Planning Report: Autoformalization for Agda

1. Preliminary Title

Bridging Multiple Natural Languages and Agda: Autoformalization Using Large Language Models

2. Background

Relevance and Motivation
Formalizing mathematics into code (e.g., Agda[1]) is critical for theorem proving, verification, and AI-driven reasoning. However, manual formalization is time-consuming and error-prone. While autoformalization has advanced for systems like Lean and Isabelle, Agda—a dependently typed language with strict syntax—remains understudied. This project addresses this gap by developing a bidirectional translation system between natural language (e.g., English and Swedish) and Agda using LLMs. The system integrates both large language models (LLMs) and Grammatical frameworks (GF) to generate synthetic data. This dual approach aims to mitigate the data scarcity issue inherent to Agda and to enhance the robustness of the translation. The success of this project would lower the barrier to using proof assistants, improve educational tools, and streamline mathematical research workflows.

3. Aim

Primary Objectives
1. Construct datasets of aligned natural language and Agda code pairs using both LLM-assisted synthesis and computational grammar-based data generation.
2. Fine-tune multilingual LLMs (e.g., Llama 3.2) for bidirectional translation (formalization and informalization).
3. Evaluat system performance with robust metrics and benchmarks.
4. Develop a prototype system to demonstrate practical usability.

4. Problem Formulation

Extended Scientific Problem Definition
Autoformalization for Agda faces three core challenges:
1. **Data Scarcity**: Limited aligned datasets for Agda hinder supervised learning.
2. **Bidirectionality**: The system must handle both formalization (natural language to Agda) and informalization (Agda to natural language).
3. **Multilingual Support**: Supporting multiple natural languages (e.g., English and Swedish) further complicates the parsing and translation process.

Research Questions
- How can synthetic datasets be generated and validated for Agda autoformalization?
- Can LLMs generalize to Agda's syntax and semantics without domain-specific pretraining?

- What metrics best evaluate the accuracy and usability of bidirectional translation systems?

Theoretical Context
Previous work, such as the Informatah project, demonstrated the effectiveness of symbolic informationization with the Grammatical Framework (GF) in translating between natural language and formal languages, with supporting multiple languages [2]. In addition, Autoformalization with Large Language Models shows that automated formalization using LLMs of informally given natural language statements is generally possible, even under the condition that the language models not trained for the Autoformalization task [3]. Similarly, Multilingual Mathematical Autoformalization (MMA) datasets show the efficacy of training LLMs for tasks in various formal systems, including Isabelle and Lean [4].

5. Limitations

Certain aspects will not be covered in this thesis due to time and computational constraints: 1. **Formalization Depth**: The project focuses on translating statements rather than generating full Agda proofs.
2. **Languages Coverage**: Prioritize English and Swedish; other languages may be excluded due to resource constraints.
3. **Text Domain**: The focus is on mathematical statements rather than arbitrary text. 4. **Computational Resources**: Training large-scale LLMs may be limited by available hardware and time.

Rationale
Limitations are imposed to ensure feasibility within the MSc timeframe while delivering a functional prototype.

6. Methodology

Workflow Overview
1. **Dataset Construction**
- **Symbolic Informalization (GF)**: - Adapt formal system data (e.g., Lean, Coq or Isabelle) to Agda syntax, enriching the current dataset. - Use symbolic informalization with Grammatical Framework (GF) to create natural language sentences equivalents to formal Agda expressions. - **LLM-Assisted Formalization**: Use GPT-o1, DeepSeek or other LLMs to generate to generate informal data from formal Agda code, creating another dataset of aligned formal-informal pairs.
- **Data Augmentation**: Expand datasets using back-translation and MMA dataset integration.

2. **Model Training**
    - Fine-tune Llama 3.2 both constructed datasets, incorporating examples from the MMA dataset for better performance and multi-language support.

- Employ back translation techniques to enhance the quality and diversity of the training data.

- Train separate models on the two datasets to compare their performance.

3. **Prototype Development**
   - Design an interactive system that allows users to input natural language and receive Agda code suggestions.

   - Integrate Agda's type checker for real-time validation.
4. **Evaluation**
   - **Metrics**: BLEU, Levenshtein distance, type-check success rate.

   - **Qualitative**: User surveys with Agda developers to assess usability.

   - **Comparative Analysis**: Compare our models with existing systems like MMA-trained models.

7. Risk Analysis and Ethical Considerations

Technical Risks
- **Data Quality**: Generated data may contain errors, impacting model performance.
Mitigation: Manual validation by Agda experts.
- **Compute Limitations**: Training LLMs requires significant GPU resources.
Mitigation: Optimize training pipeline and explore cloud-based resources. - **Time Management**: Delays in experiments or data preparation may impact project milestones.
Mitigation: Regular progress checks and contingency plans.

Ethical Risks
- **Data Privacy**: Using only publicly available datasets to avoid proprietary information issues. - **Transparency**: Clearly documenting dataset sources and model limitations.

8. Time Plan

Week	Topic	Details
1	Dedukti	Learn and install tools.
2	Agda	Familiarize with syntax, semantics, and tools.
3	GF	Install and learn Grammatical Framework for symbolic informalization.
4-5	Machine Learning Tools	Install and explore relevant ML technologies, including API pipelines and Llama models.

Week	Topic	Details
6	Hardware Needs	Identify and set up necessary computational resources.
7-9	Dataset Preparation	Collect, augment, and prepare training, validation, and test datasets.
10-12	Experiments	Train models on datasets, fine-tune, and debug issues.
13-15	Evaluation	Evaluate models using MMA datasets and additional metrics.
16-20	Report Writing	Document findings, write final thesis report, and prepare presentation.

Milestones

- **Week 6**: Completion of tool installations and hardware preparation.
- **Week 9**: Dataset preparation finalized.
- **Week 12**: Experiments completed.
- **Week 15**: Evaluation and comparative analysis completed.
- **Week 20**: Final thesis report submitted.

References

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