PROOF RECOMMENDATION FOR THE HOL4 THEOREM PROVER

Nour Dekhil

A thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Applied Science
(Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

December 2024

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Nour Dekhil**

Entitled: **Proof Recommendation for the HOL4 Theorem Prover**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair
Dr. Manar Amayri

_____ External Examiner
Dr. Manar Amayri

_____ Internal Examiner
Dr. Xinxin Zuo

_____ Supervisor
Dr. Sofiène Tahar

Approved by _____
Dr. Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_____ 2024

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Proof Recommendation for the HOL4 Theorem Prover

Nour Dekhil

Concordia University 2024

Interactive theorem proving is a complex process that often requires significant expertise, user intervention and deep domain knowledge, making it challenging for users to construct valid proofs. The HOL4 theorem prover, while a powerful tool in formal verification, presents usability challenges due to the intricate nature of proofs and the cognitive load placed on users. This thesis proposes an innovative solution to enhance the accessibility and efficiency of interactive theorem proving through the development of an AI-driven Proof Recommendation System that leverages Large Language Models. The proposed methodology focuses on two primary tasks: proof step recommendation and complete proof generation. For the proof step recommendation, models such as BERT, RoBERTa, and T5 were fine-tuned on datasets derived from HOL4 theories to predict the next logical step(s) in the proof construction. This capability aims to guide users through the proof process, making it less daunting and more manageable, especially for those with a limited experience. In the proof generation task, sequence-to-sequence models, including MarianMT and T5, were utilized to generate complete proof sequences based on the given theorem statements. This task is particularly challenging due to a need to capture complex logical patterns and ensure the validity of the generated proofs. The training involved rigorous hyper-parameter tuning and evaluation to optimize the performance of models. Experimental results demonstrate that our proposed approach not only reduces the cognitive load on theorem provers but also enhances the efficiency and accessibility of interactive theorem proving compared to related work. The tool, called HOL4PRS, achieves significant accuracy in recommending proof steps and generating proof sequences, facilitating more widespread adoption of HOL4 in critical verification tasks across various industries. This thesis contributes to the field by showcasing how integrating AI into formal verification processes can significantly advance the capabilities and applications of the interactive theorem provers.

To my parents, Noureddine and Samah, and my sisters, Ella and Ranim for their sacrifices, support, and love throughout this journey.

Acknowledgments

In the name of Allah, the Most Gracious, the Most Merciful. All praise and infinite gratitude are due to Allah, who granted me the strength, patience, and wisdom to complete this journey.

First and foremost, I am deeply grateful to my supervisor, Dr. Sofiene, for his insightful guidance, invaluable feedback, and encouragement throughout my Master's program. His expertise and dedication have been an inspiration, and I am truly fortunate to have worked under his supervision. I am grateful to Dr. Adnan for his mentorship and motivation throughout my research. A special acknowledgment to Dr. Maissa for believing in me and making this opportunity possible.

I sincerely thank Dr. Manar Amayri and Dr. Xinxin Zuo for kindly agreeing to serve on my thesis examining committee and dedicating their time to review my work.

To my HVG family, Amira, Kubra, Alain, Oumaima and Elif, thank you for making even my toughest days brighter with your warmth and kindness. Each of you has taught me valuable lessons in resilience, hard work, and perseverance that I will carry with me for a lifetime.

This accomplishment is a reflection of not only my efforts but also the support and love of everyone who stood beside me from friends and family. I am deeply grateful to each and every one of you.

Finally, to my parents and sisters, who have been my greatest supporters from afar, your endless prayers and belief in me have been my greatest source of strength. This achievement is as much yours as it is mine. A special place in my heart is reserved for my aunt Imen and my little cousins, Syrine, Adam and Ayoub. Their warmth and presence have turned Canada into a second home.

Table of Content

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ATP	Automated Theorem Proving
AST	Abstract Syntax Trees
BERT	Bidirectional Encoder Representations from Transformers
DPR	Dense Passage Retriever
FOL	First Order Logic
HOL	Higher Order Logic
HTPS	HyperTree Proof Search
HTM	Hierarchical Transformer Model
HVG	Hardware Verification Group
ITP	Interactive Theorem Proving
k-NN	k-Nearest Neighbors
LLM	Large Language Model
LSTM	Long Short-Term Memory
LSH	Locally Sensitive Hashing
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Process
MLM	Masked Language Modeling
MML	Mizar Mathematical Library
NLP	Natural Language Processing
NSP	Next Sentence Prediction
RNN	Recurrent Neural Network
T5	Text-to-Text Transfer Transformer
UCT	Upper Confidence Bounds applied to Trees

Chapter 1

Introduction

1.1 Motivation

We depend today upon technology in all walks of life. The reliability and correctness of the software and hardware systems have become extremely critical. In fact, high stake industries, including aerospace, automotive, medicine and finance, need flawless systems to prevent very expensive failures or even potentially disastrous outcomes (e.g., [1, 2, 3]). Formal verification [4] is a computer-based method in which the correctness of systems is mathematically proved and is rapidly becoming one of the most important ways of validating such systems. One of the most widely used formal verification method is theorem proving [5], which is based on developing a computer-based mathematical model of a system and ensures by using deductive reasoning that the given system behaves according to specifications under any condition. It offers mathematical assurance of correctness for proving that software or hardware behaves as it should according to its specification, thus becoming quite crucial in safety-critical settings.

Theorem proving is the process of establishing the correctness of a statement by a sequence of logical deductions from a set of pre-defined axioms and inference rules. Theorem proving thereby provides a mathematical framework to ensure that systems satisfy their specifications under all possible conditions. In contrast to empirical testing, which justifies system behavior under certain conditions for a bounded set of

inputs, theorem proving provides a universal and rigorous approach to verify properties of correctness, in particular for systems belonging to the most critical domains.

There are two broad categories of theorem proving [5]: Automated Theorem Proving (ATP) and Interactive Theorem Proving (ITP). ATP involves the use of algorithms and software to automatically generate proofs for given theorems without human intervention. This approach is typically faster and can handle large search spaces efficiently, making it suitable for problems where quick verification is essential. However, its reliance on predefined strategies can limit its effectiveness in tackling complex theorems that require nuanced reasoning or creative problem-solving. On the other hand, ITP supports powerful reasoning logics, such as higher-order-logic (HOL) [6] but involves a more hands-on approach where users actively participate in the proof construction process. Users select and apply *tactics*, guiding the proof development through a series of logical steps. This method allows for greater flexibility and an ability to tackle intricate proofs.

Various ITP systems/tools have gained wide acceptance in academia and industry over time. To that end, systems like HOL4 [7], Coq [8], Isabelle [9], Lean [10], PVS [11], Mizar [12], and Metamath [13] have become integral parts of efforts needed to establish the correctness of complex systems. For instance, PVS has been used by NASA for several decades in the formal verification of aerospace systems, such as the pioneering work on the formalization of space shuttle software requirements [14]. HOL4 has also been widely used to perform hardware verification [15, 16, 17], where it ensures that processors and all such important components behave correctly before deployment. Similarly, Coq has been applied in compiler certification (e.g., [18, 19]), while Isabelle has been used in the protocol verification of security systems (e.g., [20, 21]). These tools are designed to develop proofs that are accurate and trustworthy; therefore, they are applied in cases where a single failure may have disastrous consequences. While this is going on, all these capabilities make ITP prone to some challenges too, turning the process of theorem proving quite hard in general, and only experts can do it. It requires significant user involvement and expertise, which can make the process more time-consuming and challenging for novice users. Ultimately, the choice between ATP and ITP hinges on the specific requirements of the task at hand, balancing the need for efficiency with the depth of understanding and engagement in the verification process. This demand for expertise and time, in

the context ITP, pose challenges, particularly for those new to the field or in industrial settings where resources may be limited. As systems grow in complexity, the need for effective support mechanisms to assist users in managing this intricate process becomes increasingly crucial.

Recent advances in Artificial Intelligence (AI) open promising avenues toward supporting and enhancing ITP to meet these challenges [22]. Instead of relieving expertise, AI tools are being designed more and more to complement human insight with targeted assistance that can reduce the burden of proof construction. The AI-driven models learn patterns from large datasets of previously completed proofs so that the next-logical-step prediction can be easier and less time-consuming to generate such a proof, even with a limited theorem-proving experience.

Through the use of advancements in Machine Learning (ML) [23] new pathways have opened for automating and supporting the proof construction process in ITP. By providing intelligent suggestions and automating repetitive tasks, these innovations aim to reduce the manual effort required in ITP, thereby making it more accessible to a broader audience. Consequently, ITP may continue to be recognized as one of the most powerful proof assistants, poised to play a pivotal role in advancing formal verification across various industries.

1.2 Problem Statement

In interactive theorem proving, theorems often require intricate, multistep proofs that demand a deep understanding of both the underlying theory and the specific characteristics of the system in question. This complexity not only makes the proof construction process time-consuming but also increases the likelihood of errors, as users must navigate vast search spaces of potential proof strategies. Consequently, the reliance on human expertise becomes a significant barrier, limiting the accessibility of theorem proving tools to a select group of highly skilled individuals.

Existing theorem proving environments, such as HOL4, Isabelle, and Lean, while powerful, often lack the necessary support mechanisms to assist users in efficiently constructing proofs. The interactive nature of these systems requires users to select and apply various tactics, which can be overwhelming, particularly for those with limited experience. As a result, the learning curve associated with theorem proving

can discourage new users and slow down the verification process, ultimately impacting the development timelines of critical systems.

There exist various theorem provers that have been applied with a great success in the fields of hardware verification, software certification and cryptographic protocol analysis, where correctness is a major concern. The most pioneering among them is HOL4, which belongs to a lineage of theorem provers that traces its origins to the LCF (Logic for Computable Functions) theorem prover, developed in the 1970s. HOL4 has been instrumental in the formal verification of significant theories and systems, such as the seL4 microkernel [29], which is renowned for its safety and security in critical applications, the CompCert C compiler [32], which guarantees the correctness of compiled code, and more recently critical smart grid systems [24] and distributed multiprocessor systems [25]. These accomplishments not only showcase HOL4's effectiveness in high-assurance domains, particularly in hardware design, but also highlight its superiority over other interactive theorem provers in tackling complex engineering verification challenges.

At the Hardware Verification Group (HVG)[1] of Concordia University, we have been using HOL4 for the past 25 years on projects spanning from microelectronics hardware to embedded systems and software. HOL4 was also used for the formal reliability analysis of safety-critical and cyber-physical systems. Moreover, a considerable amount of effort has been spent in developing fundamental libraries of intricate mathematics such as measure, probability and information theories. One common feature we have learned from these works is the repetitive nature of many proofs conducted in HOL4 interactively. The expertise gained in our laboratory over the years as well as the availability of a large corpora of proof scripts provides us with a unique source of datasets that is ideal for applying AI-driven methods on HOL4.

Despite the remarkable progress in AI and machine learning, there remains, however, a significant gap in their effective integration within the HOL4 theorem prover. Current strategies in HOL4 fall short in automating the proof construction process and providing meaningful guidance to users in selecting optimal tactics. This lack of automation not only intensifies the challenges of accessibility and efficiency but also restricts the broader adoption of HOL4 in critical applications. Consequently, users with limited experience often find the intricacies of proof construction daunting,

[1] `https://hvg.ece.concordia.ca`

which can discourage their engagement with this powerful verification tool. To fully realize the potential of HOL4 in formal verification, it is imperative to address these shortcomings, thereby enhancing its usability and fostering its application across a wider range of industries.

This thesis builds on these developments by creating a system specifically designed for the HOL4 theorem prover that goes beyond mere proof recommendations to enable both proof sequence generation based on a given theorem statement and suggestions for the next tactics within an ongoing proof. Leveraging Large Language Models (LLMs) [26], this system provides targeted recommendations for the next steps in a proof sequence by analyzing the theorem statement and applying previously established tactics. The model learns complex proof patterns specific to HOL4 by training on large libraries of HOL4 proof scripts. It can thus predict appropriate optimal tactics with much accuracy and consistency. The approach proposed here analyzes the current state of a proof, comprising the sequence of tactics applied so far in order to suggest various possible next tactics to move the proof toward completion. Additionally, given a theorem statement, the system can propose a sequence of tactics that could be used to construct a proof, offering a broader and more automated solution to assist interactive theorem provers.

The research work aimed in this thesis would contribute to the rapidly developing landscape of AI-assisted theorem proving with a view toward ongoing advancement of innovations in many safety-critical domains where correctness is paramount, seeding ground for future innovation at the crossroads of AI and formal methods. In the next section, we provide an extensive review of related work in the domain in order to position the context of the proposed thesis research.

1.3 Related Work

There has been significant interest in the integration of AI into theorem proving in the past few years, with researchers investigating a variety of ways to automate and improve the process. These efforts are concerned with automating proof construction, reducing manual labor, and increasing efficiency in interactive and automated theorem proving systems. Among the explored techniques, proof step prediction and proof search are the most important approaches to help users build proofs or even partially

automate building proofs. This section will give an overview of some of the important works contributed in this regard, focusing on how machine learning, reinforcement learning [27], and other computational techniques have improved tackling challenges in formal verification and theorem proving.

1.3.1 Proof Step Prediction

Proof step prediction focuses on streamlining the theorem proving process by suggesting the optimal tactic(s) or proof step(s) within a given context. This approach enhances automation in interactive theorem proving by reducing the cognitive load and manual input required from users. Here, we review the significant contributions to the field, detailing various methodologies and models developed to automate and improve the tactic prediction.

Coq Proof Assistant

- **Huang et al.** [28] developed GamePad, a system that integrates machine learning with interactive theorem proving using the Coq proof assistant. This system advances the automation of theorem proving by predicting the next proof step and assessing how many steps remain to complete the proof, a process termed as evaluating the position within a proof. Specifically, GamePad inputs proof states from the Coq environment and outputs tactic predictions and step evaluations using a structured representation of proofs encoded as Python dictionaries and lists. The approach employs LSTM networks trained on a dataset which consists of 1,602 lemmas and expands into 83,478 proof states derived from the formalization of the Feit-Thompson theorem [29]. The GamePad system achieved a tactic prediction accuracy of 58.23% and a position evaluation accuracy of 65.30% on a testing dataset of 8,348 proof states, demonstrating its effective performance in automating theorem proving tasks.

- **Yang et al.** [30] developed CoqGym, a large-scale dataset containing 71,000 human-written proofs from 123 projects using the Coq proof assistant. Their goal was to assist theorem proving by predicting next proof steps and relevant lemmas. They introduced a deep learning-based model, ASTactic, which generates tactics as programs in the form of abstract syntax trees (ASTs). This

model was trained to transform input goals and premises into a series of tactical commands in Coq's language, aiming to automate the generation of proof steps traditionally crafted by human experts. ASTactic's approach was evaluated on a testing dataset comprising 13,137 theorems, where it successfully proved 12.2% of them, demonstrating its effectiveness in automating theorem proving processes.

- **Sanchez et al.** [31] developed Proverbot9001 specifically for the Coq proof assistant to reduce the manual effort typically required in proving software correctness by employing machine learning models to predict next proof steps. Proverbot9001 operates by taking theorems as input and outputs proofs, incorporating Feed-forward Neural Networks for tactic prediction and Recurrent Neural Networks (RNNs) for argument prediction. Additionally, the system employs advanced tree pruning techniques and a depth-first search strategy, guided by model predictions, to efficiently navigate the search space. Trained and tested on proofs from the CompCert C compiler, Proverbot9001 successfully automated proofs for 28% of theorem statements in a test dataset that included 501 theorems.

- **Blaauwbroek et al.** [32] developed a tactic prediction method for the Coq proof assistant with the aim to simplify the process of theorem proving by means of automatically predicting suitable proof steps. The system takes proof states as input and predicts suitable tactics to apply, therefore reducing manual effort in interactive theorem proving. Their approach combines a k-Nearest Neighbors (K-NN) [33] algorithm with Locally Sensitive Hashing (LSH) [34] to efficiently search a database of tactic applications to retrieve the most appropriate tactic for any given proof state. This system was evaluated on the Coq Standard Library using a dataset of 10,416 lemmas. The results showed the top predicted tactic accuracy to be 23.4% and an overall success rate of 39.3% in automatic lemma proving. Furthermore, together with CoqHammer [35], which is an external automated system, the success rate was 56.7%, which shows that learning-based tactic prediction really works well for enhancing automation in Coq.

- **First et al.** [36] developed TacTok, a tool to improve the automation of theorem proving in the Coq proof assistant through semantic-aware synthesis of proof scripts. The primary goal of TacTok is to enhance the interactive theorem proving process by predicting the next proof tactics based on both the partial proof script already written and the current proof state. The input to TacTok consists of these partial proof scripts and proof states, and the output is the next predicted tactic, effectively streamlining proof development. TacTok employs a beam search strategy [37], combined with a LSTM network, to utilize both proof state and proof script information. This method allows the system to explore multiple tactical possibilities efficiently, selecting the most promising paths for proof completion. Evaluated on a dataset of 26 software projects comprising over 10,782 theorems, TacTok demonstrated its robustness by outperforming existing tools and successfully proving 115 theorems that previous tools could not.

- **Luan et al.** [38] developed a framework for predicting tactics in the Coq proof assistant to automate proof step selection, reducing the manual effort required in constructing proofs. Their goal was to improve the proving process by using a Long Short-Term Memory (LSTM) neural network to predict appropriate tactics based on the current proof state. The input to their model includes hypotheses and proof goals, while the output is the predicted tactic to apply next. Recognizing the importance of a consistent proof style, they created a new dataset with a novice-proof approach, containing 31 theorems and lemmas in 830 lines of code. In their evaluation, top-1 accuracy refers to the percentage of times the model's first tactic suggestion was correct, achieving 58%, while top-3 accuracy indicates the model correctly predicted the tactic within its first three suggestions 87% of the time. These accuracies reflect significant improvements over the baseline methods with enhancements of 15.2% and 12.8% for top-1 and top-3 accuracies, respectively.

- **Wenda Li et al.** [39] proposed a benchmark called IsarStep for furthering automated theorem proving by predicting intermediate proof steps, namely generating missing propositions inside proofs. They aimed to mimic human reasoning in theorem proving by training models to infer such intermediate propositions that fill logical gaps between the given proof steps and conclusions. Given the

surrounding proof steps, the system is to output the intermediate propositions, which are necessary for continuity in proof. They implemented this by utilizing the Hierarchical Transformer Model (HTM), which has both local and global layers that learn intricate mathematical relations. Trained on a dataset of 204,000 lemmas from the Archive of Formal Proofs and evaluated on a testing dataset of 34,000 lemmas, HTM achieved top-1 accuracy rates of 15–25% in generating intermediate propositions, demonstrating its potential in proof step prediction within formal logic environments.

PVS

- **Yeh et al.** [40] developed CoProver, a recommender system for enhancing user interaction with ITP systems by applying proof step and lemma prediction. In this context, the system is meant to assist users in proof construction by suggesting appropriate commands and lemmas based on proof context. The input to CoProver consists of proof steps in sequences taken from the PVSLib-a [41] NASA database containing more than 184,000 polished proof steps-featurized into token sequences for training. For command prediction, CoProver uses a transformer-based RoBERTa model that encodes these proof states to capture prior proof command history for better prediction accuracy. It achieved an accuracy of 48% on command prediction, outperforming traditional baseline classifiers by a wide margin.

HOL Light

- **Bansal et al.** [42] developed a benchmark and learning environment to enable the automated formalization of large mathematical theories, utilizing the HOL Light theorem prover integrated within a reinforcement learning framework. The system, named DeepHOL, inputs theorem statements and outputs generated proofs. It employs a deep reinforcement learning strategy [43] combined with a neural architecture specifically designed for predicting theorem proving tactics and their arguments. The dataset used comprises 29,462 theorems and lemmata, derived from the formal proof of the Kepler conjecture and other foundational mathematics areas. Initial results of their system, DeepHOL, demonstrated proof success rates of up to 38.9%.

HOL4

- **Gauthier et al.** [44] developed an automated tactical prover named TacticToe within the HOL4 interactive theorem prover framework. This system learns from historical human proofs to predict and apply effective tactics in given proof states. The input to TacticToe consists of proof states, represented as sequences with sets of assumptions and conclusions, and the output is a sequence of proof tactics that lead to a theorem being proven. The approach combines K-NN for predicting tactics, theorems, and goal lists, with Monte Carlo tree search (MCTS) [45] to dynamically explore and optimize the proof search strategy. The dataset comprises 7,164 theorems from the HOL4 standard library. TacticToe demonstrates a significant ability to automate theorem proving, achieving a success rate of 66.4% on this dataset.

- **Wu et al.** [46] developed a reinforcement learning environment for HOL4 designed to predict effective tactics during proof searches. By framing theorem proving as a Markov Decision Process (MDP) [47], their system, TacticZero, enables efficient backtracking to abandon unproductive derivation paths and explore more promising alternatives. The model receives proof states as input and generates tactics and arguments to progress toward proving theorems. Trained on a dataset of 1,342 provable theorems from the HOL4 core library, with an 80-20 split between training and testing, TacticZero showed strong results. It notably outperformed traditional automated theorem provers, such as hammers [48], successfully proving 132 theorems.

Table 1.1 summarizes existing approaches to proof step prediction have made significant strides but reveal notable gaps when considered in the context of HOL4 and LLMs. While tools like TacticToe and TacticZero focus on HOL4, they rely on traditional methods like K-NN and reinforcement learning, which lack the contextual depth and sequence modeling capabilities of LLMs. Moreover, LLMs remain underexplored in interactive theorem proving, despite their proven ability to capture long-range dependencies and context in sequence-to-sequence tasks. Many works depend on limited or narrowly focused datasets, which constrain generalization and scalability, particularly for data intensive models like LLMs.

Table 1.1: Summary of Related Work in Proof Step Prediction

Tool	ML Approach	Dataset Size	Experimental Results
		Coq	
GamePad (2019)	LSTM	1.6K lemmas	58.23% tactic accuracy
CoqGym (2019)	DL	71K proofs	12.2% success
Proverbot9001 (2020)	FNN	501 theorems	28% success
Blaauwbroek et al. (2020)	K-NN, LSH	10.4k lemmas	23.4% accuracy
TacTok (2020)	LSTM, Beam Search	26 projects	115 new proofs
Luan et al. (2021)	LSTM	31 theorems	58% top-1 accuracy
IsarStep (2021)	HTM	204K lemmas	15-25% acc.
		HOL Light	
DeepHOL (2019)	DRL	29.5K theorems	38.9% success
		HOL4	
TacticToe (2020)	K-NN, MCTS	7.2K theorems	66.4% success
TacticZero (2024)	RL, MDP	1.3K theorems	132 proofs
		PVS	
CoProver (2023)	RoBERTa	184K proof step	48% command accuracy

1.3.2 Proof Search

Proof Search aims to learn from existing proofs to generate potential proof paths for given theorems. It uses advanced techniques to search sequences of tactic applications and logical steps in building complete proofs automatically. In the following, we reviewed some of the most prominent works that have been done in advancing proof search, highlighting different approaches and how effective these have been toward enhancing automated theorem proving.

Isabelle/HOL

- **First et al.** [49] developed Baldur, a tool designed to improve proof synthesis in theorem proving within the Isabelle/HOL theorem prover by generating entire proofs at once rather than using traditional step-by-step search methods. The goal of Baldur is to simplify and enhance the automation of formal proof verification, using LLMs like Minerva [50] to generate full proofs from theorem statements and, if needed, repair failed proofs based on error feedback. The input to Baldur's proof generation model is a theorem statement, and the output

is a complete proof. Using a dataset of 183,000 Isabelle/HOL theorems, including a test set of 6,336 theorems, the tool demonstrated strong results. Baldur achieved a proof success rate of 47.9% in generating correct proofs, outperforming previous search-based methods and further improving to a success rate of 65.7% when combined with the search-based tool Thor. This study illustrates the efficiency of using LLMs for whole-proof generation in automated theorem proving.

Metamath

- **Whalen et al.** [51] developed Holophrasm, an automated theorem prover designed to improve proof search for higher-order logic in the Metamath framework. The primary goal of Holophrasm is to generate formal proofs by efficiently exploring proof trees using deep learning without relying on hand-crafted features. The system inputs a theorem statement in Metamath's formal language and outputs a complete proof. Holophrasm employs a neural network augmented bandit algorithm, based on Upper Confidence Bounds applied to Trees (UCT) [52], to navigate the search space of partial proof trees, alongside a sequence-to-sequence model for action enumeration. The dataset used for training consists of the Metamath [53] set.mm module, with 21,786 theorems in the training set, 2,711 for validation, and 2,720 for testing. Holophrasm achieved a 14.3% success rate on its test set, demonstrating the potential of neural networks in guiding proof search within automated theorem proving.

- **Lample et al.** [54] developed Evariste, a system aimed at automating theorem proving in Lean, Metamath, and a custom environment by generating proof sequences with high efficiency. Their goal was to enhance the automation of formal proof synthesis. Evariste uses the HyperTree Proof Search (HTPS) algorithm [54], inspired by AlphaZero [55], to navigate proof trees. Inputs to the system consist of theorem statements, while outputs are generated proofs or tactics leading to solutions. Evariste's training involved over 37,000 theorems from Metamath's set.mm library and additional supervised training data for Lean, which allowed the system to achieve state-of-the-art results with a success rate of 82.6% on Metamath's held-out set of theorems.

Lean

- **Yang et al.** [56] presented LeanDojo, a system for automating theorem proving in the Lean proof assistant by improving proof search using retrieval-augmented language models. The focus of LeanDojo is to make the premises selection process easier and faster to find and use, with the goal of being able to perform theorem proving, which is a major bottleneck in the automation of proofs. The system takes the statements of theorems as input and returns a sequence of tactics that can be used in constructing a formal proof. LeanDojo uses the ReProver model, which is a retrieval-augmented language model that depends on the Dense Passage Retriever (DPR) [57] model for premise selection and an encoder-decoder Transformer model for tactic generation. The dataset for LeanDojo consists of 98,734 theorems and their corresponding proofs from Lean's math library, which was used to train and evaluate ReProver. Experimental results showed that ReProver proved 51.2% of theorems in a test set, outperforming non-retrieval baselines and achieving competitive results compared to state-of-the-art methods.

- **Song et al.** [58] developed Lean Copilot, a framework to assist in automated theorem proving within the Lean interactive theorem prover. The main goal of Lean Copilot is to enhance proof automation through tools that assist users in tactic suggestion, proof search, and premise selection. The system takes a theorem statement and proof goals as input and outputs a sequence of tactics or premises relevant for proof construction. Lean Copilot uses the ReProver model, based on the ByT5 encoder-decoder Transformer, with beam search to improve tactic suggestions and proof step prediction. The authors trained and evaluated Lean Copilot on Lean's math library, which contains over 98,000 theorems. In experiments on a subset of theorems from the Mathematics in Lean book, Lean Copilot's search proof tool demonstrated notable effectiveness by automating 64% of proofs autonomously and assisting users in automating 81.2% of proof steps, significantly outperforming existing rule-based tools.

Table 1.2: Summary of Related Work in Proof Search

Tool	ML Approach	Dataset Size	Experimental Results
		Isabelle/HOL	
Baldur (2023)	LLMs	183K theorems	47.9% proved, 65.7% with Thor
		Metamath	
Holophrasm (2016)	NN, UCT, Seq2Seq	21.8K theorems	14.3% success rate
Evariste (2022)	HTPS	37K theorems	82.6% success
		Lean	
LeanDojo (2023)	DPR, Transformer	98.7K theorems	51.2% success
Lean Copilot (2024)	ByT5, Beam Search	98K theorems	64% automation, 81.2% assisted steps

Table 1.2 provides an overview of existing proof search approaches, highlighting significant advancements while uncovering notable gaps, particularly in the context of HOL4 and LLMs. However, current proof search methods, while showing promising advancements, face limitations in adapting to HOL4 and usage of LLMs effectively. Many existing algorithms focus on proof tree exploration or retrieval augmented techniques, which are effective in systems like Lean or Metamath but are not directly applicable to the sequential and tactic-driven nature such as HOL4.

1.3.3 Premise Selection

Premise selection involves identifying the most relevant theorems or lemmas from a knowledge base to assist in proving a new conjecture. It reduces the complexity of theorem proving by narrowing down the search space for proof construction. Several approaches have been proposed in the literature, leveraging machine learning and deep learning techniques to improve the efficiency and accuracy of premise selection.

HOL Light

- **Kaliszyk et al.** [59] developed a method integrating machine learning with automated theorem proving within the Flyspeck project using the HOL Light proof assistant, with a focus on advancing the automation of theorem proving by predicting relevant premises for proofs. Specifically, their system inputs dependencies from a database of Flyspeck proofs and outputs premise selections to

assist ATPs in automated theorem proving tasks. The approach employs various machine learning techniques trained on a dataset that consists of 14,185 theorems, structured as proof dependencies within the vast mathematical knowledge encoded by the Flyspeck project. The learning-assisted system was evaluated on its ability to facilitate the proving process by selecting optimal premises, where it demonstrated a significant capability, managing to automatically prove 39% of the theorems in a push-button mode on a fourteen-CPU workstation, illustrating its effective performance in automating large-scale theorem proving tasks.

Mizar

- **Alemi et al.** [60] employed a two-stage deep learning framework that leverages neural sequence models to enhance the effectiveness of premise selection without relying on traditional hand-engineered features. Utilizing the Mizar Mathematical Library (MML), which contains 57,917 proved theorems organized into 1,147 articles, they implement the E prover as their ATP tool to facilitate the proof process. The first stage of their approach focuses on character-level models that treat mathematical formulas as sequences of characters, while the second stage builds upon these results with word-level models to capture more complex relationships. Their experiments reveal a success rate of 40% in automatically proving theorems, showcasing a significant advancement over previous methods.

Premise selection methods have contributed significantly to automating theorem proving by narrowing the search space for constructing proofs. However, in this thesis, we only focus on tactic prediction and proof step generation.

1.4 Proposed Methodology

The goal of this thesis is to develop an AI-driven system to enhance theorem proving in the HOL4 environment by addressing two main tasks: proof step recommendation and proof generation. This approach aims to make the theorem proving process less daunting and more accessible, especially for users who may not have extensive expertise in formal verification.

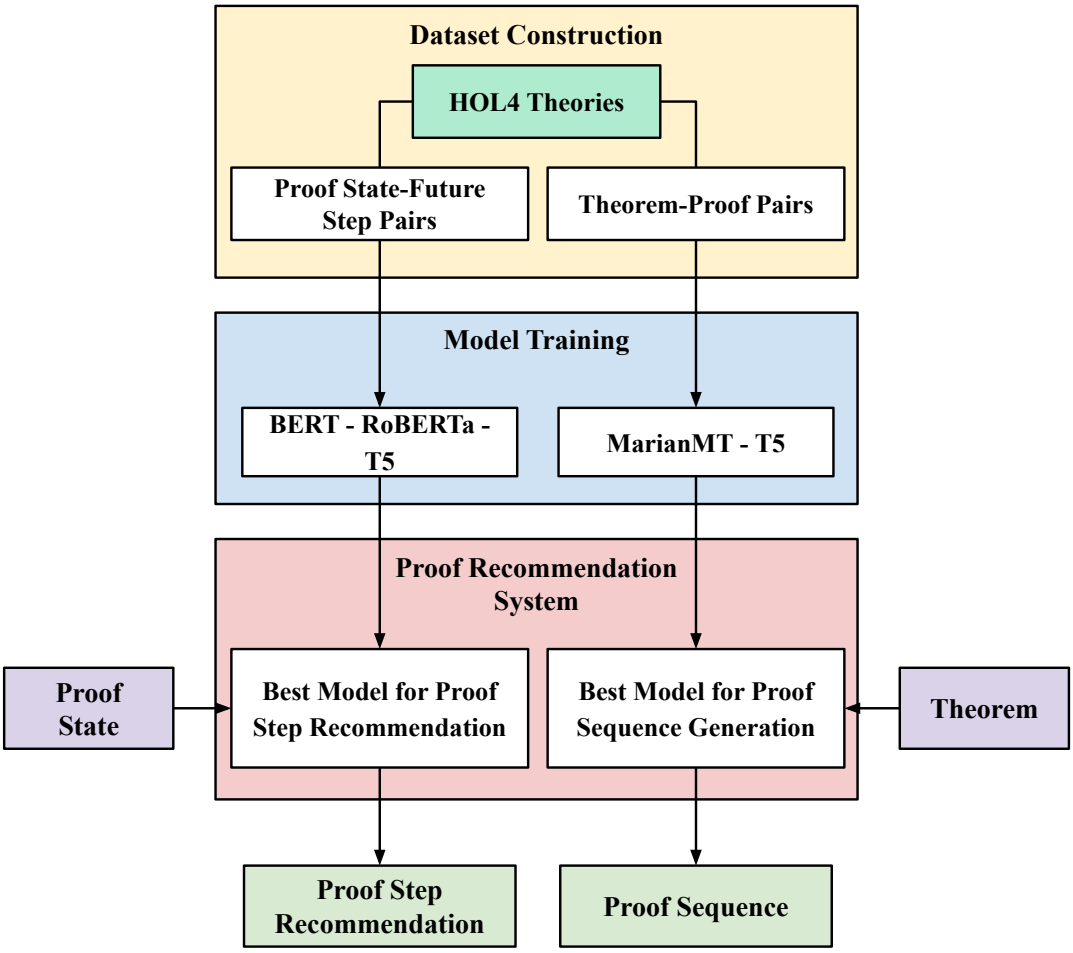

Figure 1.1: Proposed Methodology

Figure 1.1describes the details of the proposed methodology to realize the goals of this thesis. The proposed methodology begins with the construction of datasets derived from HOL4 theories which are comprehensive compilations of formalized mathematical proofs and theorems. These theories provide the raw material from which two datasets will be constructed. The first dataset consists of *Proof State-Future Step Pairs*, which represent the relationship between a proof state at a given point and the subsequent proof step needed to progress toward the proof goal. These pairs are crucial for training models that recommend the next proof steps, as they encapsulate the decision-making process involved in theorem proving. The second dataset comprises *Theorem-Proof Pairs*, which map each theorem to its corresponding proof sequence. This dataset provides the foundation for training models capable

of generating complete proofs. The creation of these datasets involves a meticulous extraction and preprocessing phase, ensuring that the data accurately reflects typical scenarios encountered during theorem proving, thus enhancing the models' training effectiveness.

Following the dataset preparation, appropriate LLMs were trained to address the two tasks. For the Proof Step Recommendation task, we fine-tune and use the BERT, RoBERTa, and T5 models. These models were selected for their ability to analyze textual and contextual data due to their underlying transformer architectures, which are specifically designed to capture deep contextual relationships within text, making them highly effective for tasks requiring nuanced text interpretation like proof step recommendation. The selected models offer capabilities specifically suited to interactive theorem proving, a fundamentally natural language problem that requires identifying textual patterns to grasp the progression of the proof. This ability is crucial to effectively guiding the theorem proving process. The training process involved experimenting with different model configurations, hyperparameters, and evaluation metrics to determine the most effective approach. The recommendation system aims to assist users by analyzing the current proof state and suggesting the next logical steps or tactics to progress the proof.

In contrast, the Proof Generation task employs the sequence-to-sequence models, MarianMT and T5 in order to generate complete proof sequences based on theorem statements. These models are particularly adept at modeling complex dependencies and maintaining coherence over long text sequences, essential for the structured nature of formal proofs. These models were configured to generate proof sequences with a maximum length of 512 tokens, aligning with their typical sequence length capacity. This task is particularly challenging due to the complexity of the proofs and the need to capture intricate logical patterns. The training process for proof generation involved rigorous hyperparameter tuning and extended training periods to refine the models' ability to produce proof sequences that are both valid and logically coherent, meeting the rigorous standards required for theorem proving. The proof generation functionality represents a higher level of automation, allowing users to input a theorem statement and receive a complete sequence of proof steps required to construct a valid proof.

After identifying the best-performing models for each task, these models were integrated into a Proof Recommendation System that offers two main functionalities. Firstly, the system provides Proof Step Recommendations by analyzing the current proof state and suggesting the next tactic to progress the proof. Secondly, the system enables Proof Generation by taking a theorem statement as input and producing a complete sequence of proof steps required to construct a valid proof. This integration not only streamlines the process but also enhances the accessibility and efficiency of theorem proving in the HOL4 environment.

We have developed a tool called HOL4PRS (HOL4 Proof Recommendation System) [61], implemented in Python [62] and deployed on Google Colab [63]. HOL4PSR is freely available on GitHub, making it accessible for users. This cloud-based deployment allows users to interact with the system without the need for local installations, facilitating ease of use and accessibility. The HOL4PRS tool assists users by analyzing the current proof state and suggesting the next HOL4 tactics to advance the proof. Furthermore, HOL4PRS facilitates Proof Generation by taking a theorem statement as input and producing a complete sequence of proof steps necessary to construct a valid proof. This integration not only streamlines the proof process but also enhances the accessibility and efficiency of theorem proving within the HOL4 environment. The evaluation of the proof step recommendation task revealed that the best-performing model, RoBERTa, achieved a top-3 accuracy of 77.3% and a top-7 accuracy of 89.88% on the combined dataset. In the proof generation task, the T5 and MarianMT models were assessed on a randomly selected theorem statement, resulting in the generation of a complete proof sequence with a Levenshtein Similarity Percentage (LSP) [64] of 78%.

1.5 Thesis Contributions

This thesis presents a practical approach to improving the accessibility and efficiency of the HOL4 theorem prover through the integration of Large Language Models. The main contributions of this work are as follows, where the publication references are available in the Biography section at the end of the thesis document:

- The thesis explores the use of LLMs, specifically BERT, RoBERTa, and T5, for predicting tactics in theorem proving. These models were fine-tuned on a

dataset of HOL4 theorems and their corresponding proofs, achieving reasonable accuracy in suggesting the next steps in proof construction. This work highlights the potential of LLMs to assist in understanding and predicting sequences in formal reasoning tasks [Bio-Cf1, Bio-Cf-2].

- In addition to tactic prediction, this thesis employs T5 and MarianMT for proof searching, framing theorem proving as a sequence-to-sequence task. These models generate sequences of proof tactics, allowing for the exploration of different proof paths. This approach aims to enhance the system's ability to provide effective proof strategies.

- We have implemented a tool called HOL4PRS for proof step recommendation and proof generation. This tool is designed to assist users in constructing proofs within the HOL4 environment and is made freely available online in a public GitHub repository [61]. HOL4PRS aims to support users by recommending suitable tactics, thereby reducing the cognitive load and improving the efficiency of proof construction [Bio-T1].

- By integrating LLMs into proof recommendation and sequence generation, this thesis contributes to the ongoing efforts to incorporate AI into formal verification. While the integration with HOL4 is still in progress, the insights and methodologies presented here provide a foundation for future exploration in AI-driven theorem proving. This work emphasizes the role of LLMs in supporting formal methods and highlights the potential for further development in various applications within the field of formal verification.

1.6 Thesis Organization

The rest of the thesis is organized as follows: In Chapter 2, we delve into the foundational concepts of theorem proving, providing a comprehensive overview of its principles and significance. We also introduce the HOL4 theorem prover, highlighting its capabilities and applications in formal verification. Additionally, we explore the role of large language models, such as BERT, RoBERTa, T5, and MarianMT, in enhancing theorem proving processes.

Chapter 3 focuses on proof step recommendation. We begin with an introduction to the problem statement, outlining the challenges faced in this area. Following this, we present our proposed methodology for addressing these challenges, detailing the dataset utilized for training and evaluation, and the process of model fine-tuning. We then discuss the experimental results obtained from our approach, concluding the chapter with a summary of the key findings.

In Chapter 4, we shift our attention to proof generation. This chapter starts with an introduction to the relevant challenges and methodologies associated with generating proof sequences. We provide a detailed description of the dataset used, the model fine-tuning process, and the experimental results that demonstrate the effectiveness of our proposed methods. The chapter concludes with a summary of the insights gained from our experiments.

Finally, Chapter 5 concludes the thesis by summarizing the primary findings and reflecting on the contributions made to the field of AI-assisted theorem proving. It also discusses potential directions for future research, highlighting opportunities for further exploration and development in this evolving area.

Chapter 2

Preliminaries

This chapter lays the groundwork by explaining the core concepts related to theorem proving, the HOL4 theorem prover, and Large Language Models (LLMs). It explores how LLMs such as T5, BERT, RoBERTa, and MarianMT can be applied to enhance the theorem proving process. These preliminaries provide essential background and context for the methodologies and experiments presented in subsequent chapters.

2.1 Theorem Proving

Theorem proving is the process of establishing the correctness of a statement by a sequence of logical deductions from a set of pre-defined axioms and inference rules. Theorem proving thereby provides a mathematical framework to ensure that systems satisfy their specifications under all possible conditions. In contrast to the empirical testing, which justifies system behavior under certain conditions, theorem proving provides a rigorous approach to verify properties of correctness, in particular for systems belonging to the most critical domains.

One can view theorem proving as a collection of conclusions derived from formalized representations of problems. To reason about specifications at a high level, it requires translating them into logical statements, along with the development of proofs that demonstrate their validity. Theorem proving is thus based on the rigorous mathematical reasoning and therefore capable of eliminating ambiguity which can never be achieved by traditional testing methods.

The development of theorem provers, specialized software tools aimed at helping users construct and verify proofs, has considerably advanced the field. Such environments are provided by tools like HOL4 [7], Coq [8], Isabelle [9], Lean [10], PVS [11], Mizar [12], and Metamath [13], where formalization of a problem, application of logical tactics, and proof steps are all verified at once. These theorem provers have been applied with a great success in the fields of hardware verification, software certification and cryptographic protocol analysis, where correctness is a major concern. These are very powerful tools, but effective use of them does require deep knowledge/understanding of the underlying mathematical theories and detailed acquaintance with the specific logic frameworks used by the underlying theorem prover.

The process of theorem proving involves several key components and steps that work together to establish the validity of a given statement or theorem. At the outset, a *proof goal* is defined, which represents the statement that needs to be proven. This goal is expressed in formal mathematical language, such as first-order logic (FOL) [65] or higher-order logic (HOL) [6], allowing for precise reasoning and manipulation. For instance, a theorem might assert that "if P and Q are true, then $P \wedge Q$ is true". The challenge lies in constructing a proof that rigorously demonstrates this assertion.

To achieve this goal, theorem provers utilize a set of axioms and definitions that serve as the foundational building blocks of the proof. *Axioms* are statements accepted as true without proof, while *definitions* provide the necessary context and meaning for the terms used in the theorem. The theorem prover operates within a specific *theory*, which is a collection of axioms, definitions, and previously proven theorems that are relevant to the proof at hand. For example, we might define $P \wedge Q$ (the logical conjunction of P and Q) as being true if both P and Q are true. Theorems are statements that can be proven based on axioms, definitions, and previously established theorems. In our example, the statement "if P and Q are true, then $P \wedge Q$ is true" can be considered a theorem that we want to prove.

The proving process typically involves the application of *tactics*, which are strategies or rules that guide the proof construction. In the context of HOL4, for example, tactics might include simplification, assumption introduction, or contradiction. Each application of a tactic results in a proof step, which transforms the current state of the proof into a new state. The sequence of proof steps taken to reach the proof goal constitutes the proof itself.

The proof construction is inherently iterative and often requires a sequence of tactics to navigate through the complexities of the proof. For example, consider a simple proof goal: proving that if P and Q are true, then $P \wedge Q$ is true. The proof begins with the goal $P \wedge Q$. The first step involves applying the tactic of assumption to introduce P and Q as true premises. This leads to the current state where we assume P is true and Q is true. Next, we use the tactic of conjunction introduction, which allows us to combine the two assumptions. From P and Q, we can conclude $P \wedge Q$. At this point, the proof goal $P \wedge Q$ is successfully proven.

2.2 HOL4 Theorem Prover

HOL4, an abbreviation for Higher-Order Logic version 4, belongs to a lineage of theorem provers that traces its origins to the LCF (Logic for Computable Functions) theorem prover, which was developed in the late 1970s at Stanford and Edinburgh. The LCF framework [66] pioneered the use of a meta-language, ML, enabling the interactive construction of proofs while ensuring soundness by embedding logical inference rules directly into the programming environment. HOL4 enhances this foundational design, offering a powerful platform for theorem proving in the context of higher-order logic.

As part of the HOL family, HOL4 has evolved from its predecessors, beginning with HOL88 in 1980s, followed by HOL90 in the 1990s, and has unddegone numerous improvements aimed at to enhance its functionality, user interface, and overall applicability. Today, it is recognized as one of the leading interactive theorem provers in both academic and industrial settings, especially in the area of formal verification.

The system is based on classical higher-order logic, a complex formalism that extends first-order logic by allowing functions and predicates to be treated as first-class entities. This expressive framework facilitates reasoning about abstract concepts, such as sets of functions or relationships between predicates, making HOL4 particularly adept at specifying and verifying complex systems. The underlying logic is contained within a small kernel, which guarantees that all proofs produced by the system are sound. This architecture ensures that any theorem proven in HOL4 adheres to its axioms and inference rules, providing a high level of confidence in the correctness of its results.

2.3 Large Language Models

In recent years, Large Language Models (LLMs) have emerged as powerful tools in Natural Language Processing (NLP) [67] and beyond, demonstrating unprecedented capabilities in understanding, generating and transforming the text. These models, based on transformer architectures, leverage self-attention mechanisms to model complex relationships within input sequences, enabling them to perform a wide variety of tasks with remarkable accuracy and efficiency.

LLMs are pre-trained on vast corpora of text data, capturing patterns, semantics and structures in the language. This pretraining equips them with a deep understanding of linguistic and contextual nuances, which can be fine-tuned for specific applications across diverse domains. Their flexibility and generalization capabilities have made them indispensable not only in traditional NLP tasks, such as translation, summarization, and question answering, but also in more specialized applications requiring structured reasoning and formal verification.

The selection of T5, BERT, RoBERTa, and MarianMT for this thesis stems from their proven ability to handle tasks requiring deep understanding, structured reasoning, and adaptability. These models represent a diverse range of architectures and training paradigms, making them particularly suitable for addressing the multifaceted challenges of interactive theorem proving. Their foundation in transformer-based architectures ensures that they can efficiently model contextual dependencies and sequence relationships, which are critical in theorem proving tasks. Furthermore, their extensive pretraining on large corpora equips them with a robust understanding of linguistic patterns and structures, allowing them to generalize effectively to new domains, such as formal reasoning. The flexibility of these models, combined with their scalability and performance across various NLP tasks, makes them ideal candidates for exploring and automating complex tasks in formal verification, particularly when framed as sequence-to-sequence problems. These general attributes underscore their transformative potential in advancing AI-driven solutions for interactive theorem proving.

In the following, we review the prominent LLMs relevant to the research contribution presented in this thesis, with a particular focus on their architecture, training objectives, and applicability to interactive theorem proving. By framing the interactive theorem proving tasks as sequence-to-sequence problems, the selected models

bring unique strengths to automating proof construction and addressing the inherent challenges in formal verification.

2.4 T5

Text-to-Text Transfer Transformer (T5) [68] is a state-of-the-art language model, introduced by Google Research, that aims to combine all NLP tasks into one simple framework. Unlike existing models optimized for specific tasks, T5 reformulates any NLP problem-such as translation, summarization, and classification-into a text-to-text task. This innovative methodology has resulted in the simplification of task representation and allows the same model architecture and training objectives across a wide range of applications.

At its core, T5 is a transformer architecture, a very powerful architecture of neural networks that relies on self-attention mechanisms for modeling contextual relationships in the input. T5 uses an encoder-decoder structure, where an encoder processes input text and a decoder generates the corresponding output text. This architecture particularly suits T5 for tasks related to conditional text generation, which may include generating sequences based on input or producing translations of given content.

The diversity in T5 stems from its training objective-that of a denoising autoencoder. It masks large portions of the input text while training and asks the model to reconstruct the original text. This objective implies that T5 would learn robust representation in language patterns and generalize effectively on unseen tasks. Further, T5 is pre-trained on a huge corpus of text that allows it to learn deep structures and semantic subtlety. Fine-tuning then on task-specific datasets further improves its performance, making it adaptable to a wide variety of domains.

This capability of handling text-to-text transformations makes T5 highly suitable, in the context of this thesis, to tackle challenges in the interactive theorem proving. By framing the interactive theorem proving tasks, such as the prediction of proof steps or the generation of proof sequences-into text generation problems, T5 exploits its advanced language modeling capability to predict sequences of tactics or generate proofs from a given theorem statement. This application represents the more general potential of T5 for domains beyond the traditional NLP by demonstrating its adaptability to tasks requiring formal reasoning and verification.

2.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) [69] is one revolutionary NLP model by Google Research. It pioneered a new approach toward the understanding of language through a bidirectional transformer architecture to contextualize the words based on both the preceding and succeeding elements in a sentence. This bidirectional mechanism contrasts sharply with earlier unidirectional models, which could analyze text only sequentially, either left to right or vice versa. By using information from all directions, BERT encapsulates a far more subtle approach toward the semantics and syntax of languages.

BERT fundamentally embeds the transformer architecture and relies on self-attention mechanisms to relate between words in a sequence. BERT, pre-trained on large-scale datasets of two levels of objectives-masked language modeling and next sentence prediction-each use Masked Language Modeling (MLM) to predict the masked words in a sentence, which results in the model learning the context of words both ways. Next Sentence Prediction (NSP) trains the model to understand relationships between sentence pairs, further enhancing its ability in processing complex textual relationships.

In this thesis, BERT plays the role of the cornerstone for AI-powered theorem proving. Its learning potential from large data and the ability to generalize across complex patterns make it a suitable candidate to understand logical structures and sequences inherent in formal proofs. This work, on applying BERT to theorem proving, explores how state-of-the-art NLP techniques go beyond traditional applications to contribute toward advances in formal verification and AI-assisted reasoning.

2.6 RoBERTa

Robustly Optimized BERT Pretraining Approach (RoBERTa) [70] is a transformer-based language model developed to enhance the performance of the BERT model. RoBERTa improves and optimizes the pretraining methodology introduced by BERT, refining some of the major limitations and thereby yielding one of the most robust and high-performing models for a variety of NLP tasks. For instance, RoBERTa extends BERT by making several changes to the pretraining process, including training on larger datasets, increasing batch size, removing the next-sentence prediction

objective, and using dynamic masking at pretraining. This enables RoBERTa to grasp richer contextual knowledge in texts and significantly improves the accuracy and generalizability of performance for downstream tasks.

The main advantage of RoBERTa is its capability to learn complex patterns in sequential data. It is thus suited for applications requiring relationship modeling between tokens, including language understanding, classification, and generation. The ability of RoBERTa to process sequences and capture subtle patterns has made it popular in areas other than mainstream NLP, including theorem proving and formal verification. The inherent structure within such data allows the model to observe patterns in logical and structured data, thus enabling it to contribute effectively to tasks like proof step prediction and sequence generation.

2.7 MarianMT

Marian Machine Translation (MarianMT) [71] is an open-source framework for the neural machine translation that efficiently translates source text into multiple target languages. It is highly scalable and adaptable, implemented using the transformer architecture, which is one of the leading models in natural language processing, including a self-attention mechanism and long-range dependencies. MarianMT adopts an encoder-decoder architecture, where the input sequence is encoded into a high-dimensional representation by the encoder, which in turn allows the decoder to produce a corresponding output sequence. It especially performs effectively for sequence-to-sequence tasks.

The strength of MarianMT lies in capturing complex dependencies and returning coherent output for structured input. This makes it particularly relevant for applications where there is a need to transform data in a structured way, for instance, in the generation of a sequence of tactics that would serve to prove a theorem. Fine-tuning MarianMT on the datasets of theorems and corresponding proof scripts adapts this model to predict sequences of tactics that lead to valid proofs. This adaptation demonstrates the flexibility of the model and how it can be used to improve efficiency and accessibility drastically in theorem proving. The use of MarianMT here bridges the gap between modern machine learning methods and the domain of formal verification, hence scaling these solutions to meet the challenges in proof construction.

Chapter 3

Proof Step Recommendation

This chapter describes our proposed a methodology for predicting the next logical step in the proof process using LLMs. We discuss the dataset preparation, the fine-tuning of transformer-based models, and the evaluation process. The results demonstrate the effectiveness of the proposed system in reducing the cognitive load on theorem provers while improving accuracy and efficiency.

3.1 Proposed Methodology

By leveraging the sequential and contextual nature of proofs, we propose and approach that transforms proof data into a structured format suitable for ML. In particular, we use LLMs, fine-tuned on the data, in order to predict the next logical tactic based on the history of the previously applied tactics, hence enabling more efficient and informed decision making during the development of proof process.

The proposed methodology involves preparing a dataset of proof sequences, converting them into pairs of proof states and the corresponding next tactics. These pairs serve as input for transformer-based LLMs, which learn to identify patterns and dependencies in the sequences. The models are fine-tuned to generate contextually relevant recommendations that align with the logical flow of the proof construction.

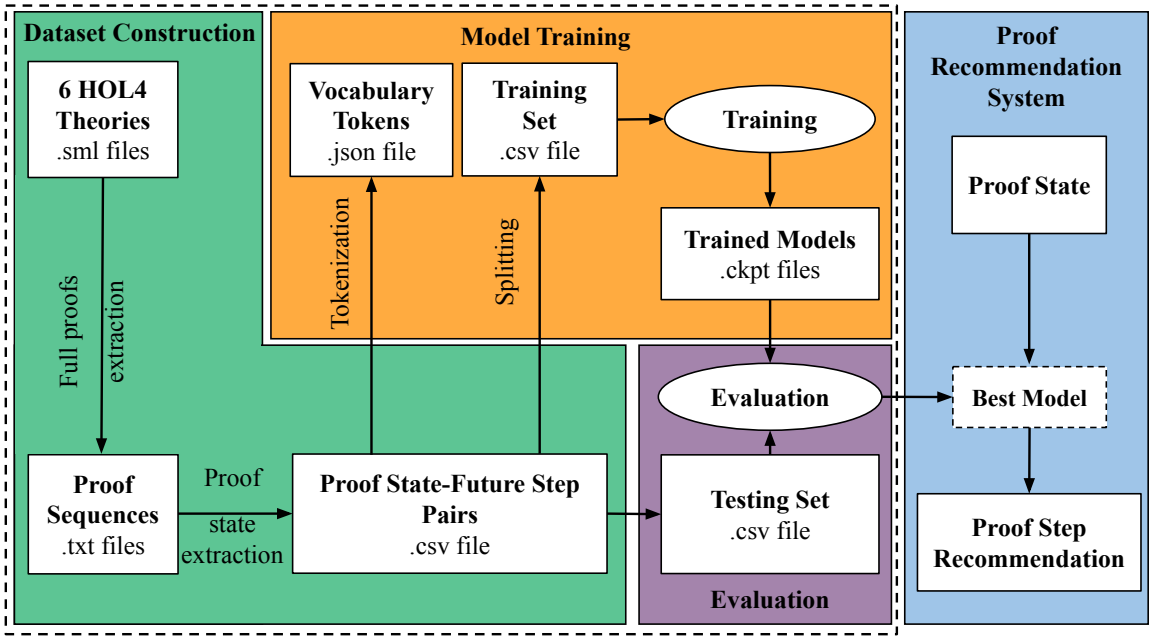

Figure 3.1: Proposed Methodology for the Proof Step Prediction

Figure 3.1 illustrate the proposed methodology for the proof step recommendation that is designed to assist users of the HOL4 theorem prover by recommending proof steps based on the current proof state. The system begins by accepting a proof state as input, represented as a tactics sequence that have been applied so far in constructing the proof. To ensure sufficient context, each input proof state must include a minimum of three tactics. This sequence undergoes advanced analysis, where patterns and strategies are identified to predict the most suitable next proof step. The system's recommendations derive from an extensive pretraining on a diverse dataset of HOL4 proofs, enabling it to generate contextually relevant and optimized tactics for each input state.

The proposed methodology is generic and adaptable, capable of being extended to other interactive theorem provers such as HOL Light, Coq and PVS. Each block of this workflow, from dataset construction to model fine-tuning and evaluation, is designed to integrate seamlessly, creating a robust system for improving interactive theorem proving. The blocks surrounded with a discontinued line indicate processes that are performed offline without requiring continuous online interaction for each operation.

The dataset construction is a foundational step in this methodology, involving the selection of six HOL4 libraries, as shown in the left upper half of Figure 3.1. Proof scripts from these libraries, written in HOL4's standard `.sml` format, are parsed to extract the complete proof sequences for theorems and lemmas. Each proof sequence represents the step-by-step application of tactics necessary to complete the proof. To create training data suitable for the models, these sequences are transformed into pairs of proof states and their subsequent tactics. A proof state, in this context, refers to an intermediate point in the proof where a certain number of tactics has already been applied. For each sequence of n tactics, $n - 4$ training instances are created by considering all possible intermediate proof states with a minimum history of three tactics. This choice is motivated by the observation that the initial tactics applied at earlier stages are often similar, leading to an imbalance in the dataset. By eliminating states shorter than three, we reduce redundancy and ensure a more diverse and balanced dataset. This approach allows the dataset to capture all stages of proof development while accommodating variability in tactic sequences and supporting the multi-label classification. Preprocessing plays a critical role in adapting the dataset for the model training. Tasks, such as tokenizing the proof sequences, assigning a numerical vocabulary, and splitting the data into training and testing sets, ensure compatibility with the selected models. These models are fine-tuned on the prepared dataset to optimize their performance for the specific task of proof step recommendation. The fine-tuning process involves comprehensive hyperparameter tuning, utilizing grid search techniques to systematically explore combinations of parameters such as batch size, learning rate, number of layers, and hidden unit sizes. This methodical approach helps in identifying the optimal settings that minimize training loss and maximize prediction accuracy. During this phase, each model configuration is rigorously evaluated across a range of hyperparameters to ensure robustness and effectiveness. For each dataset, multiple trained instances of each model are evaluated, and the best-performing instance is selected based on accuracy metrics. This selection process is guided by the grid search outcomes, which highlight the hyperparameter settings that contribute most significantly to model performance.

Model evaluation is performed using the n-correctness rate, which quantifies the percentage of cases where the correct proof steps are included among the top-n predictions generated by the model. This metric reflects the model's ability to provide

accurate and relevant suggestions within a defined range of predictions. Multiple trained instances of each model are assessed, and the instance with the highest n-correctness rate is selected as the best-performing model.

The core functionality of this methodology lies in generating a ranked list of recommended tactics tailored to the input proof state. These recommendations aim to assist users by providing contextually relevant suggestions that enhance the efficiency and efficacy of the theorem proving process. The system acknowledges the inherent complexity of theorem proving, where multiple tactics may be valid for a single proof state, and leverages its training to prioritize the most likely successful options. The workflow concludes with model evaluation using metrics such as top-n accuracy, which measures the likelihood of including the correct tactic in the top recommendations.

3.2 Dataset

This section presents an overview of the datasets employed in this thesis, which are crucial for the development and assessment of the proof step recommendation system. Sourced from various HOL4 theorem proving projects, these datasets encompass a wide array of applications in formal verification, each offering distinct challenges and scenarios that enhance the experimental framework. They include formal proofs related to dynamic dependability analysis, probabilistic behavior in wireless sensor networks, risk assessment, information flow security, and statistical modeling of normal random variables. The diversity and complexity of these datasets not only facilitate the training of AI models but also ensure that the models can generalize effectively to provide relevant recommendations in real-world theorem proving tasks. Following this overview, we will detail the systematic construction process of these datasets, highlighting the methods used to extract and organize proof scripts for optimal training and evaluation.

3.2.1 Datasets Description

This thesis utilizes six datasets sourced from HOL4 theorem proving projects, which encompass a range of applications in formal verification. These datasets form the backbone of our experimental framework, providing diverse scenarios and challenges to evaluate the proposed proof recommendation system. Below is a detailed

description of each dataset:

Dataset 1: Formal Dynamic Dependability Analysis

This dataset [72] focuses on formal methods applied to dynamic dependability analysis, specifically leveraging HOL4 theorem proving. It includes proofs and formalizations aimed at verifying the dependability and correctness of dynamic systems under varying operational scenarios. Such analysis is crucial for ensuring system stability and reliability in environments with dynamic behaviors, such as aerospace systems and safety-critical applications.

Dataset 2: Formal Probabilistic Analysis of Wireless Sensor Networks

This dataset [73] provides formal proofs for analyzing the probabilistic behavior of wireless sensor networks (WSNs). It involves the study of reliability, latency, and efficiency in sensor network operations, particularly under uncertain or dynamic conditions. WSNs are extensively used in applications like environmental monitoring, healthcare, and industrial automation, making their formal verification critical.

Dataset 3: Formal Probabilistic Risk Assessment

This dataset [74] is dedicated to probabilistic risk assessment, where theorem proving is used to model and evaluate risks in systems. It contains proofs that help quantify uncertainties and analyze potential failures within complex systems. The dataset's focus on rigorous probabilistic modeling makes it invaluable for applications like financial risk analysis, engineering system safety, and project planning.

Dataset 4: Formal Analysis of Information Flow Using Min-Entropy and Belief Min-Entropy

This dataset [75] addresses the analysis of information flow in systems, utilizing Min-Entropy and Belief Min-Entropy as key metrics. These formalizations are essential for evaluating and ensuring secure information transfer within systems, such as cryptographic protocols and secure communication channels. The dataset includes formal proofs that assess information leakage, adversarial resistance, and data confidentiality.

Dataset 5: Formalization of Normal Random Variables

This dataset [76] encompasses the formalization of normal random variables, an integral component in statistical analysis and probabilistic modeling. It provides a foundation for proofs involving Gaussian distributions and related statistical concepts, enabling formal verification in domains such as data science, econometrics, and engineering risk assessment.

Dataset 6: Proof Searching in HOL4 with Genetic Algorithm

This dataset [77] contains proofs generated using a genetic algorithm approach integrated into the HOL4 theorem prover. It exemplifies an innovative technique for automated proof searching by optimizing sequences of tactics based on heuristic methods. This dataset is particularly useful for exploring the efficiency of AI-based methods in reducing the search space of proofs.

These projects often involve intricate logic and large search spaces, making manual proof development time-consuming and challenging, especially for non-experts. Automation ensures consistency, scales across large datasets, and makes HOL4 more accessible by simplifying proof processes. The datasets from these projects, featuring diverse scenarios, structured proof sequences, and real-world applications, are ideal for training AI models. They help models generalize better, understand complex dependencies, and recommend diverse, contextually relevant tactics, ultimately enabling faster and more accurate theorem proving in practical, high-assurance domains.

3.2.2 Dataset Construction

The dataset used in this work is systematically constructed to enable proof step recommendation for the HOL4 theorem prover, aiming to predict the next proof step (tactic) based on a given proof state. The construction process starts by collecting proof scripts from the six HOL4 libraries. To accurately extract the right content for our dataset, a thorough understanding of HOL4 syntax was required in order to write the script for data extraction. Subsequently, we develop a custom script to systematically parse the proof scripts and extract all available theorems, lemmas, and their corresponding proof steps (tactics). Each proof is represented as a sequence

of tactics applied in order to complete the verification process. The resulting data contains a detailed record of all proof sequences across the targeted libraries, offering a comprehensive view of the steps involved in theorem proving within HOL4.

In order to facilitate the recommendation of proof steps, the dataset is transformed into pairs of current proof states and corresponding next tactics. A proof state in this context represents a sequence of tactics applied up to a certain point in the proof process. For each proof sequence of n tactics, we generate $n-4$ instances, considering sequences with a minimum history of three tactics to ensure meaningful context for prediction. For example, a sequence $[\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n]$ is transformed into instances, such as $(\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}, \mathcal{T}_4)$ and $(\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4\}, \mathcal{T}_5)$, and so forth. This restructuring allows the dataset to capture all possible proof states of varying lengths and associate each state with its subsequent tactic.

To reflect the inherent complexity of theorem proving, we include instances with similar tactic histories but differing future tactics. This approach acknowledges multiple tactics can be valid next steps, depending on the reasoning path. For example, consider a proof state with the tactic history $[\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]$. In some proofs, the next step might be $\mathcal{T}_4$, while in others, $\mathcal{T}_5$ could also lead to a successful proof. Both $(\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}, \mathcal{T}_4)$ and $(\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}, \mathcal{T}_5)$ are included as separate instances in the dataset. This ensures that the dataset captures such variations, which are inherent to the flexible and exploratory nature of theorem proving. By incorporating these diverse proof states, the dataset is enriched with examples reflecting the multiple valid paths that theorem proving might take. This design made the dataset robust for training LLMs capable of handling the complexities of interactive theorem proving, while aligning with the multi-label classification approach central to our methodology.

To illustrate the application of our methodology, we consider a specific example extracted from one of our SML files. Below is a segment from an HOL4 proof script, showing a series of tactics applied to prove a theorem:

```
Theorem addition_example:
  proves ‘forall n m. (n + m) + 1 = n + (m + 1)‘
  [
    REWRITE_TAC[ADD_ASSOC],
    GEN_TAC,
    GEN_TAC,
```

```
  ARITH_TAC,
  REWRITE_TAC[ADD_COMM]
];
```

From this proof script, we extract the sequence of tactics applied in the theorem proof as follows: [REWRITE_TAC, GEN_TAC, GEN_TAC, ARITH_TAC, REWRITE_TAC].

The sequence is then transformed into pairs of current proof states and the subsequent tactics, focusing solely on the types of tactics without considering their specific arguments. This transformation is illustrated below:

1. Initial State: [REWRITE_TAC, GEN_TAC, GEN_TAC]
   Next Step: ARITH_TAC

2. State: [REWRITE_TAC, GEN_TAC, GEN_TAC, ARITH_TAC]
   Next Step: REWRITE_TAC

This approach to dataset transformation captures each proof state and its corresponding next tactic, focusing purely on the type of tactic applied. By omitting the arguments, we ensure the models learn to predict the next tactic based on the sequence and type of previous tactics, independent of the specific details of their application.

The final dataset comprises six individual datasets, each corresponding to a specific HOL4 library, alongside a combined dataset (Dataset 7) that comprises all 6 datasets to assess the model's generalization ability. These datasets vary in the number of distinct tactics, proofs, and proof states, as summarized in Table 3.1. Dataset 7, with 116,156 proof states drawn from 5,136 proofs, provides the most comprehensive coverage, combining the characteristics of the individual datasets into a unified corpus. Pre-processing plays an essential role in preparing the dataset for model training. The initial step involves tokenizing the sequences, where each proof and its corresponding steps are broken down into tokens. This tokenization transforms raw text into a format that is analyzable by transformer-based models. Following tokenization, the dataset undergoes adaptation to meet the input requirements of these models. This includes aligning the tokenized data with the expected input structure, such as padding or truncating sequences to a fixed length and converting tokens into numerical indices using a predefined vocabulary. Lastly, the data is split into training

and testing subsets, with 90% of the data designated for training to maximize the learning potential, while the remaining 10% is reserved for testing to evaluate the models' performance.

Table 3.1: Summary of the Datasets

	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	**Dataset 7**
Distinct Tactics	115	132	26	44	32	89	162
Proofs	1,873	2,475	153	295	61	279	5,136
Proof States	43,167	57,602	2,973	7,371	1,784	3,259	116,156

3.3 Experimental Evaluation

In this section, we describe our efforts for the experimental evaluation of the methods developed in this chapter. In particular, we detail the methodology employed for fine-tuning transformer-based models to enhance the recommendation of proof steps within the HOL4 theorem prover. By framing the task as a multi-class classification problem, we established a framework that connects current proof states with their subsequent tactics, allowing the models to effectively learn from the sequential nature of proof construction.

In the process of evaluating various models for proof step recommendation, preliminary experiments were conducted with models like DistilBERT [78], XLNet [79], Electra [80], BERT [69], RoBERTa [70] and T5[68]. These models were initially tested for their potential to adapt to the unique requirements of theorem proving. However, the first three models performed poorly on the datasets created from the HOL4 libraries, demonstrating significant challenges in capturing the logical complexity and depth required for proof step prediction. This observation led to the refinement of our model selection process, ultimately favoring BERT, RoBERTa, and T5 due to their superior performance in handling complex pattern recognition and logical reasoning inherent in theorem proving.

Utilizing the advanced transformer models BERT, RoBERTa, and T5, we implemented a systematic approach to fine-tuning, which included careful pre-processing of the dataset and optimization of hyper-parameters. This rigorous training process was supported by a robust evaluation framework, ensuring that the models were ready to

predict the next logical steps in theorem proving. The subsequent evaluation phase focused on measuring the models' performance through metrics that account for the inherent complexity and variability of proof strategies, ultimately aiming to enhance the decision-making capabilities of users engaged in interactive theorem proving. For the experiments, we used our HOL4PRS tool, which input consists of at least three previously applied tactics that serve as the context for the current proof state. This input is essential for the model to generate relevant recommendations.

3.3.1 Model Fine-Tuning

To address the task of recommending proof steps in the HOL4 theorem prover, we frame it as a multi-class classification problem. Each proof state, represented as a sequence of previously applied tactics, is associated with a single next tactic from the dataset. This framing allows the models to capture the relationship between the current proof state and the tactics that logically follow, leveraging the sequential and contextual dependencies in proof construction.

Using the PyTorch Lightning library [81], a tool that simplifies machine learning training by managing code and automating tasks, we fine-tune the models on our task-specific dataset. Pre-processing steps include tokenizing proof sequences and pairing proof states with their corresponding next tactics, to ensure that the models learn the patterns underlying HOL4 proofs.

During fine-tuning, we systematically adjust hyper-parameters such as batch size, learning rate, and weight decay to optimize performance. We conduct training over 10 epochs, with early stopping based on validation performance to mitigate overfitting. We partition the dataset into 90% for training and 10% for testing, providing a robust evaluation framework. All experiments are executed on GPUs provided by the Digital Research Alliance of Canada [82], which facilitates efficient training and scaling of the models.

3.3.2 Evaluation Metrics

Given that a proof state can lead to multiple valid next steps, it is crucial to use an evaluation metric that accommodates this flexibility. We employ the *n-correctness rate* as our primary metric, which measures the percentage of instances where the

correct proof step appears among the top-n predictions generated by the model. For example, if the model provides a list of the top 7 recommendations, the n-correctness rate indicates whether the correct next step is included in those suggestions. This metric is particularly relevant in ITP, as it allows users to consider several potential tactics, thereby enhancing their decision-making process.

The evaluation process begins with preparing the dataset from proof sequences, ensuring that each proof state includes a history of at least three previously applied tactics. This context is essential for informed predictions. After preparing the dataset, selected models are fine-tuned, optimizing hyperparameters such as batch size and learning rate.

Once training is complete, models are assessed using the *n-correctness rate*, providing a comprehensive evaluation of their ability to generate accurate recommendations. The results are analyzed to compare model performance, identifying the best-performing instance based on the highest *n-correctness rate*. This evaluation framework effectively measures the performance of the proof step recommendation system, ensuring it meets the needs of users engaged in theorem proving and enhancing accessibility for individuals with varying levels of expertise.

3.3.3 Experimental Results

An analysis of the results highlights that dataset characteristics significantly influence model performance. In fact, datasets with repetitive proof patterns and fewer distinct tactics, such as Dataset 3, achieve higher accuracy, while datasets with a broader range of tactics, like Dataset 6, pose greater challenges. To improve generalization, we combine all six datasets into a single, comprehensive dataset. This merged dataset exposes the models to a diverse range of proof styles, enhancing adaptability.

The evaluation results are summarized in Table 3.2. Among the tested models, RoBERTa consistently demonstrates superior performance, achieving n-correctness rates of 77.3%, 89.88%, and 93.7% for top-3, top-7, and top-10 recommendations, respectively, as shown in Table 3.2. These results mark a substantial improvement over related works, which report accuracies ranging from 50%-70% for top-3 to top-5 recommendations and 87% for top-3 predictions in other settings. During the extensive testing and validation phases, no obvious overfitting was observed. The models

maintained consistent performance across both the training and testing datasets, indicating a robust generalization to unseen data. RoBERTa's robust performance underscores its ability to capture intricate proof patterns, making it particularly effective for this task. Furthermore, its adaptability to the merged dataset highlights its potential for broader applications in theorem proving. For further analysis of model performance, only the top-7 recommendations will be considered as they offer a balance between accuracy and practicality.

Table 3.2: Performance Evaluation of Tactic Recommendation Models

	T5			**BERT**			**RoBERTa**		
Datasets	**Top-3**	**Top-7**	**Top-10**	**Top-3**	**Top-7**	**Top-10**	**Top-3**	**Top-7**	**Top-10**
Dataset 1	51.3%	68.7%	76.4%	52.7%	71.9%	79.9%	**54.5%**	**73.6%**	**93.7%**
Dataset 2	60.4%	75.5%	80.5%	60.5%	78.9%	86.3%	**59.7%**	**79.5%**	**85.8%**
Dataset 3	69.8%	93.4%	95.4%	76.1%	93.9%	97%	**78.4%**	**94.4%**	**97.5%**
Dataset 4	77.3%	95.3%	97.2%	87.3%	97.0%	98.5%	**89.5%**	**97.8%**	**98.8%**
Dataset 5	76.6%	97.6%	98.2%	76.6%	97.6%	98.2%	**76.6%**	**97.6%**	**97.6%**
Dataset 6	39.9%	55.2%	61.9%	45.1%	65.4%	72.7%	**43.4%**	**64.3%**	**73.8%**
Dataset 7	**72.9%**	**85.6%**	**87.8%**	**75.4%**	**88.7%**	**92.3%**	**77.3%**	**89.8%**	**93.7%**

Figure 3.2 illustrates the performance comparison of the models T5, BERT, and RoBERTa for the Top-7 correctness rate across all datasets. As shown in the figure, RoBERTa consistently outperforms the other models, achieving the highest Top-7 correctness rate of 89.8%. This aligns with the evaluation results presented in Table 3.2, underscoring RoBERTa's robustness and superior ability to capture intricate proof patterns. Its performance not only marks a significant improvement over related works but also demonstrates its adaptability to diverse datasets, as described in next section.

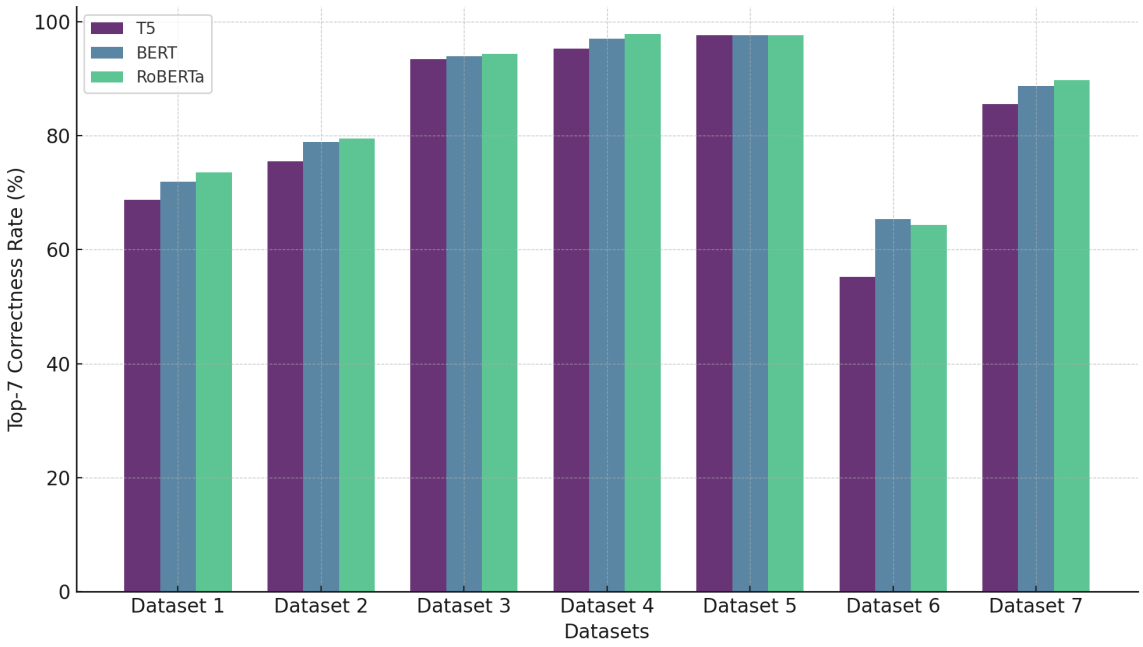

Figure 3.2: Top-7 Correctness rate for the Three Models

3.3.4 Comparison with Related Work

The proposed approach is focused on fine-tuning of LLMs such as BERT, RoBERTa, and T5, which excel at processing sequential and contextual data. This differs from earlier studies that predominantly employed k-NN, RNNs, or LSTMs. While these earlier models have their advantages, they often lack the nuanced understanding and flexibility provided by transformer-based architectures. The multi-label classification framework implemented in this chapter further enhances the system's ability to manage the complexity and variability inherent in interactive proofs.

This thesis specifically targets the HOL4 theorem prover, a tool recognized for its complexity and rigorous requirements for formal verification. By focusing on HOL4, the fine-tuned models are customized to its distinct characteristics. The dataset utilized in this thesis represents a substantial improvement in both scale and diversity. By extracting over 116,000 proof states from six different HOL4 libraries, the dataset captures a wide array of real-world projects. This is a significant enhancement over the typically smaller, more narrowly focused datasets found in related work. Furthermore, the inclusion of various libraries within HOL4 enables the models to generalize across different proof styles and contexts, thereby increasing their robustness.

In terms of performance, this thesis demonstrates remarkable progress compared to existing methods. The best performing model, RoBERTa, achieves a top-3 accuracy of 77.3% and a top-7 accuracy of 89.88% on the combined dataset. These results exceed those of many earlier studies, which generally report accuracies between 50% and 70% for similar tasks. RoBERTa's consistent performance in providing accurate recommendations across diverse datasets highlights the effectiveness of employing LLMs for proof step prediction.

In summary, the proposed approach addresses several limitations identified in previous research, such as dependence on less adaptable models, restricted datasets, and a narrow focus on specific theorem provers. By integrating advanced LLMs and concentrating on HOL4, this thesis presents a scalable, high-performing solution that significantly enhances the usability and efficiency of interactive theorem proving. These findings pave the way for future research to further explore the potential of LLMs in formal verification and proof assistance tasks.

3.4 Summary

This chapter presented the methodology and experimental evaluation of predicting proof steps in the interactive theorem prover HOL4. The approach involved constructing a comprehensive dataset from six HOL4 libraries, where proof sequences were transformed into pairs of intermediate proof states and subsequent tactics. To ensure meaningful context for prediction, proof states included a history of at least three tactics. Preprocessing steps, such as tokenization and data splitting, prepared the dataset for training models BERT, RoBERTa, and T5.

The proof step recommendation task was formulated as a multi-class classification problem, utilizing contextual dependencies in proof construction. Fine-tuning these models involved hyperparameter optimization and evaluation using metrics such as top-n correctness rates. The experimental results highlighted that RoBERTa consistently achieved superior performance, particularly on the merged dataset, which integrated proof sequences from all six libraries. This comprehensive dataset enabled models to adapt to diverse proof styles and improve generalization.

Building on the successful model training outlined in this chapter, we developed a tool, HOL4PRS, that is designed to act as a copilot independently of the HOL4 environment. This tool helps users by providing up to seven potential HOL4 tactics based on an input proof state of at least three tactics, thereby enhancing the effectiveness of theorem proving in various contexts.

The findings demonstrated the capability of the proposed methodology to deliver accurate and context-aware recommendations, providing a significant step toward optimizing interactive theorem proving workflows. This adaptable framework lays the groundwork for future applications in other theorem proving systems.

Having established the framework for proof step recommendation in this chapter, we will proceed in Chapter 4 to explore the extension of this methodology to the automated generation of complete proof sequences, demonstrating the broader applicability and scalability of our approach.

Chapter 4

Proof Sequence Generation

In the previous chapter, we have presented our methodology for the proof step recommendation task, in this chapter we focus on the task of generating entire proof sequences for given theorem statements. We frame the task as a sequence-to-sequence problem and explains the use of LLMs, namely T5 and MarianMT models, for this purpose. The chapter discusses the dataset preparation, model training, and evaluation, as well as experimental results highlighting the ability of the proposed approach to automate proof generation.

4.1 Proposed Methodology

Similar to the proof step recommendation approach, the methodology for generating a complete proof tactic sequences from theorem statements involves four main stages: dataset construction, model training, evaluation, and proof generation, as shown in Figure 4.1. The process begins with extracting data from HOL4 projects, which store proof scripts in `.sml` files. These files are parsed to retrieve theorem and lemma statements along with their corresponding proof sequences. Each proof sequence represents an ordered list of tactics applied to construct the proof. The extracted theorem-proof pairs are saved in a structured `.csv` file, providing the foundation for the subsequent steps.

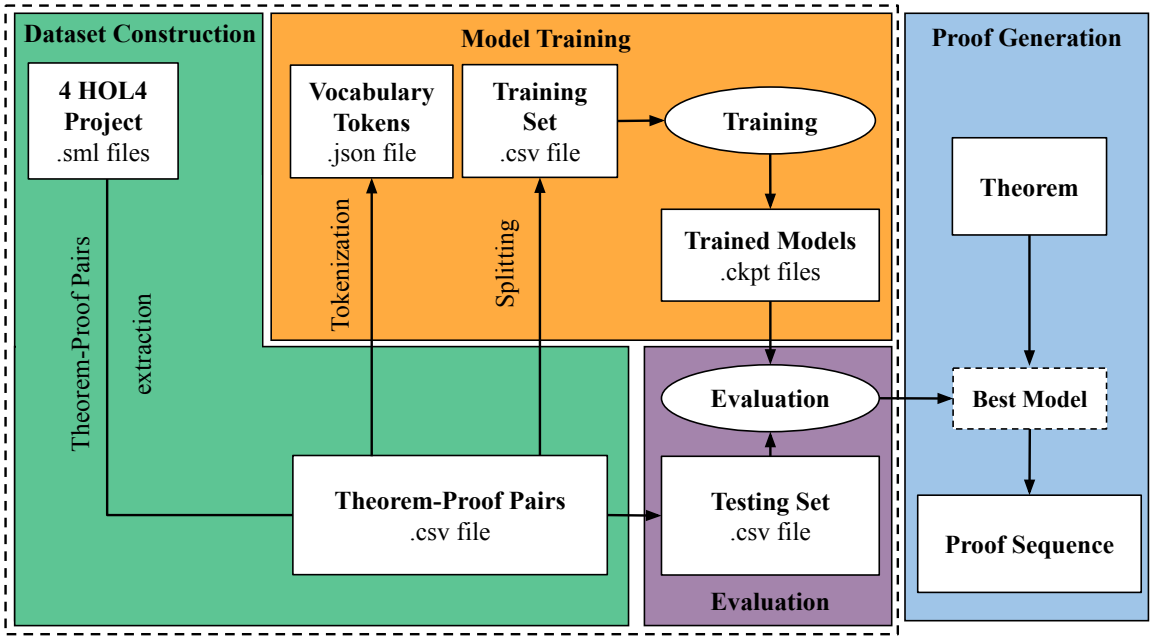

Figure 4.1: Proposed Methodology for the Proof Searching

In the model training stage, the dataset is tokenized to convert theorem statements and proof sequences into numerical representations compatible with the selected models. A vocabulary of unique tokens, stored in a `.json` file, is created during this process. The dataset is then split into training and testing sets. The selected models, such as T5 and MarianMT are fine-tuned on the training set to predict proof sequences for given theorem statements. During this process, the models learn to identify patterns and dependencies within the data to generate logical and coherent proof sequences. The trained models are saved as `.ckpt` files for later use.

To evaluate the models, the testing set is used to generate proof sequences for the theorems. These generated sequences are compared to the ground truth sequences using a similarity metric, which measures the structural and semantic alignment between the predicted and actual proof sequences. This evaluation determines the best-performing model based on its ability to produce proof sequences with high similarity to the ground truth.

In the final stage, the best-trained model is used to generate proof sequences for new theorem statements. Given an input theorem, the model predicts a sequence of tactics that form its proof. This automated process demonstrates the capability of the model to support theorem proving tasks by efficiently generating accurate and

relevant proofs. This structured workflow ensures a systematic approach to dataset preparation, model training, evaluation, and deployment for proof generation.The boxes surrounded with a discontinued line indicate processes that are performed offline, meaning that a single instance of the models will be deployed to the tool at one time, rather than requiring continuous online interaction for each operation.

4.2 Dataset

The dataset is created using proof scripts from four HOL4 projects selected from those described in the previous chapter. However, one project was excluded due to its utilization of an older version of HOL4. The size of the dataset varies across the selected projects, with Dataset 1 containing 4707 proof sequences, Dataset 2 containing 505 proof sequences, Dataset 3 containing 93 proof sequences, and Dataset 4 containing 3819 proof sequences. Note that we have not created a combined dataset as in the previous chapter because the custom script was developed to parse to extract theorem and lemma statements, along with their corresponding proof sequences. Each proof sequence represents a unique ordered list of tactics applied to complete the proof for the associated theorem or lemma.

The extraction process involves identifying theorems and lemmas in the proof scripts and collecting the exact sequence of tactics used to prove them. A proof sequence is composed of HOL4 tactics that reflect the logical steps necessary to validate the theorem or lemma. The resulting dataset consists of structured pairs, where each theorem or lemma statement is matched with its relevant tactic sequence.

To further illustrate the dataset construction for the proof search task, we use a detailed example from an HOL4 arithmetic proof script. This example demonstrates a more intricate theorem involving both multiplication and subtraction, providing insight into the complexity of theorem proofs and the tactics applied.

```
Theorem multiplication_subtraction_example:
  proves ‘forall x y. (x * y) - x = x * (y - 1)‘
  [
    REWRITE_TAC[MULT_ASSOC, MULT_1],
    IND_TAC,
    SIMP_TAC[ARITH_RULE ‘x * y - x = x * (y - 1)‘],
```

```
    ARITH_TAC,
    ASM_REWRITE_TAC[]
  ];
```

The extraction process from this proof script involves identifying the theorem statement and cataloging the series of tactics applied to prove it. Specifically, from the above theorem, we derive the theorem statement:

```
'forall x y. (x * y) - x = x * (y - 1)'
```

and we record the corresponding sequence of tactics applied during the proof as:

[REWRITE_TAC, IND_TAC, SIMP_TAC, ARITH_TAC, ASM_REWRITE_TAC]

Each tactic in this sequence plays a pivotal role in constructing the proof, reflecting the logical steps necessary to validate the theorem. These tactics are subsequently transformed into structured pairs for our dataset, demonstrating the progression from initial hypothesis to proof completion.

This example showcases the logical progression of tactics necessary for theorem proving within HOL4, from the application of rewriting rules and induction to simplification and arithmetic reasoning. By understanding and modeling these sequences, our dataset aims to enhance the capability of learning models to autonomously navigate and propose solutions in complex theorem proving scenarios.

4.3 Experimental Evaluation

This section focuses on the fine-tuning of the T5 and MarianMT models, a critical step in tailoring their sequence-to-sequence capabilities for the task of generating proof tactic sequences from theorem statements. Subsequently, we will discuss the evaluation methods used to measure the performance of these models in proof search tasks, highlighting the Levenshtein Similarity Percentage as a key metric for assessing their effectiveness in this domain. Following this, we present experimental results showcasing the impact of fine-tuning on model performance, and conclude with a comparison to potential future enhancements and advancements in the field.

4.3.1 Model Fine-Tuning

To generate proof tactic sequences from theorem statements, the T5 and MarianMT models are fine-tuned to adapt their sequence-to-sequence capabilities to this specific task. The fine-tuning process is designed to optimize the models for translating theorem statements into corresponding proof tactic sequences.

Before fine-tuning, the dataset of theorem-proof pairs is preprocessed to ensure compatibility with the models. Each theorem statement and its corresponding proof sequence are tokenized using a predefined vocabulary, converting the textual data into numerical representations. The sequences are truncated to maintain uniform lengths, and the dataset is split into training and testing subsets, with 90% of the data allocated for training and 10% reserved for evaluation. These steps ensure that the input is in a suitable format for the models to process.

The T5 model is fine-tuned by framing the task as a text-to-text problem, with theorem statements serving as input and proof sequences as the target output. Using its encoder-decoder architecture, the model learns to map the input to the output by minimizing the cross-entropy loss between the predicted and actual proof sequences. During this process, key hyperparameters are carefully adjusted: the *learning rate*, which determines the step size at each iteration of the learning process to minimize loss; *batch size*, which is the number of training samples used to train the model in one iteration; and *maximum sequence length*, the maximum limit of tokens processed by the model in one go. The fine-tuning process is carried out over multiple epochs, which are full iterations over the entire training dataset. To ensure no progress is lost, checkpoints are created periodically to preserve the best performing model.

Similarly, the MarianMT model, originally designed for machine translation, is fine-tuned to treat theorem-proof generation as a translation problem. Theorem statements are treated as the source language and proof sequences as the target language. The training objective is to minimize the cross-entropy loss, which is a measure used to quantify the difference between two probability distributions, in this case between the predicted proof sequences and the ground truth proof sequences, the actual correct sequences provided in the dataset. Hyperparameter tuning is applied to determine the best settings for learning rate, batch size, and sequence length.

For the experimentation, the T5 and MarianMT models are fine-tuned using GPUs provided by the Digital Research Alliance of Canada [82]. This process leverages

the PyTorch [83] framework, an open-source machine learning library widely used for applications such as computer vision and natural language processing, and the Hugging Face Transformers library [84], which provides a collection of pre-trained models designed for natural language understanding and generation. Utilizing these tools allows for efficient implementation, offering accelerated training and enhanced scalability to ensure the models are optimally tuned for the task.

The fine-tuning process enables both T5 and MarianMT to generate coherent and logically consistent proof sequences from theorem statements. The trained models are later evaluated to determine their effectiveness, with the best-performing models selected for deployment in proof generation.

4.3.2 Evaluation Metrics

The proof search task is evaluated using the Levenshtein Similarity Percentage [64], LSP metric specifically chosen to align with the unique requirements of the task. Predicting sequences of HOL4 tactics differs fundamentally from Natural Language Processing tasks, as it requires precise logical and sequential alignment rather than the general semantic or contextual overlaps emphasized by standard NLP metrics. The LSP is selected because it directly measures the structural and sequential correctness essential to interactive theorem proving, offering a meaningful assessment of the model's performance in generating proof sequences.

The LSP evaluates the similarity between two sequences by calculating the minimum number of edits (insertions, deletions, or substitutions) needed to transform the predicted sequence into the ground truth sequence. This metric is defined mathematically as:

$$LSP = \left(1 - \frac{Levenshtein\ Distance}{Max\ Length}\right) \times 100$$

Here, the *Levenshtein Distance* measures the number of editing operations required, and *Max Length* normalizes this value based on the length of the longer sequence. This ensures consistency across sequences of varying lengths. A similarity percentage of 100% indicates a perfect match between the predicted and ground truth sequences, while lower percentages reflect the degree of dissimilarity. By representing the result

as a percentage, the metric becomes easily interpretable for both technical and non-technical audiences, making it especially useful for comparative analyses.

The LSP is chosen because it directly evaluates the sequential correctness required for HOL4 proofs. Proofs in HOL4 consist of ordered sequences of tactics, where the logical structure and dependency between steps are critical. This metric captures structural alignment by penalizing missing or misplaced tactics in proportion to their deviation from the ground truth. Unlike binary metrics such as Exact Match Accuracy, which only indicates whether a sequence is entirely correct, the LSP accounts for partial correctness, offering a more nuanced evaluation of the model's predictions. Moreover, its ability to adapt to sequences of varying lengths ensures fair and consistent evaluation across the dataset.

While standard NLP metrics such as BLEU and ROUGE are widely used in tasks like machine translation and summarization, they are deemed unsuitable for this proof search task. BLEU, for example, relies heavily on n-gram overlaps, which are designed to measure semantic similarity rather than strict logical correctness. This makes BLEU less relevant for HOL4 proofs, where even a single missing tactic can invalidate an entire sequence. Additionally, BLEU penalizes longer sequences, which are common in this task, further reducing its applicability. Similarly, ROUGE emphasizes recall over precision, making it better suited for summarization tasks rather than tasks requiring strict ordering and logical dependency. Token-level accuracy is also avoided, as it evaluates individual tokens independently, ignoring the sequential and structural dependencies crucial to HOL4 proofs.

In conclusion, the LSP is selected as the primary evaluation metric because it addresses the unique demands of the HOL4 proof search task. It effectively measures logical similarity and sequential correctness while providing an interpretable assessment of partial correctness. By addressing the limitations of standard NLP metrics, this approach ensures a robust and meaningful evaluation framework tailored to the specific needs of interactive theorem proving, making it the most appropriate choice for this work.

4.3.3 Experimental Results

The experimental phase aims to evaluate the effectiveness of the T5 and MarianMT models in generating proof sequences from theorem statements. Following the

fine-tuning process, both models are subjected to a rigorous testing phase, where their performance is measured against a set of unseen data across four distinct datasets. The datasets are derived from the existing HOL4 projects and represent a diverse array of theorem types and proof complexities.

To assess the quality of the prediction, each generated proof sequence is first tokenized and then aligned with its corresponding ground truth sequence using a sequence alignment algorithm employing the LSP previously explained. Higher similarity percentages indicate a greater ability of the models to generate proof sequences that are not only correct in terms of tactics used but also logically coherent and applicable to the given theorem statements. Throughout the experimental evaluations, no obvious overfitting was observed. This is evidenced by the stable performance metrics when the models were applied to new, previously unseen theorem statements, ensuring that the proof generation models are reliable and generalizable across different types of proofs.

Table 4.1: Performance of the T5 Model on Various Datasets

Dataset	**Random**	**1-10**	**10-20**	**20-30**	**30-40**	**40-50**
Dataset 1	46.72%	45.66%	36.07%	36.42%	38.07%	25.61%
Dataset 2	70.93%	60.74%	55.52%	67.55%	78.04%	-
Dataset 3	52.27%	19.79%	39.46%	39.46%	14.22%	37.06%
Dataset 4	44.85%	31.40%	31.40%	-	26.23%	21.66%

Table 4.2: Performance of the MarianMT Model on Various Datasets

Dataset	**Random**	**1-10**	**10-20**	**20-30**	**30-40**	**40-50**
Dataset 1	47.05%	46.40%	34.49%	36.12%	35.68%	31.54%
Dataset 2	69.12%	61.37%	62.52%	67.88%	62.32%	-
Dataset 3	56.85%	23.09%	33.12%	29.91%	18.20%	35.56%
Dataset 4	43.96%	42.45%	30.73%	-	21.66%	21.66%

The experimental results for the T5 and MarianMT models are organized into performance metrics based on sequence lengths of theorem proofs, ranging from short sequences (1-10) to longer ones like (40-50), including a *Random* category that assesses model performance across undefined sequence lengths ranging from 1 to 512,

as detailed in Tables 4.1 and 4.2. The absence of data in certain cells of the table indicates that the dataset does not contain sequences within the relevant length range for that category. To accurately assess the models' capabilities across these varying lengths, sub testing datasets were methodically constructed. Each dataset was specifically curated to include only proofs that fell within predetermined length ranges, ensuring a targeted evaluation of model performance for each category. This approach allowed for a granular analysis of how well each model handles proofs of different complexities and lengths, revealing strengths and weaknesses in handling both shorter and more extended logical sequences.

The experimental results, illustrated in Figures 4.2 and 4.3, highlight the potential and limitations of using machine learning models, like T5 and MarianMT, for automating the interactive theorem proving process. MarianMT's high performance on the combined dataset suggests its suitability for generalized theorem proving across diverse datasets. However, its varying results on individual datasets indicate a potential need for model adjustments or specialized training to handle specific types of proof sequences or theorem complexities.

On the other hand, T5 consistently performs well across diverse datasets, underscoring its robustness and the effectiveness of its text-to-text transformation approach in handling the subtleties of theorem proving. This might suggest its utility in scenarios where a consistent level of performance is necessary across varying types of theorem statements.

MarianMT demonstrates robust performance across datasets, particularly in Dataset 1, where it achieves a peak similarity of 47.05%. However, its accuracy diminishes for longer sequences, dropping to 26.07%. In Dataset 2, MarianMT excels with a maximum similarity of 67.88% for medium-length sequences, but the absence of results for sequences exceeding 50 tactics reveals limitations in handling extended proofs. Dataset 3 presents challenges, with moderate and variable performance indicating difficulty adapting to the dataset's unique proof characteristics. Similarly, Dataset 4 shows a decline in performance as sequence length increases, highlighting the model's struggles with managing dependencies in longer proofs.

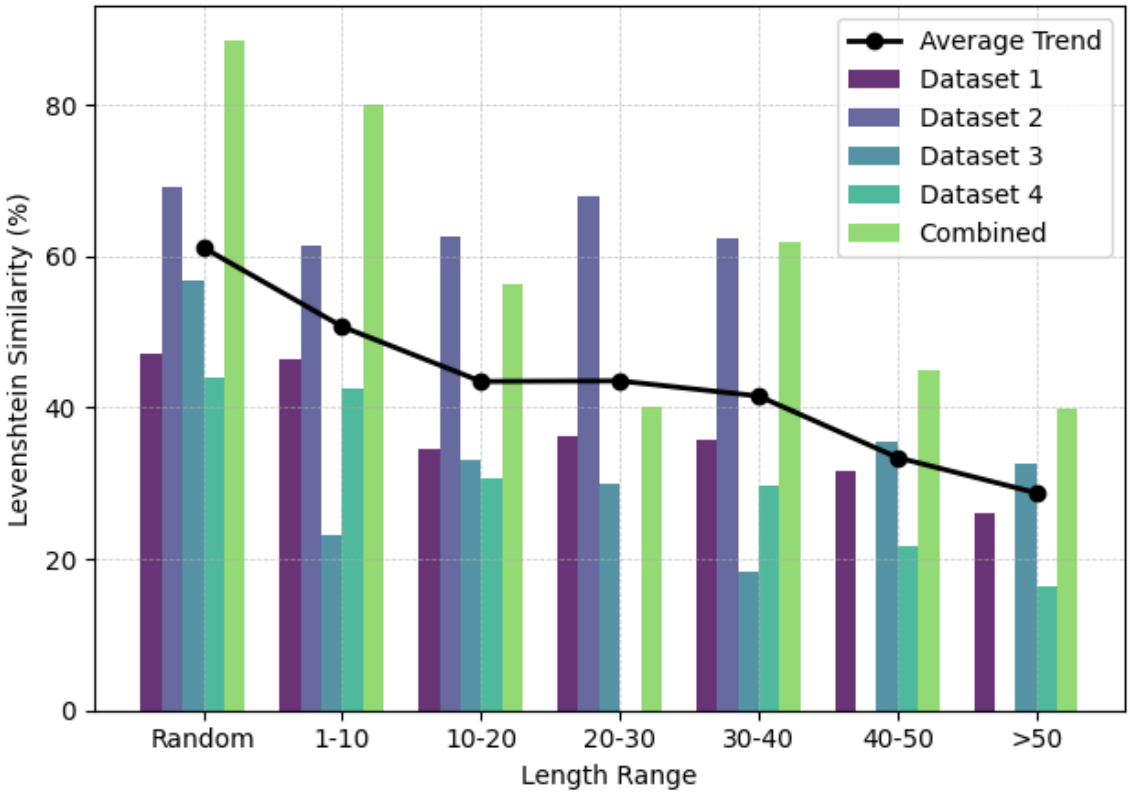

Figure 4.2: Evaluation Results for MarianMT

The T5 model displays comparable trends, achieving a peak similarity of 46.72% in Dataset 1 for shorter sequences, though its performance declines as sequences grow longer. In Dataset 2, T5 excels with a high similarity of 78.04% in the 30–40 length range, underscoring its strength in handling intermediate complexities. However, like MarianMT, T5 struggles with Dataset 3, showing inconsistent results for shorter sequences. Dataset 4 reveals similar challenges, with consistently lower performance on longer sequences, suggesting that while T5 shows promise in certain contexts, additional tuning is required to enhance its capability in tackling complex proving scenarios.

These findings not only demonstrate the feasibility of applying advanced machine learning techniques to the domain of theorem proving but also pave the way for further research into optimizing these models for enhanced accuracy and reliability in automated proof generation. Future work could explore more sophisticated metrics for evaluating proof sequence generation, delve into hybrid models that combine the strengths of T5 and MarianMT, or investigate the integration of domain-specific

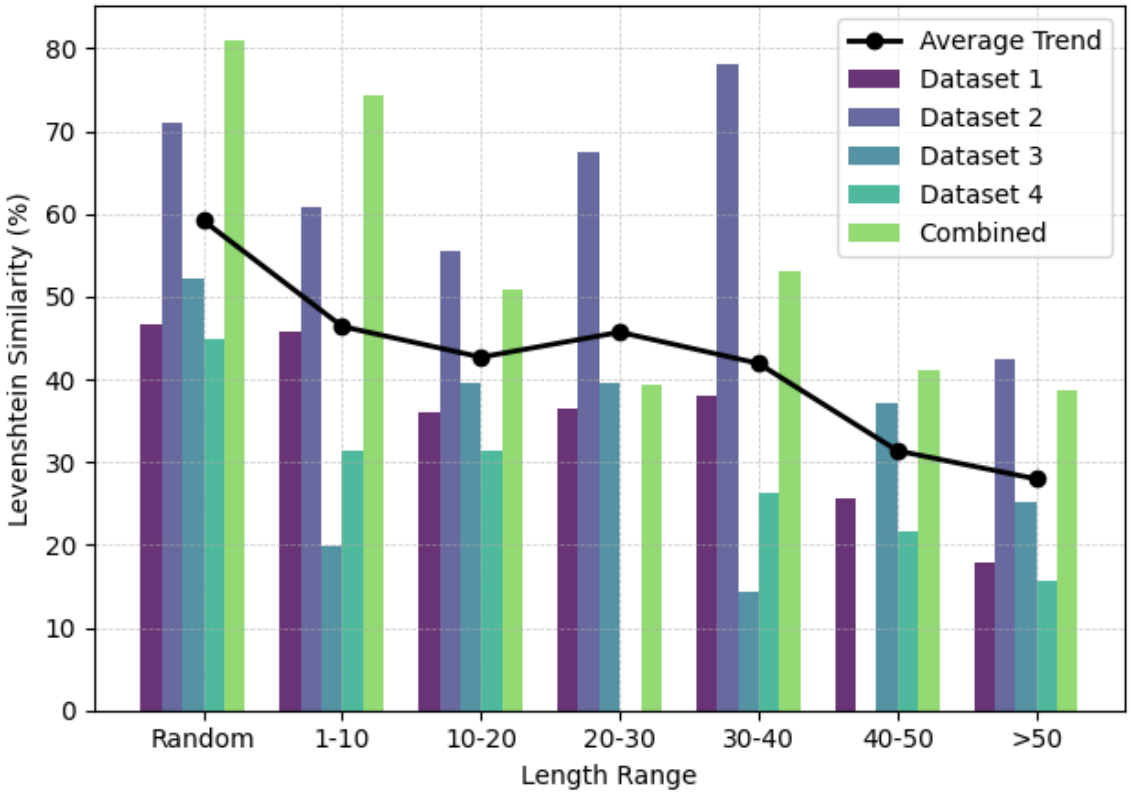

Figure 4.3: Evaluation Results for T5

knowledge into the training process to further refine the models' proof generation capabilities.

4.3.4 Comparison with Related Work

The proposed proof search methodology introduces significant advancements in both approach and outcomes when compared to existing work. By framing proof search as a sequence-to-sequence problem, this chapter uniquely leverages transformer-based models, such as T5 and MarianMT, to generate complete proof sequences. Unlike traditional methods that often rely on heuristic-driven proof tree exploration or retrieval-augmented techniques, our approach captures complex dependencies within proofs through a machine translation framework. This enables a deeper understanding of proof structures, leading to more accurate and flexible proof generation.

Targeting the HOL4 theorem prover, this work addresses the unique challenges posed by higher-order logic, such as the intricate dependencies between tactics and the diverse range of proof styles found in HOL4 libraries. While most existing works

focus on theorem provers like Isabelle, Lean, or Metamath, this work fills a critical gap by advancing proof search specifically for HOL4. The methodology provides tailored solutions to HOL4's logic framework, making it a valuable contribution to expanding the applicability of automated proof generation.

The dataset constructed for this research further sets it apart, encompassing 116,156 proof states extracted from four HOL4 libraries. This comprehensive dataset ensures robust training and evaluation, capturing diverse proof contexts and enabling the models to generalize effectively. In contrast, related works often utilize smaller or more narrowly focused datasets, limiting their adaptability to broader proving environments. The diversity of the dataset used in this chapter enhances the models' ability to handle varying proof complexities within HOL4, providing a solid foundation for automating theorem proving.

The experimental results highlight the strong performance of T5 and MarianMT in the proof search task, with peak similarity scores of 78.04% and 67.88% for medium-length sequences in Dataset 2. While these models showed limitations in handling longer sequences, such as MarianMT dropping to 26.07%, they still demonstrate the feasibility of LLMs approaches for HOL4 proof generation. Compared to related works, such as Baldur's success rate of 65.7% with search-based methods and Holophrasm's 14.3% in Metamath, it is important to note that different metrics were used. Despite this, the higher similarity scores observed in this thesis suggest that the sequence-to-sequence approach effectively captures proof structures in HOL4, a significantly more complex proving environment. These results underscore the models' capability in generating accurate proof sequences and pave the way for further refinement to address challenges with extended proofs.

In comparison to related work, we introduced an innovative and effective methodology for proof search that integrates state-of-the-art language models with a focus on HOL4. Its contributions in approach, dataset design, and performance outcomes establish a new benchmark in AI-driven theorem proving, particularly for HOL4. This work not only addresses existing gaps in proof search methodologies but also lays the groundwork for further advancements in automating complex formal verification tasks.

4.4 Summary

In this chapter, proof sequence generation is explored using sequence-to-sequence models like T5 and MarianMT. The models are trained on each dataset of HOL4 theorems and their proofs in order to generate complete proof sequences from given theorem statements. The chapter presented the proposed proof sequence generation methodology, evaluates the models' performance, and compares the results with existing work, demonstrating the effectiveness of these models in automating theorem proving tasks.
The HOL4PRS tool was developed to deploy these models for generating proof sequences. Users can input a theorem statement, and HOL4PRS utilizes the trained models to output a structured sequence of proof steps, thereby facilitating the proof construction process.

The experimental results highlight the capabilities and limitations of both T5 and MarianMT models in generating proof sequences for theorem proving. MarianMT demonstrated its robustness on the combined dataset with a high similarity score of 88.56%, showcasing its potential for generalized theorem proving across diverse datasets. However, its performance on individual datasets revealed challenges in handling complex or longer proof sequences, indicating the need for further refinement to adapt to such scenarios. T5, on the other hand, exhibited consistent performance across varied datasets and excelled with a similarity of 78.04% on Dataset 2, particularly for medium-length sequences. Despite these strengths, T5 also faced difficulties with shorter sequences in Dataset 3 and longer sequences across datasets, signaling areas for improvement in managing diverse proof structures.

Both models demonstrated promising results but require additional optimization to address their limitations. MarianMT's performance suggests it is well-suited for generalized tasks but needs fine-tuning for dataset-specific complexities. Similarly, T5's text-to-text approach proved effective in certain contexts but would benefit from enhancements to better handle edge cases and longer dependencies. These findings emphasize the potential of machine learning for automated theorem proving and provide a strong foundation for future research to refine these techniques further and expand their applicability to more complex proving tasks.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis has undertaken a critical examination of the challenges associated with interactive theorem proving, particularly within the framework of the HOL4 theorem prover. The intricate nature of formal proofs often necessitates a high level of expertise and deep domain knowledge, which can be a significant barrier for many potential users. As formal verification becomes increasingly vital in ensuring the reliability and safety of systems across various industries, addressing these usability challenges is paramount.

Our research has identified key obstacles that hinder the effective use of HOL4, including the steep learning curve associated with proof construction and the cognitive load imposed on users during the theorem proving process. To mitigate these issues, we proposed an innovative AI-driven proof recommendation system that leverages the capabilities of LLMs. This system is designed to assist users in two primary tasks: Proof Step Recommendation and (Complete) Proof Generation.

The methodologies developed in this thesis involved a systematic approach to both proof step recommendation and proof generation. For the Proof Step Recommendation task, the input to the HOL4PRS tool consists of the current proof state, which includes a sequence of previously applied tactics. The output is a set of recommended next tactics that the user can apply to progress the proof. On the other hand, in the Proof Generation task, the input is a theorem statement that the user wishes to prove. The output is a complete sequence of proof steps required to construct a

valid proof for that theorem.

Through rigorous experimentation and model fine-tuning, we demonstrated that AI can effectively assist in the process of interactive theorem proving. Our models, specifically BERT, RoBERTa, and T5, were fine-tuned on datasets derived from HOL4 theories to predict the next logical step(s) in proof construction. The RoBERTa model particularly excelled indicating a high capability of recommending accurate proof steps within the first seven suggestions. This capability not only reduces the cognitive burden on users but also streamlines the overall proof development process.

Additionally, this thesis showcased the practical feasibility of generating entire proof sequences using the capabilities of the sequence-to-sequence models T5 and MarianMT. This approach can significantly enhance the user's ability to construct complex proofs with less effort. The experimental evaluation demonstrated their effectiveness in handling shorter proof sequences, yet both models struggled with longer and more complex proofs, highlighting the need for further optimization.

The experimental results demonstrate that the integration of AI into the theorem proving workflow can lead to substantial improvements in both the accuracy of recommendations and the overall efficiency of proof generation. This work highlights the potential of AI to transform the landscape of interactive theorem proving, making it more accessible to a broader audience and facilitating its application across various industries where formal verification is paramount.

We belive that the methodologies proposed in this thesis are applicable to other theorem provers such as Coq or PVS. However, in order to apply the tool to different interactive theorem provers, comprehensive adaptation steps are essential. These include constructing tailored datasets from the target prover, fine-tuning the selected LLMs on this new data, and developing functionalities for proof step prediction and complete proof generation. Additionally, rigorous evaluation and iterative tuning of the models are required to ensure they support the unique syntax and logic of the new theorem prover.

5.2 Future Work

The findings of this research open several promising avenues for future work that could significantly enhance the capabilities and usability of the proof recommendation

system developed for the HOL4 theorem prover. One of the primary areas for further investigation is the integration of the proof recommendation and generation system directly into the HOL4 environment. Currently, the system functions as an external assistance tool, which may limit its effectiveness and usability. By embedding the system within HOL4, users could benefit from a more seamless workflow, allowing for real-time recommendations and a more intuitive interaction with the theorem prover.

Another important direction for future research is the incorporation of premise selection into the recommendation system. While this thesis has focused on predicting proof steps and generating sequences, the ability to identify relevant theorems or lemmas could significantly enhance the accuracy of proof generation. By narrowing down the search space and providing contextually relevant premises, the system could facilitate more efficient proof construction and improve overall performance.

Expanding the dataset used for training the models is also a critical area for future exploration. Currently, the dataset is limited to specific libraries within HOL4. By incorporating proofs from additional libraries or other theorem provers such as Coq, Isabelle, or Lean, the system could achieve greater generalizability and adaptability across various formal verification environments. This broader dataset coverage would not only enhance the robustness of the model but also allow it to learn from diverse proof strategies and styles.

Additionally, exploring the application of reinforcement learning could open new pathways for optimizing proof strategies and dynamically adapting tactics based on real-time feedback within the theorem proving process. This approach could further enhance proof generation and recommendation capabilities across the diverse datasets we have created.

In addition to dataset expansion, there is potential for further optimization of the models employed in the system. While transformer-based architectures have shown strong performance, exploring alternative approaches, such as reinforcement learning frameworks [85], could yield improvements in both prediction accuracy and computational efficiency. Such innovations could lead to a more responsive and effective proof recommendation system.

Finally, user experience is another vital aspect that warrants attention in future work. Conducting user studies to evaluate the practical utility of the system in real-world scenarios would provide valuable insights into its design and functionality.

Gathering feedback from both novice and expert users could help refine the system, ensuring it meets a diverse range of needs and preferences. Additionally, establishing feedback loops would allow the system to adapt to various proof styles, further enhancing its usability.

In summary, while this research has laid a solid foundation for AI integration in theorem proving, the outlined future work presents exciting opportunities to further enhance the system's capabilities, usability, and trustworthiness, paving the way for more effective formal verification processes.

Bibliography

[1] U.S. Department of Transportation. U.s. department of transportation releases results from nhtsa-nasa study of unintended acceleration in toyota vehicles. `https://www.transportation.gov/briefing-room/us-department-transportation-releases-results-nhtsa-nasa-study-unintended-acceleration`, 2019.

[2] Nancy G. Leveson and Clark S. Turner. An analysis of the Therac-25 accidents. *IEEE Computer*, 26(7), 1993.

[3] European Space Agency. Ariane 501 - presentation of inquiry board report. `https://www.esa.int/Newsroom/Press_Releases/Ariane_501_-_Presentation_of_Inquiry_Board_report`, 1996.

[4] Omar Hasan and Sofiène Tahar. Formal verification methods. In *Encyclopedia of Information Science and Technology*, pages 7162–7170. IGI Global, 2015.

[5] J. Harrison. *Handbook of Practical Logic and Automated Reasoning*. Cambridge University Press, 2009.

[6] Daniel Leivant. Higher-order logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 229–322. 1994.

[7] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[8] Yves Bertot and Pierre Castran. Interactive theorem proving and program development: Coq'art the calculus of inductive constructions. *The Computer Journal*, 49(1):130–131, 2005.

[9] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[10] Leonardo Mendona de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In *Automated Deduction*, volume 9195 of *LNCS*, pages 378–388. Springer, 2015.

[11] Sam Owre, John M. Rushby, and Natarajan Shankar. PVS: A prototype verification system. In *Automated Deduction*, volume 607 of *Lecture Notes in Computer Science*, page 748–752. Springer, 1992.

[12] Andrzej Trybulec and Howard Blair. Computer assisted reasoning with mizar. In *International Joint Conference on Artificial Intelligence*, volume 1, page 26–28. Morgan Kaufmann, 1985.

[13] Norman D. Megill and David A. Wheeler. *Metamath: A Computer Language for Pure Mathematics*. Lulu Press, 2019.

[14] Judith Crow and Ben Di Vito. Formalizing space shuttle software requirements: four case studies. *ACM Transactions on Software Engineering and Methodology*, 7(3):296–332, 1998.

[15] Sofiene Tahar and R. Kumar. A practical methodology for the formal verification of RISC processors. *Formal Methods in System Design*, 13:159–225, 1998.

[16] Anthony Fox and Magnus O. Myreen. *A Trustworthy Monadic Formalization of the ARMv7 Instruction Set Architecture*, page 243–258. Springer, 2010.

[17] Sumayya Shiraz and Osman Hasan. A library for combinational circuit verification using the HOL theorem prover. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(2):512–516, 2018.

[18] Xavier Leroy. Formal verification of a realistic compiler. volume 52, page 107–115. Communications of the ACM, 2009.

[19] Sandrine Blazy, Zaynah Dargaye, and Xavier Leroy. *Formal Verification of a C Compiler Front-End*, page 460–475. Springer, 2006.

[20] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Computer Security Foundations Workshop*, page 82–96. IEEE, 2001.

[21] Giampaolo Bella and Elvinia Riccobene. Formal analysis of the kerberos authentication system. *Journal of Universal Computer Science*, 3:1337–1381, 1997.

[22] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Toward verified artificial intelligence. *Communications of the ACM*, 66(6):82–91, June 2023.

[23] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[24] M. Abdelghany and S. Tahar. Reliability analysis of smart grids using formal methods. In *Handbook of Smart Energy Systems*, pages 1–15. Springer, 2022.

[25] Yamen Elderhalli, Osman Hasan, and Sofiène Tahar. Dynamic dependability analysis of shuffle-exchange networks. *Formal Methods in System Design*, 62(1–3):285–325, 2024.

[26] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language Models are Few-Shot Learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[27] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

[28] Daniel Huang, Prafulla Dhariwal, Dawn Song, and Ilya Sutskever. Gamepad: A learning environment for theorem proving. In *International Conference on Learning Representations*. OpenReview.net, 2019.

[29] Walter Feit and John G. Thompson. Solvability of groups of odd order. *Pacific Journal of Mathematics*, 13(3):775–1029, 1963.

[30] Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning*, volume 97, pages 6984–6994. PMLR, 2019.

[31] Alex Sanchez-Stern, Yousef Alhessi, Lawrence Saul, and Sorin Lerner. Generating correctness proofs with neural networks. In *International Workshop on Machine Learning and Programming Languages*, page 1–10. ACM, 2020.

[32] Lasse Blaauwbroek, Josef Urban, and Herman Geuvers. Tactic learning and proving for the Coq proof assistant. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 73, pages 138–150. EasyChair, 2020.

[33] Thomas M. Cover and Peter E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[34] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Symposium on Theory of Computing*, pages 380–388. ACM, 2002.

[35] Łukasz Czajka and Cezary Kaliszyk. Hammer for Coq: Automation for dependent type theory. *Journal of Automated Reasoning*, 61(1-4):423–453, 2018.

[36] Emily First, Yuriy Brun, and Arjun Guha. TacTok: semantics-aware proof synthesis. *Proceedings of the ACM on Programming Languages*, 4:1–31, 2020.

[37] Bruce T. Lowerre. *The Harpy Speech Recognition System*. PhD thesis, Carnegie Mellon University, 1976.

[38] Xiaokun Luan. Using lstm to predict tactics in Coq. In *International Conference on Software Engineering and Knowledge Engineering*, volume 2021, page 132–137. KSI Research Inc., 2021.

[39] Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C. Paulson. IsarStep: a benchmark for high-level mathematical reasoning. In *International Conference on Learning Representations*, 2021.

[40] Eric Yeh, Briland Hitaj, Sam Owre, Maena Quemener, and Natarajan Shankar. CoProver: A Recommender System for Proof Construction. In *Intelligent Computer Mathematics*, volume 14101 of *LNAI*, pages 237–251. Springer, 2023.

[41] SRI International. PVS Libraries, 2024. Available at: `https://pvs.csl.sri.com`.

[42] Kshitij Bansal, Sarah M. Loos, Markus Norman Rabe, Christian Szegedy, and Stewart Wilcox. HOList: An environment for machine learning of higher order logic theorem proving. In *International Conference on Machine Learning*, volume 97, pages 454–463. PMLR, 2019.

[43] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[44] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, and Michael Norrish. TacticToe: Learning to prove with tactics. *Journal of Automated Reasoning*, 65:257–286, 2020.

[45] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2006.

[46] Minchao Wu, Michael Norrish, Christian Walder, and Amir Dezfouli. TacticZero: learning to prove theorems from scratch with deep reinforcement learning. In *International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2024.

[47] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[48] Cezary Kaliszyk, Dennis Kühlwein, and Josef Urban. HOL(y)Hammer: Online ATP service for HOL Light and HOL4. *Mathematics in Computer Science*, 9(1):5–22, 2014.

[49] Emily First, Markus N. Rabe, Talia Ringer, and Yuriy Brun. Baldur: Whole-proof generation and repair with large language models. In *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, page 1229–1241. ACM, 2023.

[50] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo

Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

[51] Daniel Whalen. Holophrasm: a neural automated theorem prover for higher-order logic. *ArXiv*, abs/1608.02644, 2016.

[52] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Conference on Machine Learning*, pages 282–293. Springer, 2006.

[53] Norman Megill. *Metamath: A Computer Language for Mathematical Proofs*. Lulu Press, 2007.

[54] Guillaume Lample, Timothee Lacroix, Marie anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. HyperTree proof search for neural theorem proving. In *Neural Information Processing Systems*. Curran Associates Inc., 2022.

[55] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[56] Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Neural Information Processing Systems*, 2023.

[57] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

[58] Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in Lean. *ArXiv*, abs/2404.12534, 2024.

[59] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, August 2014.

[60] Alexander A. Alemi, François Chollet, Niklas Een, Geoffrey Irving, Christian Szegedy, and Josef Urban. DeepMath - deep sequence models for premise selection. In *International Conference on Neural Information Processing Systems*, page 2243–2251. Curran Associates Inc., 2016.

[61] HOL4PRS: Proof Recommendation System for the HOL4 Theorem Prover. `https://github.com/hvg-concordia/HOL4PRS`, 2024.

[62] Python Software Foundation. Python: A Programming Language. https://www.python.org/, 2024.

[63] Google. Google colaboratory. https://colab.research.google.com/, 2024.

[64] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[65] Jon Barwise. An introduction to first-order logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 5–46. Elsevier, 1977.

[66] Michael J. Gordon, Arthur J. Milner, and Christopher P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, volume 78 of *Lecture Notes in Computer Science*. Springer, 1970.

[67] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2000.

[68] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Maching Learning Research*, 21:1–67, 2019.

[69] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, pages 4171–4186. ACL, 2019.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[71] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *System Demonstrations*, pages 116–121. ACL, 2018.

[72] Dataset 1: Formal Dynamic Dependability Analysis using HOL Theorem Proving. `https://hvg.ece.concordia.ca/projects/prob-it/pr9.php`, 2024.

[73] Dataset 2: Formal Probabilistic Analysis of Wireless Sensor Networks. `https://hvg.ece.concordia.ca/projects/prob-it/wsn.php`, 2024.

[74] Dataset 3: Formal Probabilistic Risk Assessment using Theorem Proving. `https://hvg.ece.concordia.ca/projects/prob-it/pr10.php`, 2024.

[75] Dataset 4: Formal Analysis of Information Flow Using Min-Entropy and Belief Min-Entropy. `https://hvg.ece.concordia.ca/projects/prob-it/pr5.php`, 2024.

[76] Dataset 5: Formalization of Normal Random Variables. `https://hvg.ece.concordia.ca/projects/prob-it/pr7.html`, 2024.

[77] Dataset 6: Proof Searching in HOL4 with Genetic Algorithm. `https://dl.acm.org/doi/10.1145/3341105.3373917`, 2024.

[78] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[79] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. Curran Associates Inc., 2019.

[80] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*. OpenReview.net, 2020.

[81] William Falcon. Pytorch lightning. `https://github.com/PyTorchLightning/pytorch-lightning`, 2024.

[82] Digital Research Alliance of Canada. Compute Canada Resources. `https://alliancecan.ca/en`, 2024.

[83] PyTorch. Pytorch: An open-source deep learning platform. `https://pytorch.org/`, 2024.

[84] Hugging Face. The ai community building the future. `https://huggingface.co/`, 2024.

[85] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Biography

Education

- **Concordia University**: Montreal, Quebec, Canada.
  M.A.Sc., Electrical & Computer Engineering (January 2022 - December 2024)
- **National Engineering School of Sfax**: Sfax, Tunisia.
  Engineering Diploma, Computer Engineering (September 2019 - July 2022)
- **Higher Institute of Applied Sciences and Technology of Gabès**: Gabès, Tunisia. (September 2017 - July 2019)

Awards

- Mitacs Globalink Graduate Fellowship, Canada, 2023.
- Special Entrance Award, Concordia University, Canada, 2023.
- Split Merit Scholarship, Concordia University, Canada, 2023.

Work History

- **Research Assistant**, Hardware Verification Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada (2022-2024).

Publications

Conference Papers

- [**Bio-Cf1**] N. Dekhil, A. Rashid, and S. Tahar: HOL4PRS: Proof Recommendation System for the HOL4 Theorem Prover; Proc. Conference on Intelligent Computer Mathematics (CICM), Montreal, QC, Canada, pp. 352–359 (2024).

- [**Bio-Cf2**] N. Dekhil, A. Rashid, and S. Tahar: Proof Recommendation System for the HOL4 Theorem Prover; Proc. Conference on Artificial Intelligence and Theorem Proving (AITP), Aussois, France (2024).

- [**Bio-Cf3**] S. Khan, N. Dekhil, E. Mamatjan, S. Hassan, and Y. Mamatjan. An Automated Online Recommender System for Stroke Risk Assessment: Proc. Conference of The Canadian Medical and Biological Engineering Society (CMBES), Vancouver, BC, Canada (2023).

- [**Bio-Cf4**] N. Dekhil, Y. Mamatjan, S. Hassan, and M. Salih. A Novel Recommender System for Stroke Risk Stratification. Proc. Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), Ottawa, ON, Canada (2022).

Tools

- [**Bio-T1**] N. Dekhil. HOL4PRS: Proof Recommendation System for the HOL4 Theorem Prover. `https://github.com/hvg-concordia/HOL4PRS` (2024).