Autoformalization for Agda via Fine-tuning LLMs

1. Introduction

Autoformalization is the task of translating mathematical natural language expressions into fully formalized, machine-verifiable forms. Recently, many attempts at autoformalization are based on training or fine-tuning large language models. However, these related works have encountered a key problem: it is very difficult to build or even find a high-quality, large parallel corpus of formal language and natural language. This difficulty is faced by all formal languages, not only for the most popular Lean, not to mention Agda, which has relatively fewer codes. In the work of Jiang et al. [1], they used a large language model, the GPT-4, to convert the formal language expressions of Isabelle and Lean into English to constructe a dataset containing 332K informal-formal pairs. Compared to using well-trained mathematics and computer experts to perform manual translation to build a dataset, this method is quite cheap and convenient, saving a lot of time and money. However, the disadvantages are also difficult to ignore. First, the accuracy of the translation results based on the GPT-4 is not optimistic. Jiang et al. estimated that the accuracy of the MMA dataset they constructed was about 74% based on sampling, which is obviously insufficient for further fine-tuning the large language model []. Noise or even incorrect translation may cause the model to learn incorrect translation patterns, thereby affecting the overall performance of the model. In addition, the risk of hallucinations in the data output by large language models cannot be ignored []. Their subsequent experimental results also verified these shortcomings well. The large language model fine-tuned on MMA only produced 29-31% of acceptable sentences.

Different from the method of constructing MMA, though *informath* project, this paper uses the collected Dedukti formal language expressions from different sources as an intermediary to convert them into formal languages Agda, Coq, and Lean expressions, and generate diverse and controlled natural language expressions in three natural languages: English, French, and Swedish, and constructing a dataset (Synthetic Multilanguage Autoformalization Dataset, SMAD) containing approximately 30K informal-formal pairs. This means that each group of data in the dataset consists of four formal languages with the same meaning and different expressions in three corresponding natural languages (4 to N). Most importantly, with the help of *Informath*, we can ensure that each informal-formal pair in the dataset is fully correct, which can improve the quality of subsequent fine-tuning of LLMs.


```
{"dedukti": "prop10 : Proof (even 0) .", "agda": "postulate prop10 : even 0", "coq": "Axiom prop10 : even 0 .", "lean": "axiom prop10 : even 0",
```


Table 1: Examples of SMAD

In this work, we fine-tuned the open-source large language model Qwen2.5-7B[] on SMAD and established an autoformalization system for Agda for the first time. To ensure the reliability and generalization of model testing, we used Dedukti code from different sources to the training set data to construct the test set. The Blue-4 Score of the fine-tuned model on the test set reached 76.16, which was 48.75% higher than the score of 51.20 of the un-fine-tuned baseline model. Other metrics including ROUGE-1/2/L also improved by 18.57%/40.44%/38.43% respectively compared with the baseline model. At the same time, we also checked the syntax correctness of the Agda code generated by the model. Compared with the error rate of the baseline model as high as 83.76%, the syntax error rate of the fine-tuned model dropped sharply to less than 8%. In addition, we also verified whether joint training of multiple formal languages and/or multiple natural languages can help improve the quality or training efficiency of autoformalization of Agda, a formal language with a shortage of code. Experimental results show that multi-formal/multi-natural language joint training can effectively improve various indicators of the model when the number of Agda-Eng samples is small, among which the reduction of syntax error is the most obvious.

2. Background

Agda. Agda [3] is a dependently typed functional programming language. Due to strong and dependent typing, Agda can also be used as a proof assistant, allowing proofs of mathematical theorems and running proofs such as algorithms. It has many similarities with other functional programming languages and interactive theorem provers based on dependent type theory, such as Coq and Lean. Compared to more popular formal languages, Agda has relatively little code. This poses a serious challenge to building a high-quality informal–formal pairs dataset for machine learning.

Dedukti. Dedukti is a logic framework based on the modular theory of the λΠ-calculus, in which many theories and logics can be expressed. λΠ-calculus modulo is a dependently typed λ-calculus with type rewriting rules added, which can express proofs in the form of modular calculus. In the Informath project, Dedukti is used as an interlingua to convert code into Agda Coq and Lean expressions.

Grammatical Framework (GF). It provides a framework for defining abstract syntax trees that map to multiple concrete natural languages, thereby supporting translation, parsing, and generation between languages. GF is particularly powerful in controlled natural languages and improves the interpretability of machine translation systems by minimizing ambiguity.

Informath. The *Informath* [4] project uses GF to solve the problem of converting mathematical expressions between multiple formal languages and multiple natural languages. The core structure of Informath is a bidirectional pipeline: formal language (Agda Coq Lean) ↔ Dedukti ↔ MathCore ↔ natural language. Dedukti acts as an interlingua for formal languages, while MathCore acts as an interlingua for natural languages. Through MathCore, Informath

can generate multiple controlable statements for multiple natural languages based on given formal expressions. Therefore, Informath provides a reliable and diverse dataset of multilingual expressions of formal-informal pairs without relying on manual translation or LLM-based informalization.

3. Problem Statement

This paper aims to build an autoformalization system for Agda by fine-tuning LLMs. At the same time, we also aim to verify whether joint training of multiple formal languages can help improve the quality or training efficiency of LLMs in translating Agda. Similarly, does joint training of multiple natural languages help? For this purpose, we build a parallel dataset (SMAD) containing 4 formal languages and 3 natural languages. Our dataset is different from MMA in that it does not come from text generated by LLMs, but relies on multi-language generation from the GF-based *Informath* project. Different slicing strategies on SMAD allow us to conduct experiments such as "all formals languages - English" vs. "Agda - English" (multiple formal languages vs. single formal language) and "Agda - all naturals languages" vs. "Agda - English" (multiple natural languages vs. single natural language). We also study the impact of the scenario of scarce Agda code resources on the fine-tuned model by downsampling the Agda-English data. This can also verify whether adding "other formal language-natural language pairs" can improve the Agda autoformalization ability of the fine-tuned model. Finally, we also tested the informalization ability of the fine-tuned model, that is, input formal statements into the model and check whether the output natural language is consistent with the original meaning and whether it is fluent and natural.

4. Methodology

Datasets and experimental groups. The data involved in the experiment are all from the self-constructed SMAD dataset. The naming convention for the datasets used to train/test the models is as follows:

- **SMAD**: The name of the dataset.
- **train/test/all**: Specifies that this is a train/test/non-splited set.
- **full**: Includes all formal language - natural language pairs.
- **eng**: Includes only formal language - English pairs.
- **agda**: Includes only Agda - natural language pairs.
- **agda_eng**: Includes only Agda - English pairs.
- **small (Optional)**: A subset of the dataset containing 10,000 training samples or 1,000 test samples. This slicing strategy is mainly used for small-scale testing and verification.

For an example：

- **M_SMAD_test_agda_eng_small**: A test set which belongs to a slice of the SAMD dataset, containing a small subset of only Agda–English pairs.

We set up six experimental groups (A-F) using different slices of the SMAD dataset according to the experimental purpose:

Experiment Group	Training Data Coverage	Purpose
A	Each FL (Dedukti/Agda/Coq/Lean) ↔ English trained independently (4 models)	Baseline: single formal language × English
B	All FL ↔ English combined training (1 model)	Verify the effect of joint training of multiple formal languages
C	Agda ↔ Each NL (English/French/Swedish) trained independently (3 models)	Baseline: single natural language × Agda
D	Agda ↔ All NL combined training (1 model)	Verify the effect of joint training of multiple natural languages
E	All FL ↔ All NL (4×3 total 12 pairs) combined training	Full multi-pair model
F	Only Agda ↔ English, but split into smaller data-scarce scenarios (e.g., 10%, 1%) to simulate low-resource scenario	Verify the effect of data quantity vs. accuracy

These groups test the effect of multi-formal languages (A vs B) and multi-natural languages (C vs D) training, as well as scaling of data (A/D vs F experiments).

Training setup. We fine-tuned the pre-trained Qwen2.5-7B model using LLaMA-Factory on a single NVIDIA RTX 4090 GPU (24GB VRAM). We used LoRA [] fine-tuning: the original model weights were kept frozen while small low-rank matrices were added to each layer and learned. The LoRA core parameters we used were as follows: rank r = 8, LoRA alpha = 32, dropout = 0.1. We used a dynamic learning rate, initially set to 5e-5. Depending on the

size of the dataset, we set batch_size = 2 and gradient_accumulation_steps = 8, which is equivalent to a batch_size of 2*
8 = 16. We trained one to multiple epochs for different models so that the training time of each model was the same, making it easier and more fair to compare the results. For example, we trained the model using the largest set of slices (38,736 examples, all FL/all NL) for 1 epoch (2,421 steps); the smallest set (3,228 Agda-English) required 12 epochs (same 2,421 steps) to match the same training amount of the former.

Evaluation metrics. We evaluate the translation by comparing the formal language code output by the fine-tuned model with the test set text based on SMAD slices. We measure the quality of the model using BLEU-4 and ROUGE-1/2/L scores, which are standard measures of n-gram overlap in translation evaluation. BLEU focuses on measuring the accuracy and exact match of the translation, which is more biased towards Precision, while ROUGE focuses on measuring the information completeness and coverage of the summary, which is more biased towards Recall. We also measure the syntax error rate of the Agda code generated by the model: the proportion of outputs that cannot be parsed into syntactic correct formal sentences. Finally, to summarize the performance, we use a custom score defined as:

```
Score = 0.35*(1 – ERROR%) * 100 + 0.35*BLEU-4 + 0.1*(ROUGE-1 + ROUGE-2 + ROUGE-L).
```

This balances the paper quality of the translation and the syntax correctness that is critical for practical applications. We use these metrics to compare models and analyze the results of the experiments.

5. Result and Discussion

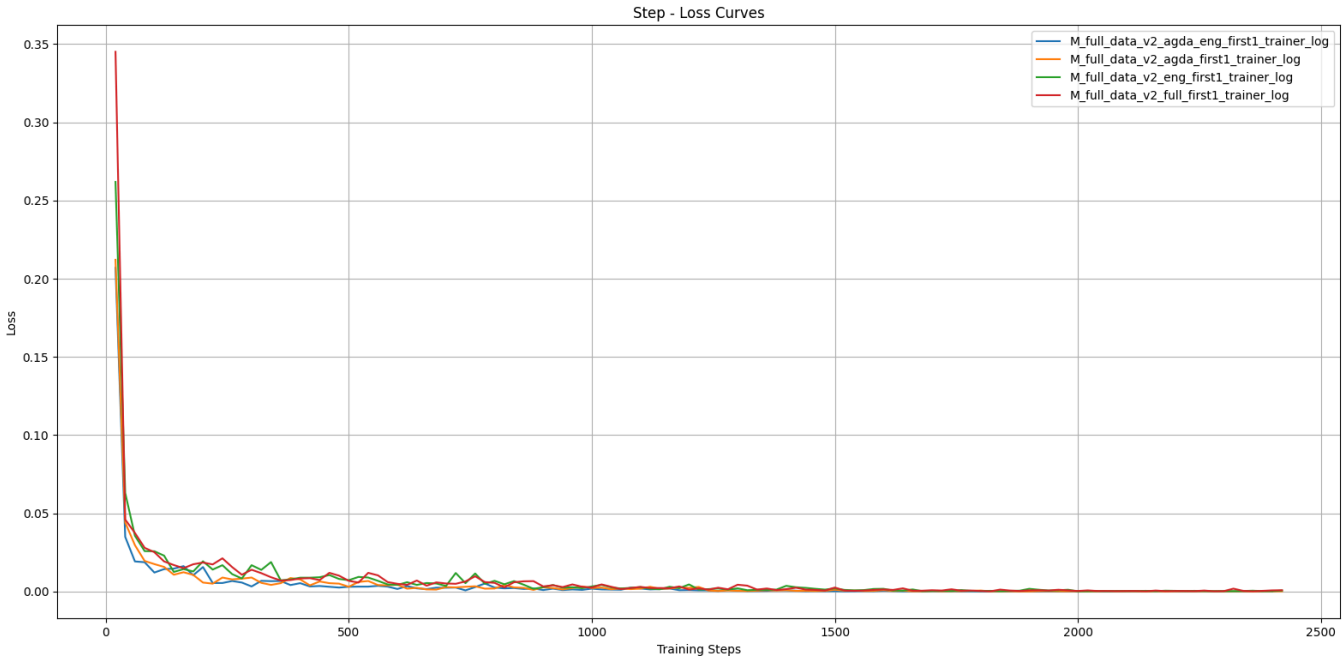

Figure 1: Training losses of models using different training sets sliced from SMAD.

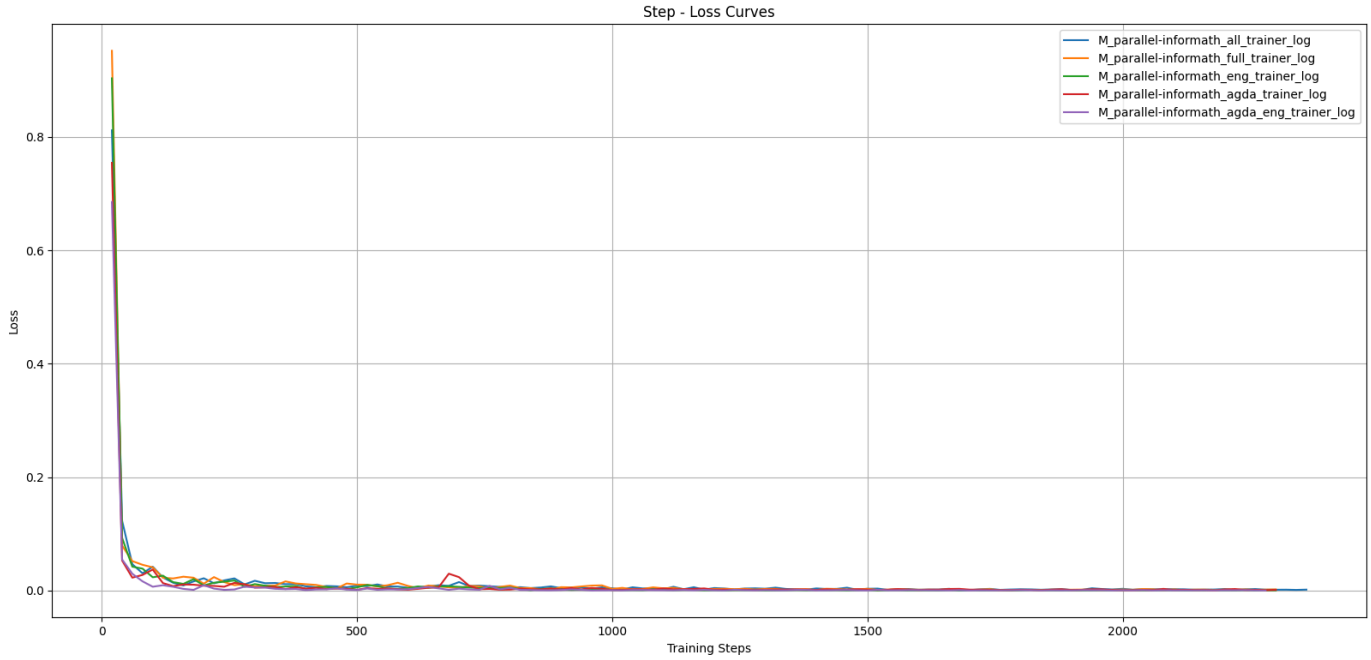

Figure 2: Training losses of models using different training sets sliced from parallel-informath (data from another resourse).

No.	Model Name	BLEU-4	BLEU-4 Δ%	ROUGE-1	ROUGE-1 Δ%	ROUGE-2	ROUGE-2 Δ%	ROUGE-L	ROUGE-L Δ%	Syntax Error
1	Base Line	51.20	0.00%	75.08	0.00%	53.36	0.00%	60.12	0.00%	294
2	M_full_data_v2_full_first1 (38,736)	98.36	92.00%	99.31	32.26%	98.94	85.42%	98.81	64.36%	14
3	M_full_data_v2_eng_first1 (12,912)	98.63	92.57%	99.48	32.49%	99.13	85.77%	98.99	64.66%	13
4	M_full_data_v2_agda_first1 (9,684)	98.98	93.28%	99.64	32.71%	99.48	86.44%	99.32	65.21%	17
5	M_full_data_v2_agda_eng_first1 (3,228)	98.91	93.15%	99.54	32.57%	99.26	86.03%	99.27	65.13%	19
6	M_parallel-informath_full (14,748)	88.98	73.79%	92.28	22.91%	85.85	60.89%	91.92	52.90%	67
7	M_parallel-informath_eng (4,916)	88.61	73.10%	92.53	23.24%	87.08	63.19%	92.04	53.09%	87
8	M_parallel-informath_agda (3,687)	88.30	72.45%	91.04	21.26%	85.33	59.91%	91.06	51.47%	159
9	M_parallel-informath_agda_eng (3,228)	85.71	67.38%	89.62	19.36%	85.14	59.56%	89.53	48.92%	114

Table 2: Evaluation on Agda–English test set (BLEU-4, ROUGE-1/2/L, syntax-error%), comparing baseline and finr-tuned models.

By comparison, we can find that:

- **Baseline model vs. fine-tuned models:** (1 vs 3/9) For the un-fine-tuned model, the BLEU-4 score on the test set reached 51.20, but the syntax error rate was as high as 83.76%, which is obviously unusable, and the overall score was 42.46. In contrast, even the worst fine-tuned model achieved a BLEU-4 score of 85.71, an improvement of 67.38% over the baseline, and the syntax error rate also dropped sharply to 13.04%, with an overall score of 86.86, an improvement of 104.57%. For the best model, the syntax error rate dropped to only 3.62%.

- **Effect of joint training:** (2 vs 5) Although multi-formal/multi-natural language joint training has little help on the main indicators when the amount of data is large enough, on the contrary, multi-task joint training has learned how to translate other formal/natural language without significantly damaging the performance of main target (translate between Agda-English), and the training cost (number of steps/time) has nearly no increase.
- **Training under Low-resource:** (6 vs 9) Multi-formal/multi-natural joint training can effectively improve various indicators of the model when the number of Agda-Eng samples is small, among which the reduction of Syntax Error is the most obvious.

6. Conclusion

We achieved excellent BLEU, ROUGE scores and surprising grammatical error rates by fine-tuning Qwen2.5-7B using LoRA on the SMAD dataset. Our work is the first to incorporate Agda into fine-tuning large language models for autoformalization. At the same time, experiments also show that: for the case of scarce Agda resources, multi-language joint training can slightly improve the performance of the model. For the case of abundant data, multi-language joint training greatly improves the efficiency of training without losing training quality. The time originally dedicated to training a single formal language can be used to train 4 different formal languages simultaneously.

7. Reference

1. Jiang, A. Q., Li, W., Jamnik, M. (2023). Multilingual Mathematical Autoformalization. *arXiv preprint arXiv:2311.03755.*
2. Wu, Y., Jiang, A. Q., Li, W., Rabe, M. N., & Szegedy, C. (2022). Autoformalization with Large Language Models. *arXiv preprint arXiv:2205.12615.*
3. Norell, U., James C. (2009). Dependently typed programming in Agda. *Proceedings of the 4th international workshop on Types in language design and implementation*.
4. Ranta, A. (2024). Towards Multilingual Autoformalization and Informalization of Mathematics. *Proceedings of the Swedish Language Technology Conference (SLTC) 2024.*