Date: 2024/12/11

Master Thesis Project Proposal

Autoformalization for Agda

|  |  |
| --- | --- |
| *Name and email address of student 1:*  *Pei Huang, peih@student.chalmers.se* | *Completed courses relevant for thesis work:*  *Deep machine learning - SSY340*  *Design of AI systems - DAT410*  *Applied machine learning - DAT341*  *Introduction to data science and AI - DAT565* |
| *Name and email address of student 2:* | *Completed courses relevant for thesis work:* |

|  |  |
| --- | --- |
| *Suggested CSE supervisor (if you have one, otherwise skip this row)* | *Aarne Ranta* |
| *Company supervisor (if applicable)* | *Name of supervisor, name of company* |

**Introduction**

The formalization of mathematics has become an essential component in advancing fields like theorem proving, program verification, and artificial intelligence. However, manual formalization, which involves translating natural language mathematical descriptions into formal mathematical code, remains labor-intensive and error-prone. Autoformalization, the automation of this translation process, has recently gained attention as a potential solution.

This thesis aims to address the challenge of autoformalization for Agda by leveraging large language models (LLMs) such as Llama to enable bidirectional translation between Agda and natural languages (e.g., English and Swedish). his not only lowers the barrier to entry for using proof assistants but also supports broader applications such as educational tools, mathematical research, and automated theorem proving. Achieving this goal would mark a significant advance in the domains of formal methods, automated theorem proving, and human-computer interaction in mathematical reasoning.

**Context**

Previous work, such as the *Informatah* project, demonstrated the effectiveness of symbolic informationization with the Grammar Framework (GF) in translating between natural language and formal languages, with supporting multiple languages ​​[1]. 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 [2]. Similarly, Multilingual Mathematical Autoformalization (MMA) datasets show the efficacy of training LLMs for tasks in various formal systems, including Isabelle and Lean [3].

Despite these advancements, Agda has been relatively underexplored in autoformalization research. Most efforts in autoformalization focus on proof systems like Lean or Isabelle, leaving a significant gap for Agda users. This thesis leverages and extends the state of the art by targeting Agda with fine-tuned LLMs and curated datasets for Agda-specific autoformalization, ensuring compatibility with its syntax and semantics.

**Problem**

Translating informal mathematical statements into formal Agda expression faces unique challenges:

1. **Data scarcity:** Agda lacks a sufficiently large, high-quality dataset of aligned informal and formal expressions.
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.

**Goals and Challenges**

The primary **goals** of this thesis are:

1. **Dataset Construction**: Build two large-scale datasets of paired natural language descriptions and Agda code by using GF based symbolic informalization and LLMs for reverse translation (formal-to-informal) with manual verification.
2. **Model Training and Fine-tuning**: Adapt state-of-the-art LLMs to support Agda-specific autoformalization through fine-tuning on the constructed dataset.
3. **Multilingual Support**: Fine-tune a multilingual LLM to support English, Swedish and other languages as inputs.
4. **Model Performance Comparison**: Evaluate and compare the performance of models trained on the two datasets, assessing their strengths and weaknesses in different scenarios.

The key scientific and engineering **challenges** include:

1. **Agda's Syntax Complexity**: Managing Agda’s strict syntax and type system during natural-to-formal translation.
2. **Data Scarcity**: Overcoming the limited availability of Agda-specific datasets by generating and validating synthetic data.
3. **Evaluation Metrics**: Ensuring the accuracy and usability of the autoformalization system through robust evaluation.
4. **Resource Requirements**: Addressing the computational demands of training and fine-tuning large-scale language models efficiently.

**Approach**

This project adopts a data-driven and iterative approach to address the identified challenges.

**Dataset Construction:**

1. **Symbolic Informalization with GF**:
   1. Adapt formal system data (e.g., Lean, Coq or Isabelle) to Agda syntax, enriching the current dataset.
   2. Use symbolic informalization with Grammatical Framework (GF) to create natural language sentences equivalents to formal Agda expressions.
2. **LLM-assisted Informalization**: Use LLMs (e.g. GPT-4o, GPT-o1, etc.) to generate informal data from formal Agda code, creating another dataset of aligned formal-informal pairs.
3. **Integration with MMA**: Incorporate MMA datasets for additional multilingual and multidomain coverage.

**Model Training:**

1. Fine-tune an open-source and high-capacity LLM, such as Llama 3.2 on both constructed datasets, incorporating examples from the MMA dataset for better performance and multi-language support
2. Employ back translation techniques to enhance the quality and diversity of the training data.
3. Train separate models on the two datasets to compare their performance.

**Prototype Development:**

Design and implement a prototype system with the following features:

* **Interactive Interface**: Enable users to input natural language queries and receive Agda code suggestions.
* **Error Checking**: Integrate Agda’s checker to validate generated code, ensuring syntax and content correctness.

**Evaluation:**

1. **Accuracy**: Measure the proportion of Agda code that translated successfully.
2. **Translation Quality**: Use metrics like BLEU Perplexity score and Levenshtein edit distance, to assess the quality of generated translations.
3. **Benchmarks**: Test the system on custom Agda code sets.
4. **Usability Studies**: Conduct surveys with Agda users to assess ease of use and satisfaction.
5. **Comparative Analysis**: Compare our models with existing systems like MMA-trained models.

**References**

1. Ranta, A. (2024). Towards Multilingual Autoformalization and Informalization of Mathematics. *Proceedings of the Swedish Language Technology Conference (SLTC) 2024.*
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. Jiang, A. Q., Li, W., Jamnik, M. (2023). Multilingual Mathematical Autoformalization. *arXiv preprint arXiv:2311.03755*.