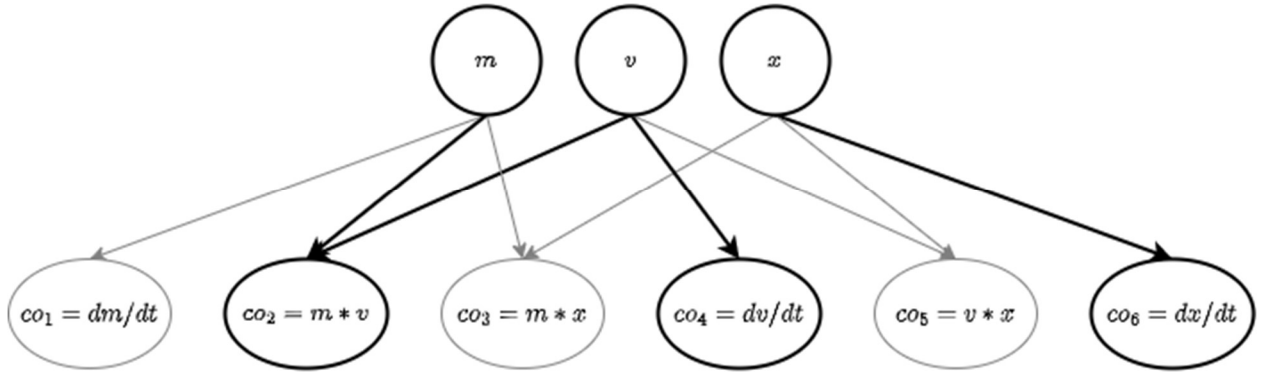


Figure 60: Level-1 byproducts

from which we can derive the semantically significant expressions according to our a priori domain knowledge:

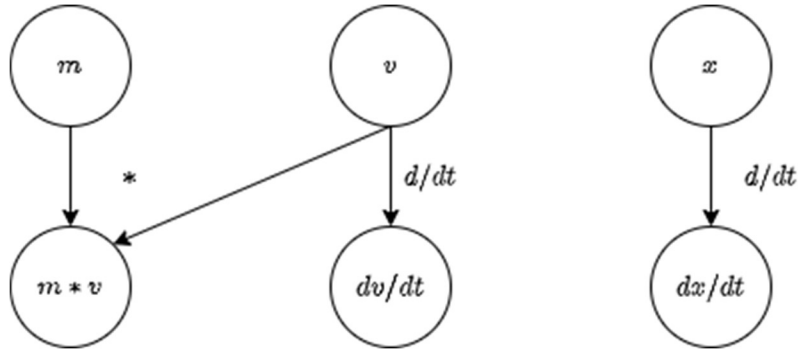


Figure 61: Confirmed semantically valid byproducts

### **Level-1**

After showcasing the methodology on the constructing the initial subjects, we continue with one level higher in the graph's hierarchy.

#### **Level-1: Subject**

1.  $v = \frac{dx}{dt}$
2.  $= \frac{dv}{dt}$
3.  $p = mv$

To keep things minimal, we use a subset of the grammar's production rules containing the four basic mathematical operations (addition, subtraction, multiplication, division) as binding operators, to create the graph  $G_{res_1}$ . Again, restrict to the analysis of pairs, although the operations used can surpass the duality and receive multiple arguments.

No unary operator is included; therefore, the subjects are not augmented.

#### **Level-1: Binding operators**

1. Addition (+)
2. Subtraction (−)



3. Multiplication (\*)

4. Division (/)

The only operation that could arise indeterminable expressions is division, in which case we perform checks for each generated expression, in order to compute the conditional relation. We follow the methodology as described in level 0, displaying below some of the pairs as samples:

$$(0): v = \frac{dx}{dt}, (1): a = \frac{dv}{dt}$$

	<b>R1</b>	<b>R2</b>	<b>Synthesis</b>	<b>Validity</b>	<b>Conditions</b>
<b>Addition</b>	$v = \frac{dx}{dt}$	$a = \frac{dv}{dt}$	$co_0 = a + v$ $= \frac{dx + dv}{dt}$	Conditional	$[dt \neq 0]$
<b>Subtraction</b>	$v = \frac{dx}{dt}$	$a = \frac{dv}{dt}$	$co_1 = a - v$ $= -\frac{dv - dx}{dt}$	Conditional	$[dt \neq 0]$
<b>Multiplication</b>	$v = \frac{dx}{dt}$	$a = \frac{dv}{dt}$	$co_2 = a * v = \frac{dv * dx}{dt^2}$	Conditional	$[dt \neq 0]$
<b>Division</b>	$v = \frac{dx}{dt}$	$a = \frac{dv}{dt}$	$co_3 = \frac{v}{a} = \frac{dx}{dv}$	Conditional	$[dv \neq 0, a \neq 0]$

Table 21: Pair synthesis (1)

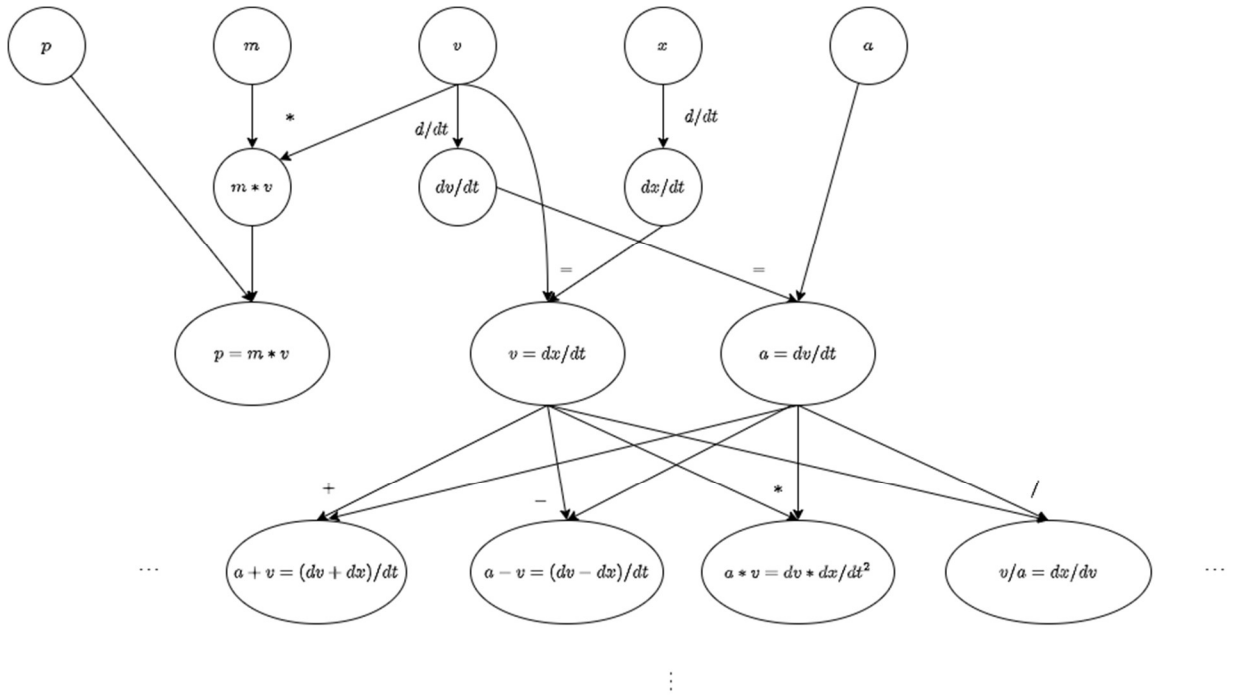


Figure 62: Pair synthesis (1)

(1):  $a = \frac{dv}{dt}$ , (2):  $p = mv$

	R1	R2	Synthesis	Validity	Conditions
<b>Addition</b>	$a = \frac{dv}{dt}$	$p = mv$	$co_4 = a + p = m * v + \frac{dv}{dt}$	Conditional	$[dt \neq 0]$
<b>Subtraction</b>	$a = \frac{dv}{dt}$	$p = mv$	$co_5 = a - p = -m * v + \frac{dv}{dt}$	Conditional	$[dt \neq 0]$
<b>Multiplication</b>	$a = \frac{dv}{dt}$	$p = mv$	$co_6 = a * p = m * v * \frac{dv}{dt}$	Conditional	$[dt \neq 0]$
<b>Division</b>	$a = \frac{dv}{dt}$	$p = mv$	$co_7 = \frac{a}{p} = \frac{dv}{m * v * dt}$	Conditional	$[dt \neq 0, v \neq 0, m \neq 0]$

Table 22: Pair synthesis (2)

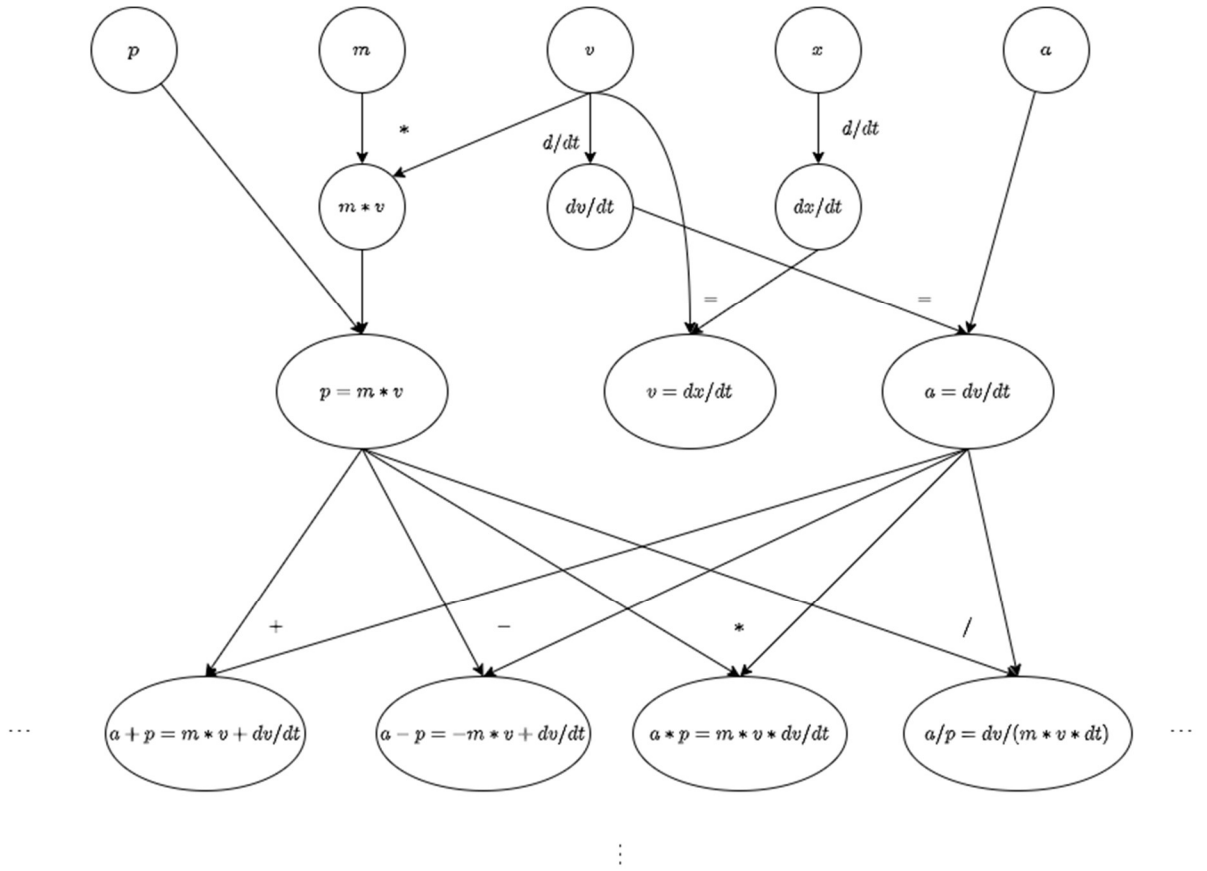


Figure 63: Pair synthesis (2)

The combined  $G_{res}$  is partially depicted below:

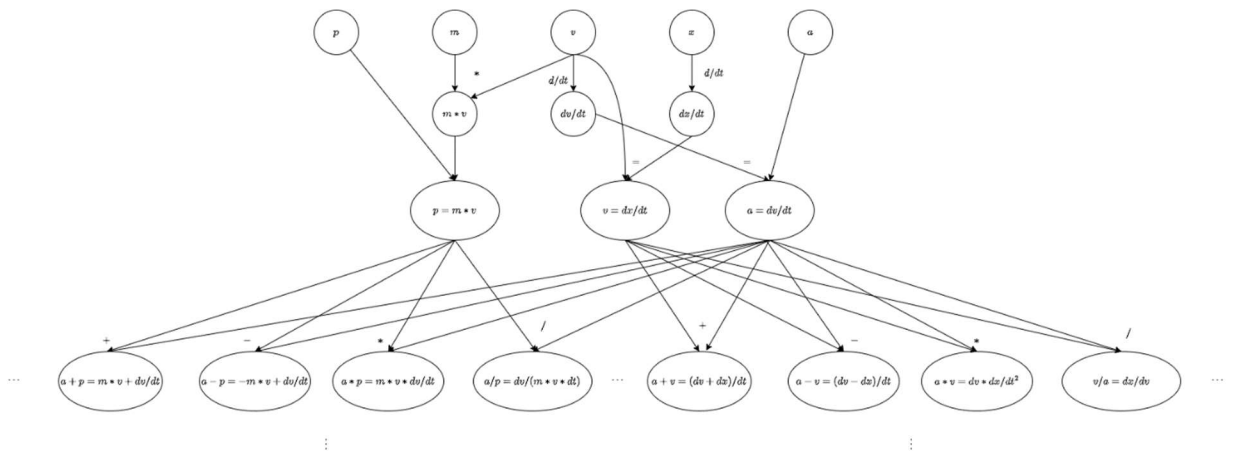


Figure 64: Combined  $G_{res}$

## A.4.2 Paths to valid expressions for inner and outer vector product as binding operators.

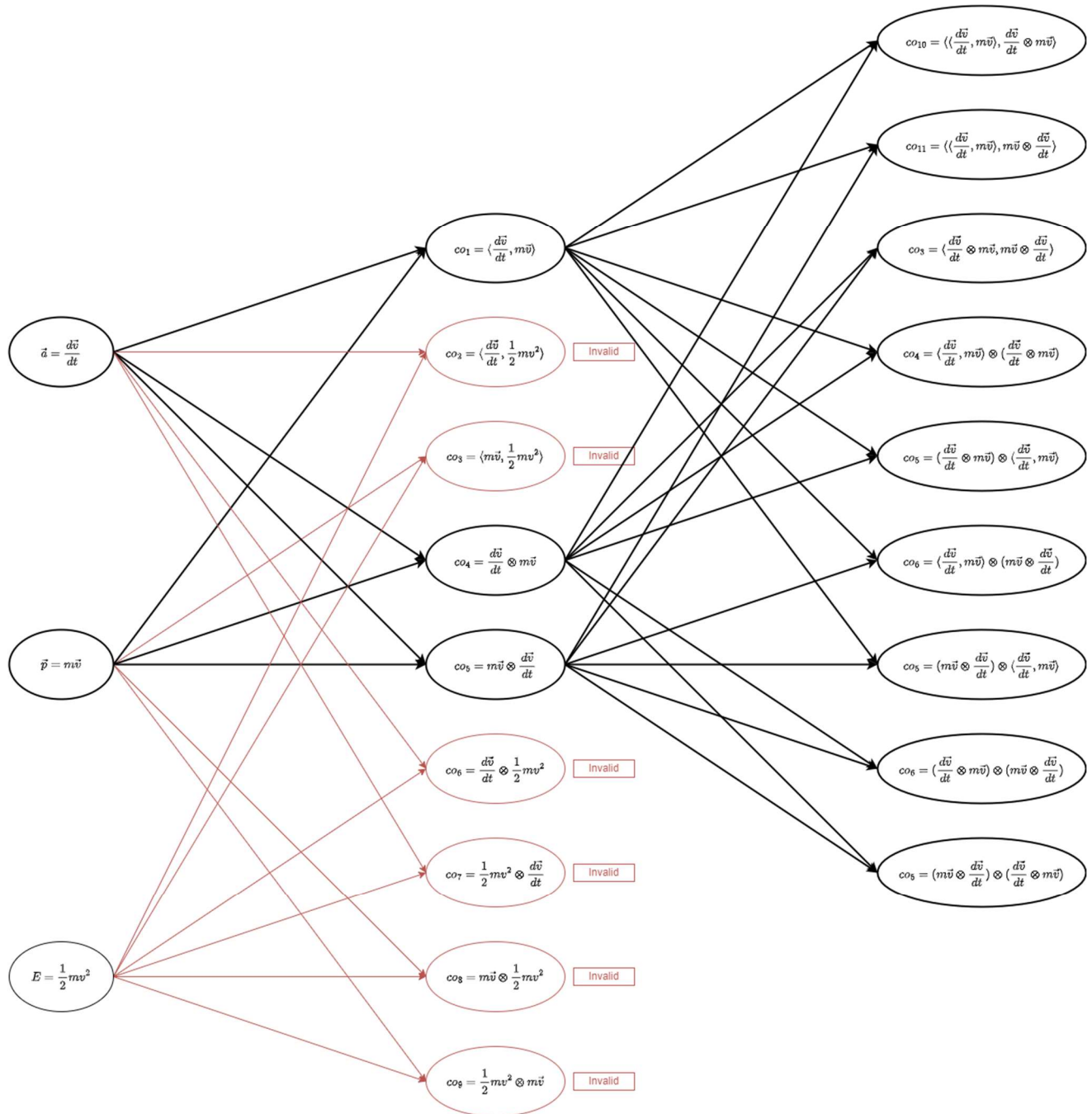


Figure 65: Synthesis Methodology Example

