LEAN 4 in the AI Software Domain

Research Project-Software Engineering
2023/08 Polimi，Langze YE 10915278， Xuan ZHANG 10878008

A. Introduction

The intersection of artificial intelligence (AI) and programming languages has led to groundbreaking innovations in software development. As AI continues to evolve, the demand for languages that enable robust reasoning, formal verification, and efficient modeling is becoming increasingly essential. In this context, LEAN 4 emerges as a language of significant interest. LEAN 4, an advanced formal theorem proving language, offers a unique blend of high-order logic, formal reasoning, and functional programming capabilities. This paper explores the applications of LEAN 4 in the realm of AI software, delving into its potential to enhance logic-based reasoning, machine learning, and natural language processing. By combining formal verification and AI, LEAN 4 opens up avenues for more reliable, transparent, and secure AI systems.

B. About LEAN 4

LEAN 4 is an advanced formal theorem proving language that provides developers with powerful tools for formal reasoning and modeling. It is based on Dependent Type Theory, a powerful type system that allows developers to express rich logical relationships in code. Unlike traditional programming languages, LEAN 4's type system focuses not just on the structure and data types of the code, but also on the logical nature of the code.

An important feature of LEAN 4 is its support for higher-order logic. This means that developers can express more complex logical relationships in LEAN 4, including first-order logic, predicate logic, and more advanced forms of logic. This gives LEAN 4 an advantage when dealing with complex mathematical theorems and logic problems.

Formal proofs are another core feature of LEAN 4. It allows developers to express and verify mathematical theorems, algorithmic properties, and software behavior in a formal manner. By using formal proofs, developers can accurately prove the correctness of code, thus greatly reducing the likelihood of errors and defects.

LEAN 4 supports interactive theorem proving, which means that developers can build proofs step-by-step, interacting with the system to build complex arguments over time. This interactive approach helps developers gain a deeper understanding of the problem and ensures the accuracy of the proof.

In the field of AI software, these features of LEAN 4 provide powerful tools for formal reasoning and modeling. It can be used to express the nature of complex algorithms and models, to verify logical relationships in the reasoning process, and to ensure the correctness and security of AI systems.LEAN 4's capabilities in formal reasoning and proofs make it an indispensable tool in AI software development.

C. LEAN 4 in Logical Reasoning - Mathematical Theorem Proving

In order to demonstrate the formal theorem proving power of LEAN 4, we can consider a simple mathematical theorem, such as the law of exchange for the addition of natural numbers. In LEAN 4, we first need to represent the problem formally. We can define a function that represents the addition of natural numbers and state this exchange law theorem.

```
theorem add_comm : ∀ (a b : ℕ), a + b = b + a :=
begin

end
```

Figure 1： Represent the law of exchange for the addition of natural numbers in LEAN4

Then we need to fill in the proof steps between 'begin' and 'end'. In LEAN 4, proofs are constructed by step-by-step logical reasoning and application of known theorems. Each step of the proof requires sound reasoning based on logical rules. For the exchange law theorem, the codes are as following:

```
theorem add_comm : ∀ (a b : ℕ), a + b = b + a :=
begin
  -- Introducing natural numbers a and b
  intros a b,
  -- Analyze b by induction
  induction b with b' ih,
  -- Base case: b = 0
  { rw [nat.add_zero, nat.zero_add] },
  -- Inductive case: b = b' + 1
  { rw [nat.add_succ, ih, nat.succ_add] }
end
```

Figure 2: Proof steps for the exchange law theorem

In this case, at first, natural numbers a and b are introduced and they are the most fundamental assumptions needed for the proof process. Then we use induction to analyze the variable b. INTRODUCTION is used to make a mathematical induction that will analyze various scenarios of the variable b and will apply the appropriate reasoning steps in each case.

{ rw [nat.add_zero, nat.zero_add] } means when b is 0, we use the rw (rewrite) strategy to apply a known mathematical equation by replacing the terms on both sides of the equation. Here, we use the properties of addition of natural numbers, where **nat.add_zero** means that the result of adding

0 to a natural number, and **nat.zero_add** means that the result of adding 0 to any natural number is still that natural number.
Finally, when b is b' + 1. We use the rewrite strategy again, applying the properties of addition of natural numbers and the induction hypothesis (ih), respectively. here, **nat.add_succ** denotes the definition of the successor of addition of natural numbers, and **nat.succ_add** denotes the result of adding the successor number to the other.
Then, we're done with proof. With this example, we can find that each step must be logically rigorous and requires a proof based on a theorem already in the library. In fact, many complex mathematical theorems can be broken down into smaller steps using the LEAN4 Theorem Prover.

What's more, the interactivity of the LEAN 4 theorem prover often helps us a lot when we try to solve more complex problems.

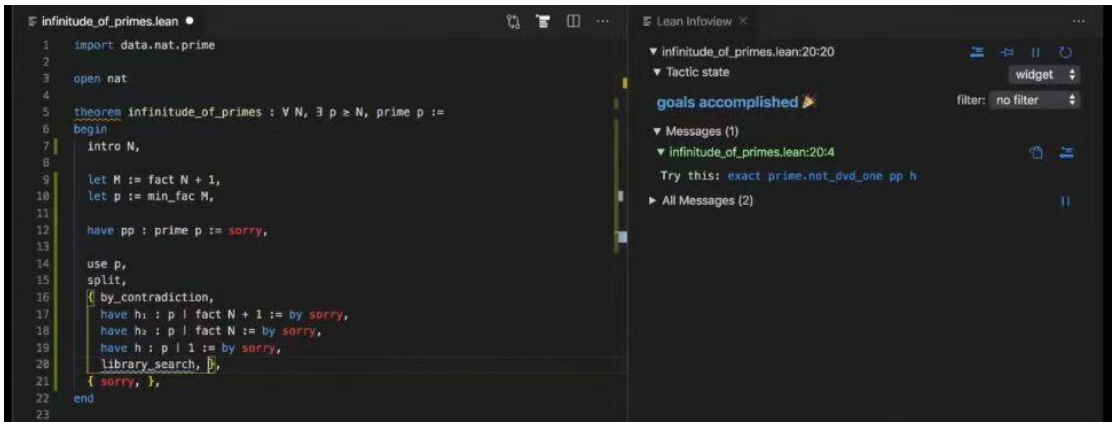

Figure 3: Prove infinity of primes in LEAN 4

For example, in the picture above, we want to prove the infinity of primes with LEAN 4. We have written a general proof framework but missing theorem support. Every red "sorry" need to be replaced by a known theorem in the library or other effective methods of proof. We can try to type "library_search" or "suggest" to find some useful theorems in the library for our proving. On the right of the picture, LEAN 4 calls a theorem inside the library in LEAN infoview to help us refine our proof logic. These constant interactions with the AI often provoke us to think more about the problem, which opens up our minds and contributes to problem solving.

It's also worth noting that, when dealing with very complex tasks, we can divide the task into many small red "sorry" and hand over to different people to complete the proof in parallel. Thus LEAN 4 Interactive theorem provers make the proof of mathematical theorems more efficient and rigorous.

D. LEAN 4 on Applications of Machine Learning

In addition to being used for mathematical theorem proving, the formal proof capabilities of LEAN 4 can be used in a wide range of machine learning applications, such as representing neural network structures and formal machine learning algorithms. The potential of LEAN 4 will be unfolded in detail below in conjunction with the code.

1. To represent of neural network structures

Neural networks are a key component in machine learning.LEAN 4 can be used to formally represent the structure of a neural network, ensuring that its specific layers, activation functions, etc. are defined and related correctly. A simplified example is shown below:

```
-- Defining Neural Network Layers
structure neural_layer :=
(input_dim : ℕ)
(output_dim : ℕ)
(weights : matrix ℚ input_dim output_dim)
(biases : vector ℚ output_dim)
(activation : vector ℚ output_dim → vector ℚ output_dim)

-- Define a simple neural network
def simple_nn : neural_layer :=
{
  input_dim := 2,
  output_dim := 3,
  weights := ⟨[1, 2, 3, 4, 5, 6], rfl⟩,
  biases := vector.vec_of_fn (λ _, 0),
  activation := λ x, x
}
```

Figure 4:represent of neural network structures in LEAN 4

In this example, we use LEAN 4's record type **neural_layer** to define the structure of a neural network layer. We define attributes such as input dimensions, output dimensions, weights, biases, and activation functions. We then create a simple neural network layer **simple_nn.**

2. Formalized Machine Learning Algorithms

LEAN 4 can be used to formalize the nature and behavior of machine learning algorithms. For example, we might consider formalizing a simple linear regression algorithm:

```
-- Simplified linear regression algorithm
def linear_regression (x y : list R) : R :=
let n := x.length in
let x_sum := list.sum x in
let y_sum := list.sum y in
let xy_sum := (x.zip y).map (λ ⟨xi, yi⟩, xi * yi) in
let x_squared_sum := (x.map (λ xi, xi * xi)).sum in
let m := (n * list.sum xy_sum - x_sum * y_sum) / (n * x_squared_sum - x_sum *
let b := (y_sum - m * x_sum) / n in
m * x.head + b
```

Figure 5: Formalized Machine Learning Algorithms in LEAN 4

This is a simple linear regression algorithm, and by formalizing it and proving some properties, we can ensure that the algorithm is correct. In addition, we can formalize other machine learning algorithms and select the optimal solutions of various algorithms through an interactive theorem prover, depending on the actual situation.

In brief, with these examples, we can see the potential of LEAN 4 in machine learning. LEAN 4 helps improve the reliability and trustworthiness of machine learning systems.

E. LEAN 4 in Natural Language Processing

LEAN 4 is also widely used in the field of Natural Language Processing (NLP) to support formal representations of natural language. It also can help with semantic analysis.

1. Formal representation of natural language

In natural language processing, the formal representation of text is key. LEAN 4 can be used to define vocabulary, syntactic structures and semantic relationships in natural language. The following is a simplified example:

```
inductive word : Type
| hello : word
| world : word
| lean : word

-- Defining Syntactic Structures
inductive sentence : Type
| simple : word → sentence
| compound : word → sentence → sentence

-- Defining Semantic Relationships
def semantics : word → string
| word.hello := "Hello"
| word.world := "World"
| word.lean := "Lean"
```

Figure 6: Formal representation of natural language in LEAN 4

In this example, we define a simplified vocabulary of "hello", "world" and "LEAN". Then, we define syntactic structures, where sentences can be simple vocabulary or compound sentences. Finally, we define a semantic function that maps the vocabulary to the corresponding string.

2. Semantic analysis

In addition to performing basic formal natural language, he can also be used to assist in semantic analysis. The following is an example for determining whether two sentences have the same semantics:

```
def same_semantics : sentence → sentence → bool
| (sentence.simple w1) (sentence.simple w2) := w1 = w2
| (sentence.compound w1 s1) (sentence.compound w2 s2) :=
  w1 = w2 && same_semantics s1 s2
| _ _ := false
```

Figure 7:Semantic analysis in LEAN 4

In this example, we define a function **same_semantics**, used to determine whether two sentences have the same semantics. Here is just a simplified example, the actual application requires more complex semantic analysis and reasoning.

F. Conclusion and analysis

With the above example, we can see that LEAN 4's impact on the AI domain lies in its formalization capabilities and interactivity. Firstly, LEAN 4 can be used as an interactive theorem prover to prove complex mathematical problems, while the interactivity it provides can deepen one's thinking about the problem. What's more, thanks to its formalization capabilities, LEAN 4 can be used to solve

complex problems in other domains as well. For example, in the field of machine learning, LEAN 4 can be used to represent neural network structures, formalize machine learning algorithms, verify model properties, etc. Similarly, LEAN 4 can formalize natural language, analyze semantics, etc., which is helpful in the field of natural language processing (NLP).

In conclusion, LEAN 4 has a number of advantages for applications in the AI software space like its formalized capacity, rigor and credibility, and its interactive nature also helps reduce manual labor.

However, every coin has 2 sides, we also realized its limitations in the process of learning LEAN. For example, LEAN 4 has a relatively steep learning curve and takes some time and effort to master. At the same time, the configuration of the compilation environment and the import of libraries are complex. In practical applications, specialized training and resources may be required. What's more, although LEAN 4 supports formalization, facing complex problems will require more expertise and programming ability. LEAN's interactive proof system also has some limitations in solving complex problems.

All in all, in my opinion, the potential for LEAN 4 to be used in the AI software space remains huge. As technology evolves, LEAN 4 may further enhance the ability to automate formalisms, thereby reducing manual intervention and better accommodating the needs of large-scale systems. Specific LEAN 4 libraries may also emerge for the AI domain, including formal machine learning algorithms, model validation tools, etc., to simplify applications. If future versions of LEAN can be stabilized with constant updates, it will be one of the most promising tools of the future!

G. References

- LEAN official website: https://leanprover.github.io/about/
- LEAN 4 quick start: https://github.com/leanprover/lean4/blob/master/doc/quickstart.md
- Prove the infinity of primes by LEAN 4: https://www.bilibili.com/video/BV1wU4y1475g/?spm_id_from=333.337.search-card.all.click&vd_source=e1e93e424f66b968a618cde73640d75a
- The LEAN 4 Theorem Prover and Programming Language: https://link.springer.com/chapter/10.1007/978-3-030-79876-5_37
- Introduction to the LEAN 4 theorem prover and programming language: https://www.youtube.com/watch?v=S-aGjgIDQZY&t=296s
- Writing Math Proofs using Lean4: https://www.youtube.com/watch?v=DXvA1qaQFtY&t=8s
- LEAN 4 overview for Mathlib users: https://www.youtube.com/watch?v=8MFGhOWeCNE
- LEAN 4 Grammar Tutorial: https://www.bilibili.com/video/BV12m4y1x7CA/?spm_id_from=333.337.search-card.all.click&vd_source=e1e93e424f66b968a618cde73640d75a
- LEAN prover zulipchat room: https://leanprover.zulipchat.com/