A Multimodal AI System: comparing LLMs and Theorem Proving Systems

Phillip G. Bradford * and Henry Orphys

Posted Date: 19 January 2026

doi: 10.20944/preprints202507.1895.v4

Keywords: first-order logic; theorem-proving; ErgoAI; Prolog; LLM; large language model; multimodal AI; neurosymbolic; logical model theory; Löwenheim–Skolem theorems; uncountable set; Lefschetz first-order logic principle; elementary equivalence

Article

A Multimodal AI System: Comparing LLMs and Theorem Proving Systems †

Phillip G. Bradford [1,2,*] and Henry Orphys [2]

[1] The University of Connecticut, Stamford, CT 06901, USA
[2] Neural Tax Networks, Inc., Connecticut, USA
* Correspondence: phillip.bradford@uconn.edu; Tel.: +01-203-252-6450
† The 10th International Conference on Information and Communication Technology for Intelligent Systems (ICTIS - 2025), Accepted and to appear, Springer, 2025. Full version submitted.

Abstract

This paper discusses a multimodal AI system applied to legal reasoning for tax law. The results given here are very general and apply to systems developed for other areas besides tax law. A central goal of this work is to gain a better understanding of the relationships between LLMs (Large Language Models) and automated theorem-proving methodologies. To do this, we suppose (1) two cases for the theorem-proving system: one where it has a countable number of total meanings for its countable number of atoms and the other is where it has an uncountabile number of total meanings for its countable number of atoms, and (2) LLMs can have an uncountable number of token meanings. With this in mind, the results given in this paper use the downward and upward Löwenheim–Skolem theorems and logical model theory to contrast these two AI modalities. One modality focuses on syntactic proofs and the other focuses on logical semantics based on LLMs. Particularly, one modality uses a rule-based first-order logic theorem-proving system to perform legal reasoning. The objective of this theorem-proving system is to provide proofs as evidence of valid legal reasoning when enacted laws are applied to particular situations. These proofs are syntactic structures that can be presented in the form of narrative explanations of how the answer to the legal question was determined. The second modality uses LLMs to analyze and transform a user's tax query so this query can be sent to a first-order logic theorem-proving system to perform its legal reasoning function. The main goal of our application of LLMs is to enhance and simplify user input and output for the theorem-proving system. Using logical model theory, we show how there can exist an equivalence between laws represented in logic of the theorem-proving system, fixed in time when the theorem-proving system was set up, and new semantics given by LLMs. These results are based on logical model theory and Löwenheim–Skolem theorems.

Keywords: first-order logic; theorem-proving; ErgoAI; Prolog; LLM; large language model; multimodal AI; neurosymbolic; logical model theory; Löwenheim–Skolem theorems; uncountable set; lefschetz first-order logic principle; elementary equivalence

1. Introduction and Motivation

This section introduces the motivation for this paper. It also includes the background that informed our approach to this work.

Neural Tax Networks [1,2] is building a proof-of-technology 'CTP — Certifiable Tax Prover' for a multimodal AI system. The first modality is based on an automated theorem-proving system for first-order logic. This is realized by using the logic programming language ErgoAI [3] for proving statements in tax law. The second modality is expected to use LLMs (Large Language Models). The primary application of LLMs is to help users input information into CTP and then help users understand the system's output or answer. This application of LLMs combined with a first-order

 doi:10.20944/preprints202507.1895.v4

logical theorem-proving systems is the focus of this paper. This applications of LLMs is challenging and it has not yet been fully tried and tested.

This paper leverages logical model theory to better understand the limits of applying these two modalities together. Particularly, this paper gives insight into the limits on combining LLMs and theorem-proving systems into a single system capable of interpreting tax situations given U.S. tax law. This supposes that law, written in natural language, can be accurately represented by first-order logic or even subsets of first-order logic used by theorem-proving systems such as ErgoAI. We make a key assumption: the LLMs work with an uncountable number of meanings over all time. We argue this is not an unreasonable assumption since deeper specialization in new areas of the law or new areas of human understanding often drives the creation of ideas or new meanings. New words are sometimes created for these new meanings, additional meanings may be associated with existing words, or meanings may exist that are not associated with words. Moreover, (1) LLMs are based on very large amounts of data that is occasionally updated, (2) linguistic models such as Heaps' law [4] allows partial estimates of the number of unique words in a document based on the document's size, which does not have an explicit bound, (3) new meanings may exist that are not associated with words for instance from diagonalization, and (4) natural language usage by speakers adds new words, or creates additional meanings of words over time. At the same time, we argue that the meanings of words must be maintained from when laws are enacted. The common meanings may fall out of use, but they must still be rectifiable.

Resolution theorem-proving systems have domains that contain at most a countable number of atoms. An atom is an identifier that represents a constant, a string, name or value. Particularly, these words and word meanings are generally established when the rules and facts are set. In addition, for applications of theorem-proving systems to legal reasoning, the rules are often set with specific word meanings in mind.

A special case of the upward Löwenheim–Skolem Theorem indicates if we have an arbitrarily large number of word or token meanings for our system, then we have an uncountable set of all word or token meanings. This may apply to LLMs but in some cases it may also apply to theorem-proving systems. Alternatively, we can take a limit over all time providing an uncountable number of word or token meanings over an infinite number of years for LLMs.

A key question we work on is: Suppose a theorem-proving system (such as a system designed to analyze U.S. tax law) does not add any new rules while new word or token meanings are added to the system's domain by an LLM. This LLM essentially "feeds" the words or phrases represented by LLM tokens into the theorem-proving system. For example, suppose a theorem-proving system based on U.S. tax law is required to answer a question input by the user using a word tokenized by the LLM as having a certain meaning. A user of our system may use contemporary jargon, where laws may remain expressed in the terminology from when the laws were enacted. Similarly, suppose a new word or a new meaning of an existing word is not in the text of the tax law that otherwise would apply to the user's question. In this case, the theorem-proving system's rules would work the same for the original domain. That is, the domain of words contained in the tax law when it was enacted. While allowing new meanings for words, word fragments, or punctuation that are tokenized by an LLM, to work with the same set of rules in the theorem-proving system. We are not giving an effective method for handling new meanings, rather we are saying one exists. Furthermore, our analysis assumes an uncountable domain of word, atom, or token meanings, a subset of which may be supplied to the theorem-proving system by the LLMs. We then apply Löwenheim–Skolem Theorems and logical model theory.

A novel application of the Löwenheim–Skolem Theorems and elementary logical model equivalence, see Theorems 7 and 8, indicates that the same rules of the theorem-proving system can still work even when an uncountable number of word, atom, or token meanings are in the theorem-proving system's domain. A similar idea is from the Lefschetz Principle of first-order logic [5]. This indicates the same first-order logic sentences can be true given specific logical models with different cardinalities.

 doi:10.20944/preprints202507.1895.v4

For instance, one logical model may have a countably infinite cardinality and another may have an uncountable cardinality. Yet, the same set of first-order logic sentences can hold true using these very different logical models.

1.1. Technical Background

LLMs are based on [6]. Discussion of LLMs can be found in individual review articles, such as [7,8]. The current paper builds on theoretical foundations, so it covers general capabilities of LLMs rather than discussing any specific LLM.

This paper seeks insight on multimodal AI systems based on the foundations of computing and mathematics. To do so we analyze the effects of the combination of a first-order logic theorem-proving system with LLMs. Combining theorem-proving, which is a classical AI methodology, with newer AI/ML methods for LLMs highlights the multimodality of our system. The multimodality of our system is neurosymbolic since it combines deep learning from LLMs and symbolic theorem-proving. Such neurosymbolic combinations have been fruitful [9].

LLMs have been augmented with logical reasoning [10]. Our work uses logical reasoning systems augmented by LLMs. There has also been work on the formalization of AI based on the foundations of computing. For example, standard notions of learnability can neither be proven nor disproven [11]. This result leverages foundations of machine learning and a classical result of the independence of the continuum hypothesis. The Turing test can be viewed as version of an interactive proof system. This view maps the Turing test to key foundations of computational complexity [12].

1.2. Technical Approach

The domain of an LLM is the specific area of knowledge or expertise that the LLM is focused on. It's essentially the context in which the LLM is trained and applied. Thus it determines the types of tasks it can perform and the kind of information it can process and generate.

Applying LLMs are in the next phase of our proof-of-technology. We already have basic theorem-proving working in ErgoAI with several tax rules. Given appropriate facts with these rules the current system derives the expected results. It is the practical combination of these modalities that will allow complex questions to be precisely answered. Ideally, given the capabilities of LLMs, the users will be able to easily work with the system.

Many theorem-proving systems supply answers with explanations. In our case, these explanations are proofs from the logic-programming system ErgoAI.

ErgoAI is an advanced multi-paradigm logic-programming system by Coherent Knowledge LLC [3,13]. It is Prolog-based and includes non-monotonic logics, among other advanced logics and features. For instance, ErgoAI supports defeasible reasoning. Defeasible reasoning allows a logical conclusion to be defeated by new information. This is sometimes how law is interpreted: If a new law is passed, then any older conclusions that the new law contradicts are dis-allowed.

 doi:10.20944/preprints202507.1895.v4

Table 1. List of select symbols and their basic definitions

Symbol	Definition
$G = < N, \Sigma, P, S >$	A context-free grammar
$\aleph_0$	Countable infinite cardinality
$\aleph_1$	Uncountable infinite cardinality of $\mathbb{R}$
κ	A cardinality number representing the cardinality of sets
σ	Non-logic symbols and operators of a logic language
LAW	Natural language law and legal clarifications
$\mathcal{L} = [L, D, \sigma]$	A first-order logic $\mathcal{L}$-language with logic symbols L, domain D , and σ
I	Logical interpretation of a first-order logic language
$\mathcal{M}_{\text{orig}}$	Logical model of first-order theorem-proving system with originalism
$\mathcal{N}_{\text{llm}}$	Logical model of an LLM
$\mathbb{N}$	The set of natural numbers $\{1, 2, \cdots\}$
$\mathbb{Z}$	The set of integers $\{\cdots, -3, -2, -1, 0, 1, 2, 3, \cdots\}$
$\mathbb{Q}$	The set of rational numbers
$\overline{\mathbb{Q}}$	The set of solutions to polynomial equations with coefficients from $\mathbb{Q}$
$\mathbb{R}$	The set of real numbers
$\mathbb{C}$	The set of complex numbers

1.3. Structure of This Paper

Section 2 discusses select previous work on multimodality and the law. It briefly reviews expert systems and approaches where LLMs can perform some level of legal reasoning. Section 3 gives a logic background illustrated by select ErgoAI statements. This section shows ErgoAI (Prolog) have at most a countably infinite number of atoms that may be substituted for variables. Section 4 introduces logical model theory to give a high-level view of how our system works. Section 5 discusses the upward and downward Löwenheim-Skolem theorems as well as elementary equivalence from logical model theory. We give results based on these foundations to show (1) LLMs and theorem-proving systems have uncountable models under certain circumstances, and (2) Models of both LLMs and the first-order logic theorem-proving systems cannot be distinguished by first-order logic expressions.

Table 1 shows key symbols and definitions.

2. Multimodality and the Law

This section reviews work relevant to automated legal reasoning. It starts by discussing expert systems and tax law. Next this section discusses the importance of explainability of answers for tax law expert systems. The last subsection outlines several previous multimodal systems for reasoning in tax law.

2.1. Expert Systems

This subsection starts discussing tax preperation systems. Then it gives background for expert systems and programming languages for the law.

There are several commercial tax preparation software systems. These systems do not backup their conclusions with proofs. Rather these systems are highly structured and focus on percise data entry. These systems focus on filling out tax forms and explaining tax law to their users.

Some LLMs are directly used to analyze taxes [33]. There are also systems that offer multimodal combinations of LLMs and logical proof systems. See for example [34].

 doi:10.20944/preprints202507.1895.v4

Expert systems simulate the decision-making and reasoning of human experts. These systems use a knowledge base with domain specific information and rules. The knowledge base, domain specific information, and rules help these systems solve complex problems.

A legal expert system uses artificial intelligence for legal reasoning. These AI systems mimic human experts in the law. They can offer guidance on specific legal issues or tasks. These systems can help lawyers, legal professionals, or the public navigate legal processes, understand legal principles, and complete legal forms. On one hand, several papers have been published discussing the limitations of expert systems when applied to the law [14–16]. On the other hand, several LLM-based legal expert systems are now on the market. Examples of these systems are the following. Reuters offers a legal drafting and analysis system [17]. Lexis Nexus offers a personalized AI expert system that acts as an assistant [18]. Eve offers an AI legal system for the full legal case lifecycles [19].

ErgoAI, developed by Coherent Knowledge LLC, is an example of a programming language that can be easily applied to legal reasoning for expert systems. There is also work on developing programming languages designed for the law. Catala is a programming language designed specifically for programming law [20]. Catala's focus is on U.S. and French law and it targets the construction of legal reasoning or expert systems.

2.2. Knowledge, Rule Bases and Inference Engines

This subsection highlights details of legal expert systems. It also discusses the importance of explainability for legal arguments especially when these are automatically generated.

Knowledge bases for tax law expert systems are usually derived from statues enacted by legislatures. Statutes are essentially legal rules written in precise natural language. In the U.S.A., regulations and clarifications are added by the U.S. Treasury Department and comparable state and local institutions. Rulings on the application of law to specific facts are given in case law and, occasionally, in private letter rulings issued by the U.S. Treasury Department. These cases and rulings are opinions governing how the language of statutes is to be interpreted and applied. Case law and governmental rulings are written in natural language.

Defintion 1 is given to make our discussion more succinct.

Definition 1 (Tax law and clarifications: LAW)**.** *The legal statutes, case law, regulations, clarifications and other rulings, in natural language, are* tax law and clarifications. *The tax law and clarifications are in a set* LAW.

An expert system's inference engine applies the system's rules and facts to the facts provided by the user to produce conclusions. Ideally, in a tax law expert system the conclusions reached by the system are the same as those which would be reached by a court or administrative body that considered the same question. Those conclusions are based on the facts of a particular situation. These facts are provided to the system and are intended to be the same facts that would be provided to a court or a government regulator if a court or regulator were to consider the same set of facts and the same question. The objective, in our case, is to derive precise conclusions that are provably accurate from the given facts.

An important requirement for the tax law system is full explainability: the answer to a query should contain the entire formal proof that leads to the conclusion. As this allows the user to understand, and have confidence, in the conclusion to an extent not possible with systems that rely solely on LLM technology. Such reasoning and explainability should also be viewed favorably to courts or administrative bodies evaluating the same questions.

2.3. LLMs and Other Approaches to Legal Reasoning

This subsection discusses applying LLMs to legal reasoning. Particularly, it briefly touches on methods that augment LLMs to help justify legal arguments.

 doi:10.20944/preprints202507.1895.v4

Large language models can be used to answer tax questions, but the responses they provide may be inaccurate. Even so, there are many applications of LLMs to the law. These legal applications include decision support systems, legal document analysis, and litigation risk assessment [21].

LLMs often struggle with logical reasonings tasks. This is not surprising because LLMs are essentially artificial neural networks trained on huge corpora of data to generate answers to natural language queries. Empirical distributions or probability play roles in LLM training. LLMs alone do not give any reasoning to justify their conclusions. Explainability for LLMs often requires other methods, or tools, and is often multimodal.

Chain-of-thought (CoT) prompting [23] is a prompt engineering technique designed to improve LLMs' performance on tasks requiring logic, calculation, and decision-making. CoT prompting does this by structuring an input prompt to mimic human reasoning. To construct a CoT prompt, insert something like "Describe your reasoning in steps" or "Explain your answer step by step" to an LLM query. This prompting technique asks the LLM both to generate a result and also to give detailed steps it used to arrive at the result.

CoT prompting can give reasoning from LLMs for simple logical reasoning tasks. These tasks may require a few reasoning steps. However, CoT often fails in complex logical reasoning tasks. CoT prompting may not provide explicit proofs. To address this weakness of CoT prompting, several neurosymbolic approaches use LLMs with theorem-proving systems on complex logical reasoning tasks [24,25] .

Symbolic Chain-of-Thought (SymbCoT) also uses multimodality to handle the logical reasoning combined with LLMs [22]. Such LLM-based frameworks integrate symbolic expressions and logic rules with CoT prompting. This is because CoT's reliance on natural language is insufficient for precise logical calculations. SymbCoT works to overcome this by using LLMs to translate natural language queries into symbolic formats, create reasoning plans, and verify their accuracy. Then SymbCoT uses the logic rules to derive its results. This multimodal combination of symbolic logic and natural language processing works so LLMs answers are more logically explainable.

SymbCoT and similar Multimodal approaches typically have the steps: (1) translation (e.g., using LLMs) from a natural language statement to a logical reasoning problem as a set of first-order logic formulas, (2) planning and executing the reasoning steps by a symbolic solver or a theorem-proving system to answer the logical reasoning problem, and (3) translation (e.g., using LLMs) of the solution of the logical reasoning problem, from the last step, back to natural language.

Another approach is to use directly explainable AI techniques for LLMs. For instance, Shapley values or Integrated Gradients may compute attributions of features impacting results of AI. For an NLP application see [26]. The Shapley values method is implemented in the *shap* Python library [27]. This library is integrated with gradient methods and implemented in Tensorflow [28] and Captum [29]. Specific tax law discussions of explainability in AI include [30–32].

3. First-Order Logic Systems

This section reviews first-order logic. This section culminates with a discussion of aspects of legal reasoning in ErgoAI.

First-order logic expressions have logical binary connectives $\wedge, \vee$ and a unary Boolean operator $\neg$. They also have variables $x_1, x_2, \cdots$ and quantifiers $\forall, \exists$. First-order logic quantifiers only apply to variables. Functions are relations that take one or more arguments from their domains and each function returns a value from its range. Functions ranges are often subsets of their domains. Predicates are relations in first-order logic take one or more arguments from the domain. All predicates have the range $\{$ true, false $\}$. It is the variables, quantifers, functions, and predicates that differentiates first-order logic from more basic logics. We assume all logic expressions are well-formed, finite, consistent, and in first-order logic.

The focus here is first-order logic proofs that are syntactic structures. For instance, some such syntatic proofs are Hilbert-style proofs [43]. First-order logic proofs can be syntatically be laid out in

 doi:10.20944/preprints202507.1895.v4

linear proof steps or as trees. The linear proof steps or trees have nodes. The nodes are connected with edges for easy understanding. Such proofs are easily presented on a computer screen. If the proof's goal is to establish an expression is provable, then this proof has this expression as its start. The nodes of a syntatic proof are facts, axioms, and expressions. The edges connect nodes by an application of an inference rule. If a logical formula is provable, then finding a proof often reduces to trial and error or exhaustive search.

The inference rule for Prolog-like theorem-proving systems for a subset of first-order logic is generally SLD-resolution [38]. To apply such inference rules may require substitution and unification. This is a mix of syntatic and semantic proofs, though ErgoAI can output a syntatic proof of provable first-order logic expressions.

Modern foundations of some tax laws are not all that different from ancient principles of tax law. Certain current U.S. tax laws are sometimes analogous to those from ancient governments [35–37]. While the meaning of words evolves, ideally the semantics of particular tax laws remain the same over time. Modern tax law may be more complex than ancient tax law, for instance because modern U.S. tax law requires millions of words to specify [39]. Nonetheless, modern tax law ideas often subsume many of the ancient principles of tax law.

Under the legal theory of Originalism, the words in the law should be given the meaning those words had at the time the law was enacted. Scalia and Garner's book is on how to interpret written laws [40, p 78]. For example, they say,

"Words change meaning over time, and often in unpredictable ways."

Assumption 1 (Originalism)**.** *The principle of* originalism *states that a law should be interpreted based on the meaning of its words when the law was enacted.*

Of course, law can be changed quickly by a legislature. Assuming originalism, the meaning of the words in the newly changed laws, when the new laws are enacted, is the basis of understanding these new laws.

Specifying meanings or words for theorem-proving systems uses atoms. Atoms are labels for constants. Atoms are made from alphanumeric characters provided the first character is a lower-case alphabetic character. Atoms also serve as labels for predicates and functions, but the focus here is on atoms representing meanings by acting as labels for constants.

Program-terms are atoms, constant, numbers, and variables. In most programming languages atoms and program-terms are specified by regular expressions. The language of a regular expression is all strings that can be generated by it. A context-free language is all strings that can be generated by a particular context-free grammar. Regular languages are context-free languages.

Program-terms can be grouped together using context-free grammars. This holds true for the syntax of atoms in ErgoAI or Prolog. A context-free grammar can represent the syntax of parenthesized expressions in ErgoAI.

In tax law, some of the inputs may be parenthesized expressions. For example, some amounts owed to the government are compounded quarterly based on the "applicable federal rate" published by the U.S. Treasury Department. Such an expression may be best written as a parenthesized expression. So, the text of all well-formed input expressions can be defined as a context free grammar. Of course, logical expressions or computer programs can only approximate the meanings of laws or tax rules. There may be a great deal of context beyond originalism that supports the meaning in laws.

An example of "context beyond originalism the supports the meaning in the natural language of law" is, for U.S. federal statutes, the reports of the House and Senate committees that drafted, in some cased modified, and ultimately approved the bills sent to the floor of the House and Senate to be voted on by the members of those chamber.

 doi:10.20944/preprints202507.1895.v4

Definition 2 (Context-free grammar (CFG)). *A context-free grammar $G =< N, \Sigma, P, S >$ where N is a set of non-terminal variables, Σ is a set of terminals or fixed symbols, P is a set of production rules so $p \in P$ is such that $p : A \Rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup \Sigma)^*$, and $S \in N$ is the start symbol.*

The expression $(N \cup \Sigma)^*$ denotes zero or more elements from the set $N \cup \Sigma$. Furthermore, ϵ is the empty string.

The language generated by a context-free grammar is all strings of terminals that can be generated by the production rules of a CFG. The number of strings in a language generated by a CFG is, at most, countably infinite.

The natural numbers are $\mathbb{N} = \{ 1, 2, \cdots \}$ and integers are $\mathbb{Z} = \{ \cdots, -3, -2, -1, 0, 1, 2, 3, \cdots \}$.

Definition 3 (Countable and uncountable numbers). *If all elements of any set T can be counted by some or all of the natural numbers $\mathbb{N}$, then T has* countable *cardinality $|T| \leq \aleph_0$. Equality holds when there is a bijection between T and a non-finite subset of $\mathbb{N}$. In this case, T is countably infinite. The real numbers $\mathbb{R}$ have cardinality $|\mathbb{R}| = \aleph_1$ which is* uncountable.

The term infinite means either countable infinite or uncountable.

Cantor showed $\aleph_1 > \aleph_0$ while founding modern set theory. The assertion that there is no cardinality between $\aleph_1$ and $\aleph_0$ is the continuum hypothesis. Gödel and Cohen showed the independence of the continuum hypothesis from a standard set of axioms for set theory [41,42,44].

It is well-known that the sets $\mathbb{N}$ and $\mathbb{Z}$ have the same number of elements. This is because there is a bijection between $\mathbb{N}$ and $\mathbb{Z}$. This bijection maps $0 \in \mathbb{Z}$ to $1 \in \mathbb{N}$, each positive number in $\mathbb{Z}$ uniquely mapped to an even number in $\mathbb{N}$ and each negative number in $\mathbb{Z}$ iks uniquely mapped to an odd number in $\mathbb{N} - \{1\}$. Clearly every element in $\mathbb{N}$ is uniquely mapped to an element in $\mathbb{Z}$. The converse also holds. Thus, $|\mathbb{N}| = |\mathbb{Z}| = \aleph_0$.

The next theorem is classical. In our case, this result is useful for applications of the Löwenheim–Skolem Theorems. See Theorem 8. . The proof of the next theorem is based on showing a correspondence between $\mathbb{Z}$ and a certain CFG.

Theorem 1 (Domains from context free grammars). *A context free grammar can generate an language with countably infinite strings.*

Proof. Consider a context-free grammar $G = (N, \Sigma, P, S)$ made of a set of non-terminals $N = \{ S, W, T, D, D_s \}$, a set of terminals $\Sigma = \{ 0, 1, 2, \cdots, 9, +, - \}$, a set of productions P, and the start symbol S. Let $\epsilon \notin \Sigma$, be the empty string.

The productions of P are:

$$\begin{aligned} S &\Longrightarrow W\, T\, D_s \\ W &\Longrightarrow -\,|\,\epsilon \\ T &\Longrightarrow 1\,|\,2\,|\,\cdots\,|\,9 \\ D &\Longrightarrow 0\,|\,1\,|\,2\,|\,\cdots\,|\,9 \\ D_s &\Longrightarrow D\, D_s\,|\,\epsilon \end{aligned}$$

So, the CFG G can generate a string corresponding to any integer $\mathbb{Z}$. All integers form a countably infinite set, completing the proof. □

The languages that can be defined by regular expressions are a proper subset of the languages that can be defined by context-free grammars. So the cardinality of the set of all atoms or expressions of atoms in ErgoAI and Prolog is at most $\aleph_0$.

 doi:10.20944/preprints202507.1895.v4

All legal expressions from the set LAW are countable. This is because each individual word in the domain of all words used in the set LAW can be uniquely numbered by a value in $\mathbb{N}$. Indeed, each word can be represented by the numerical value of their UTF-8 representation. Separately, any numerical values required for computing taxes can also be represented using integers and parenthesized expressions.

3.1. Basic Logic

This subsection starts by discussing propositional logic. The discussion then goes into the foundations of mathematical logic.

Propositional, or 0-order logic, only has propositions, or Boolean variables with no arguments. Propositional logic has no functions or variables, but propositional logic variables can be joined with other propositional variables using the AND ($\wedge$) as well as the OR ($\vee$) binary operators. A proposition variable X can also be negated $\neg X$.

A tautology is a formula that is always true. For example, consider a Boolean propositional variable X representing when an purchase is for business. So X has range $\{\text{true}, \text{false}\}$. Then

$$f \quad = \quad X \vee \neg X$$

must always be true so f is a tautology. The formula $\neg f = \neg X \wedge X$ is a contradiction. Contradictions are always false. All propositional logic expressions are either true or false. Proving a propositional logic expression to be true or false can be done mechanically by trying all possible true and false values for each Boolean variable.

Formulas in first-order logic may be provable while not being tautologies. This is because first-order logic formulas can have variables that take different values. Substituting some values for the variables in a first-order formula may make it true, while substituting other values may not make the formula true. This contributes to the complexity of provability of first-order formulas.

The statement $\vdash f$ indicates that the first-order logic formula f is syntatically provable in the logical system at hand. In our case, using a Hilbert-style proof. The set $\{ f \}$ is one or more first-order logic formulas. The general expression $\{ f \} \vdash g$ means the sentence g is syntatically provable using set $\{ f \}$, the domain of this set of formulas, and one or more inference rules.

Suppose g is a first-order logic formula. A formula may have free variables. A free variable is not bound or restricted. If a variable is quantified it is not a free variable. A quantifer Q is such that $Qx \in I, g$ where g is a first-order formula. If $Q = \forall$, then this means $\forall x \in I, g$. Or, if $Q = \exists$, then $\exists x \in I, g$.

First-order logic theorem-proving programming languages such as ErgoAI or Prolog default to $\forall x$ for any free variable x.

Definition 4 (Logical symbols of a first-order language [41–43]). *The logical symbols of a first-order language are,*

1. *variables* $x_1, x_2, \cdots$
2. *binary logic operators* $\wedge, \vee$ *and a unary logic operator* $\neg$
3. *quantifiers* $\exists, \forall$
4. *scoping using () or [] or a such-that symbol :*
5. *a concatenating symbol ,*
6. *equality* $=$

In a logic language, functions are relations giving values for asserting facts. So a function's range may be a subset of its domain. Predicates are relations whose range is $\{$ true, false $\}$. Predicates make statements about facts.

 doi:10.20944/preprints202507.1895.v4

Definition 5 ($\mathcal{L}$-language). *If $\mathcal{L} = [L, D, \sigma]$, then this is a first-order $\mathcal{L}$-language where L is the set of logical operators and quantifers, D is the domain, and σ is the signature of the system which is its constants, predicates, and functions.*

The signature σ contains $\mathcal{L}$'s constants, functions, and predicates. The signature does not include logical operators and quantifers [43].

Definition 6 (Logic term). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$, then a logic* term *is a well-formed expression made from all constants, variables, and functions using a finite number of applications of the recursive definition: If $t_1, \cdots, t_k$ are terms, then $f(t_1, \cdots, t_k)$ is also a term where $f \in \sigma$ is a $k \geq 0$-ary function.*

A term does not have logical connectives, quantifiers, and predicates.

A formula f is an expression of an $\mathcal{L}$-language. In this case, we write $f \in \mathcal{L}$. A set of one or more formulas $\{ f_i \}_{i \in \theta}$ of an $\mathcal{L}$-language is written as $\{ f \} \subseteq \mathcal{L}$, if the index set θ is understood. A formula f has no free variables if each variable in f is quantified by either $\forall$ or $\exists$.

Definition 7 (Logic formula). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$, then a logical* formula *$f \in \mathcal{L}$ is,*

F1 *$P(t_1, \cdots, t_k)$ where $P \in \sigma$ is a predicate and $t_1, \cdots, t_k$ are terms*

F2 *$t_i = t_j$ where t_i and t_j are terms*

F3 *$\neg g$ where $g \in \mathcal{L}$ is a formula*

F4 *$g \vee h$ and $g \wedge h$, where $\{ g, h \} \subseteq \mathcal{L}$ are a formulas*

F5 *$\forall x g$ and $\exists x g$ where x is a variable and $g \in \mathcal{L}$ is a formula.*

An atomic formula is any formula defined by F1 and F2.

Assigning a formula symbol f to logic expression is done using $:=$. For example $f(x) := \forall x_1 \exists x_2 [x_1 > x_2]$ indicates the formula $f(x)$ is the logic expression $\forall x_1 \exists x_2 [x_1 > x_2]$.

A formula may be neither true nor false, if the formula has free variables.

Definition 8 (Sentence). *Consider a first-order logic language $\mathcal{L}$ and a formula $f \in \mathcal{L}$. If f has no free variables, then f is a sentence.*

Atomic formulas have at most one predicate. If an atomic formula is made of only sentences, then it is a fundamental building block for logic or programming statements.

The next result is classical, see [42,43].

Theorem 2 (First-order logic is semi-decidable). *Suppose $\mathcal{L} = [L, D, \sigma]$ is a first-order logic language, and let $\{ f \} \subseteq \mathcal{L}$, then*

1. *If $\{ f \}$ is true, then there is an algorithm that can verify $\{ f \}$'s syntatic truth in a finite number of steps.*
2. *If $\{ f \}$ is false, then in the worst case there is no algorithm can verify $\{ f \}$'s syntatic falsity in a finite number of steps.*

An interpretation defines a domain. It also defines semantics for constants, functions, and predicates. Given a first-order langauge $\mathcal{L} = [L, D, \sigma]$ and $f \in \mathcal{L}$, then an interpretation that makes the sentence f true is a model.

Definition 9 (First-order logic interpretation [41,43, p. 139]). *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$ and a set I. The set I is an* interpretation *of $\mathcal{L}$ iff the following holds:*

1. *The interpretation I has a domain D_I*

 doi:10.20944/preprints202507.1895.v4

2. *If there is a constant $d \in D$, then it maps uniquely to an element $d_I \in D_I$*
3. *If there is a function $f \in \sigma$ where f takes n arguments, then there is a unique $F \in D_I$ where F is an n-ary function*
4. *If there is a predicate $r \in \sigma$ where r takes n arguments, then there is a unique n-ary predicate $R \in D_I$.*

An interpretation also defines the semantics of all elements of D_I.

An interpretation does not have variables or quantifiers. This is because interpretations give fixed meaning or semantics to logical formulas.

The semantics of an interpretation I of a first-order logic language $\mathcal{L} = [L, D, \sigma]$ is defined with natural language, mathematics, or mathematical examples. Consider $\mathcal{L}$'s domain D, signature σ, and an interpretation I. The interpretation I has domain D_I. So I can be applied to a sentence f by substituting values from D_I into $F \in D_I$, where F corresponds to $f \in \mathcal{L}$. If a variable x occurs more than once in F, then the substitution must consistently replace each occurrence of x with the same value.

Suppose, $f \in \mathcal{L}$ corresponds to $F \in D_I$ for the interpretation I. The expression $I \models f$, means values from I are substituted into $F \in D_I$ giving $F(I)$.

Definition 10 (Logical model)**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ and an interpretation I of $\mathcal{L}$, then $\mathcal{M} \subseteq I$ is a* model *iff all $\{ f \} \subseteq \mathcal{L}$ are so that each $F(\mathcal{M})$ is true, where each $F \in D_I$ corresponds uniquely with some $f \in \{ f \}$.*

Models are denoted by $\mathcal{M}, \mathcal{N}, \mathcal{J}$ and $\mathcal{K}$. Since a model $\mathcal{M}$ is an interpretation, let M be its domain D_I. Similarly $\mathcal{N}$'s domain is N. Therefore, $\mathcal{M} \subseteq \mathcal{N}$ means $M \subseteq N$ and the semantics of M coincides with the equivalent semantics in N. Even further, we assume $\mathcal{M} \subseteq \mathcal{N}$ means both $\mathcal{M}$ and $\mathcal{N}$ share σ from $\mathcal{L}$. Finally, for any model $\mathcal{M}$ the cardinality of its domain M is denoted as $|\mathcal{M}|$. So, $|\mathcal{M}| = |M|$.

The expression $\models f$, for $f \in \mathcal{L}$, indicates all interpretations make the sentence f true. Model-theoretic proofs use logical models to prove statements. If $\{ f \} \subseteq \mathcal{L}$ is a set of one or more sentences of a first-order $\mathcal{L}$-language and g is a single sentence of $\mathcal{L}$, then $\{ f \} \models g$ holds exactly when all models of $\{ f \}$ are models of g. For example, suppose an interpretation I contains the natural numbers $\mathbb{N}$ and no other integers, in its domain. Therefore, given the sentence $f(x) := \forall x \in \mathbb{N}[x^2 > 0]$, then the expression $I \models f(x)$ is true. However, if the domain of I is updated to contain the non-negative integers $\{0\} \cup \mathbb{N}$, then $I \not\models f(x)$. Going further, if $g(x) := \forall x \in \{0\} \cup \mathbb{N}[x^2 \geq 0]$, then $f(x) \models g(x)$ but $g(x) \not\models f(x)$.

Definition 11 (First-order logic semantics)**.** *Given a set of first-order logic formulas $\{ f \} \subseteq \mathcal{L} = [L, D, \sigma]$ and an interpretation I where for each $f \in \{ f \}$ then f corresponds to $F \in D_I$, then $\{ f \}$ is:*

Valid *if every $F(I)$ is true, for all $F \in D_I$*

Inconsistent or Unsatisfiable *if $I \not\models \{ f \}$*

Consistent or Satisfiable *if $\{ f \}$ is true under at least one interpretation I*

If a set of formulas is inconsistent, then anything can be proved. In syntatic proof terms, $\{ f \}$ is inconsistent iff $\{ f \} \vdash g$ and $\{ f \} \vdash \neg g$ for a sentence g. If a set of formulas is inconsistent, then anything can be proven [41]. For example, these are inconsistent formulas,

$$\begin{aligned} f_1(x) &:= \forall x \; taxable(x) \\ f_2(x) &:= \exists x \; \neg taxable(x) \end{aligned}$$

where *taxable* is a predicate.

If a set of formulas $\{ f \} \subseteq \mathcal{L}$ is valid for an interpretation I, then this is a semantic or model-theoretic proof of $\{ f \}$ restricted to I.

 doi:10.20944/preprints202507.1895.v4

Suppose $\{ f \} \subseteq \mathcal{L}$ is a set of first-order logic formulas. If $\{ f \}$ is syntatically provable, then it is valid and hence semantically provable since it satisfies all interpretations. Say $g \in \mathcal{L}$, then write $\{ f \} \vdash g$ when all models of $\{ f \}$ are models of g.

Theorem 3 (Logical soundness)**.** *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$. For any set of one or more first-order logic formulas $\{ f \} \subseteq \mathcal{L}$ and a first-order sentence $g \in \mathcal{L}$: If $\{ f \} \vdash g$, then $\{ f \} \models g$.*

Gödel's completeness theorem [43, p. 202] indicates if a set of first-order logic formulas is valid hence semantically provable, then it is syntatically provable. If g is true for all models of $\{ f \}$, then g is semantically provable.

Theorem 4 (Logical completeness)**.** *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$. For any set of one or more first-order logic formulas $\{ f \} \subseteq \mathcal{L}$ and a first-order logic sentence $g \in \mathcal{L}$: If $\{ f \} \models g$, then $\{ f \} \vdash g$.*

3.2. The models $\mathcal{N}_{llm}$ and $\mathcal{M}_{orig}$

This subsection discusses models used in the remainder of the paper. These models are for first-order theorem-proving systems and LLMs.

The logic model $\mathcal{M}_{\text{orig}}$ is based on tax law as expressed in first-order logic programs. The model $\mathcal{M}_{\text{orig}}$ encodes tax law assuming originalism. That is, $\mathcal{M}_{\text{orig}}$ fixes all word and phrase meanings from when the laws were enacted, see Assumption 1. For example, the constant *minimalTaxableIncome* can be associated with \$29,200 in 2024, but \$32,200 in 2026. So, if *minimalTaxableIncome* in $\mathcal{N}_{\text{llm}}$ is the more recent threshold of \$32,200 in 2026 but *minimalTaxableIncome* in $\mathcal{N}_{\text{llm}}$ is the previous threshold of \$29,200 in 2024. Of course, *minimalTaxableIncome* can be a function that differentiates its return value by year.

All first-order logic formulas using $\mathcal{M}_{\text{orig}}$ are sentences. All sentences are assumed to be consistent. Recall languages, such as ErgoAI or Prolog, assume universal quantification for any free variables.

The logic model $\mathcal{N}_{\text{llm}}$ uses LLMs on user input for its domain. The model $\mathcal{N}_{\text{llm}}$ should sufficiently correspond with the model $\mathcal{M}_{\text{orig}}$ for the first-order logic programs used for tax law defined using $\mathcal{M}_{\text{orig}}$. Particularly, $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ are applied to sentences are assumed to be consistent.

Theorems 3 and 4 tell us that any set of first-order sentences that has a valid model can be syntatically proved. And symmetrically, any set of first-order sentences that can be syntatically proved has a valid model so it can be semantically proved.

Theorem 2 indicates, if a set of first-order sentences $\{ f \}$ is valid, then there is an algorithm that can find a syntactic proof of $\{ f \}$. However, if a set of first-order sentences $\{ f \}$ is not valid, then in the worst case, there is no algorithm that can show $\{ f \}$ is not valid. So, given a set of true first-order sentences $\{ f \}$, we can use resolution theorem-proving algorithms to determine their truth [41]. Resolution theorem-proving can always prove valid statements in first-order logic. These valid statements must be expressed in clausal form. This can be done by direct translation or by building sufficiently equivalent clauses. This extends to the subset of first-order logic found in ErgoAI or Prolog. This is because SLD-resolution operates on a subset of first-order logic. This subset is first-order logic Horn-clauses. We are assuming that first-order logic Horn-clauses are sufficient to properly express any statement from the set LAW.

The focus here is on ErgoAI as a syntatic theorem prover for first-order logic of Horn-clauses. We use the theorem-prover in ErgoAI for Horn-clause sentences. The focus is on ErgoAI's theorem-prover for first-order logic Horn-clauses. Any first-order statement that is provable in ErgoAI has a valid model, see Theorem 3. Nonetheless, the Horn-clauses of first-order logic in ErgoAI or Prolog are still semi-decidable, so Theorem 2 applies. Particularly, all true Horn-clause sentences $\{ f \}$ are provable since Horn-clauses are a subset of first-order logic. If the Horn-clause sentences $\{ f \}$ are false, then verifying their falsity, in the worst case cannot be done. This is because functions are central to the

 doi:10.20944/preprints202507.1895.v4

unverifiability of false first-order sentences. Indeed, functions in Horn-clauses are sufficiently powerful to maintain unverifiability of false first-order sentences.

Unification and negation-as-failure add complexity to first-order logic programming interpretations for Horn-clauses [49,58]. Furthermore, facts can change. Handling a logic program's database change in first-order logic programming can be done using additional semantics [45].

3.3. Examples in ErgoAI

This subsection gives expamples using ErgoAI.

ErgoAI has frame-based syntax which adds structure to traditional Prolog statements and the frame-based syntax also includes object oriented features [3]. The ErgoAI or Prolog expression E **:-** B is a rule. This rule indicates that if the body B is true, then conclude the head E is true.

Listing 1 is an ErgoAI rule for determining if an expenditure is a deduction. The notation ?X is that of a variable. The expression ?X:Class indicates the variable ?X is an instance of Class. In this listing, the variable ?X also has Boolean properties ordinary, necessary, and forBusiness.

Listing 1: A rule in ErgoAI in frame-based syntax

```
?X:Deduction :-
  ?X:Expenditure,
  ?X[ordinary => boolean],
  ?X[necessary => boolean],
  ?X[forBusiness => boolean].
```

The rule in Listing 1 has a body indicating that if there is an ?X that is an expenditure with true properties ordinary, necessary, and forBusiness, then ?X is a deduction. This rule is taken as an axiom.

Listing 1 can be expressed in terms of provability. This is because ErgoAI facts are axioms. ErgoAI rules may be axioms or logical inference rules.

The ErgoAI code in Listing 2 has three forBusiness expenses. It also has two donations that are not forBusiness. Since these two donations are not explicitly forBusiness, by negation-as-failure ErgoAI and Prolog systems assume they are not forBusiness. The facts are the first five lines. There is a rule on the last line. Together the facts and rules form the database of axioms for an ErgoAI program.

Listing 2: Axioms in ErgoAI in frame-based syntax

```
employeeCompensation:forBusiness.
rent:forBusiness.
robot:forBusiness.
foodbank:donation.
politicalParty:donation.

?X:liability :- ?X:forBusiness.
```

A program in the form of a query of the database in Listing 2 is in Listing 3. This listing shows three matches for the variable ?X.

Listing 3: A program in ErgoAI in frame-based syntax

```
ErgoAI> ?X:forBusiness.

>> employeeCompensation
>> rent
>> robot
```

Hence an ErgoAI system can prove *employeeCompensation*, *rent*, and *robot* all are forBusiness. We can also query the liabilities which gives the same output as the bottom three lines of Listing 3.

 doi:10.20944/preprints202507.1895.v4

There are many rules that can be found in tax statutes. Many of these rules are complex. Currently there are about 10^6 word instances in the U.S. Federal tax statutes and about 2×10^6 word instances in the U.S. Federal case tax law. Of course most of these words are repeated many times, though they may be in different contexts.

4. Theorem Proving, Logic Models, and LLMs

This section focuses on logical model theory as it applies to LLMs along with theorem-proving. This section wraps up by discussing how logic programming relates to first-order logic semantics.

Definition 12 (Theory for a model). *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$ and a model $\mathcal{M}$ for $\mathcal{L}$, then $\mathcal{M}$'s* theory *is,*

$$Th(\mathcal{M}) \quad = \quad \{ f \in \mathcal{L} : \mathcal{M} \models f \}.$$

Given a model $\mathcal{M}$ of a first-order language $\mathcal{L}$, then the theory $\mathrm{Th}(\mathcal{M})$ is all true sentences from $\mathcal{L}$ for the model $\mathcal{M}$.

In the case of LLMs, tokens are translated into vectors (embeddings) in high-dimensional space. Tokens represent words, word fragments, or punctuation symbols. Each feature of a token has a dimension. In many practical cases there are thousands of dimensions [46]. Similar token vectors have close semantic meanings. This is a central foundation for LLMs [47,48].

Semantics in LLMs is based on the context dependent similarity between token embeddings [47]. These similarity measures form an empirical distribution. Over time, such empirical distributions change as the meanings of words evolve. We believe our users will use contemporary jargon to enter their legal questions. Contemporary jargon's semantics may be different from the semantics of the laws when the laws were enacted. See Assumption 1.

Given two embeddings x and y both from the set of all embeddings V and a similarity measure $s : V \times V \to [-1, +1]$. So values close to 1 indicate high similarity and values close to -1 indicate embeddings with close to the opposite meanings. The function s may be a cosine similarity measure, for instance. In any case, for an embedding x suppose there is a set of embeddings $V_x \subseteq V$ where for all $y \in V_x$, then y is similar enough to x so y can be substituted for x given a suitable threshold for a particular context.

The expression $\models^*$ indicates semantic closeness by adding that close token embeddings can be substituted for each other.

Definition 13 (Extended logical semantics for LLMs). *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$ and a set of first-order sentences $\{ f \} \subseteq \mathcal{L}$ with a model $\mathcal{M}$. Then $\mathcal{M} \models^* \{ f \}$ iff all pairs $\{a, b\} \subseteq M$, where M is $\mathcal{M}$'s domain, have similarity $s(a, b)$ that is above a suitable threshold.*

In Definition 13, the expression $\mathcal{M} \models^* \{ f \}$ means given the model $\mathcal{M}$, of similar domain elements, the sentences $\{ f \}$ are all true. The similarity score can be computed with the values in a feature table such as in Figure 3.

Consider a first-order language $\mathcal{L} = [L, D, \sigma]$, a sentence $f \in \mathcal{L}$, and a similarity function s. The next implication may fail to hold,

$$\mathcal{N}_{\text{llm}} \models f(x_1, \cdots, x_n) \quad \Rightarrow \quad \mathcal{M}_{\text{orig}} \models f(s(x_1), \cdots, s(x_n)).$$

Here is an example when this implication fails. Suppose, the similarity function s is so that $s(x_1) = s(x_2)$ for some x_1 and x_2 where $x_1 \neq x_2$. This can happen for the first-order logic formula,

 doi:10.20944/preprints202507.1895.v4

$$f(x_1, x_2) \quad := \quad \exists x_1 \, \exists x_2 [\neg (x_1 = x_2)].$$

For the formula $f(x_1, x_2)$ to be true, there must be least two distinct items in the domain of $f(x_1, x_2)$.

Consider the tax deductibility of salary and bonuses for compensation. Suppose both *salary* and *bonus* are distinct inputs for the domain of the model $\mathcal{N}_{\text{llm}}$. Further these are the only two elements in $\mathcal{N}_{\text{llm}}$. A similarity function s may make it so that $s(salary) = s(bonus)$ in preperation for theorem-proving with $\mathcal{M}_{\text{orig}}$. So there is a single meaning for both *salary* and *bonus* that is fed into the domain of the model $\mathcal{M}_{\text{orig}}$. That is, there is a single element in $\mathcal{M}_{\text{orig}}$. Indeed, in $\mathcal{M}_{\text{orig}}$ since *salary* and *bonus* are the same, so $|\mathcal{M}_{\text{orig}}| = 1$ This means, $\mathcal{N}_{\text{llm}} \models f(x_1, x_2)$ and $\mathcal{M}_{\text{orig}} \not\models f(s(x_1), s(x_2))$ since $f(x_1, x_2)$ cannot be true for a domain of a single element.

There are ways to handle such similarity functions by sacrificing relations between models. For example, consider a sentence $f \in \mathcal{L}$ where f has n arguments, then construct $f' \notin \mathcal{L}$ that has $2n$ arguments. This allows us to use a similarity function s so that $f(x)$ corresponds with $f'((x, s(x)))$. Particularly, if $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$, then we can have,

$$\mathcal{N}_{\text{llm}} \models f(x_1, \cdots, x_n) \quad \Leftrightarrow \quad \mathcal{M}_{\text{orig}} \models f'((x_1, s(x_1)), \cdots, (x_n, s(x_n))).$$

Since $f \in \mathcal{L}$ and $f' \notin \mathcal{L}$ the models $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ are not being compared under the same conditions.

4.1. How the Neural Tax Networks System Works

Our system requires translation of natural language tax law to first-order logic or ErgoAI Horn-clause programs.

Definition 14 (Knowledge Authoring)**.** *Consider a set* LAW*, of tax laws and clarifications in natural language, then finding equivalent first-order logic rules and facts is knowledge authoring. Performing knowledge authoring and placing the generated rules and facts in a set R is expressed as,* $\text{LAW} \rightarrow R$.

Automated knowledge authoring is very challenging [50]. We do not have an automated solution for knowledge authoring of tax law, even using LLMs. Consider the first sentence of U.S. Federal Law defining a business expense [51],

> "26 §162 In general - There shall be allowed as a deduction all the ordinary and necessary expenses paid or incurred during the taxable year in carrying on any trade or business, including—"

The full definition, not including references to other sections, has many word instances. The semantics of each of the words must be defined in terms of legal facts and rules.

A tax situation is entered by a user through the front end. The tax queston and facts are passed to the backend. The backend translates the tax question and facts into ErgoAI. In response, the proofs are sent to the front end and presented.

Our goal is to have users enter their questions in a structured subset of natural language with the help of an LLM-based software bot. This subset of natural language will be using contemporary jargon. The user input is the set of natural language expressions U. The set U contains user supplied facts and a tax question. The set R is made of ErgoAI facts, rules, and queries. It is consistent. The basis of R is the set LAW of tax laws and clarifications. The set R will be built using knowledge authoring, perhaps with a great deal of human labor. An LLM will help map these natural language statements and a tax question in U into ErgoAI facts and queries. These facts and queries must be compatible with the logic rules and facts in the set R. We have not yet settled on how to leverage AI to map from user entered natural language tax questions into ErgoAI facts and queries.

 doi:10.20944/preprints202507.1895.v4

Definition 15 (Determining facts and queries for R)**.** *Consider natural language facts and a tax question in a set U. We write $U[R] \models^* H$ for the map of U into into the set H of ErgoAI expressions that are compatible with the ErgoAI facts and rules R of the set* LAW*. The set H is a tax query along with the relevant facts.*

The expression $U[R] \models^* H$ indicates the model for the ErgoAI expression $U[R]$ makes the ErgoAI expression H true. The operator $\models^*$ allows similar domain elements to be substituted for each other. We can run an LLM to assist in generating H.

Using LLMs, Definition 15 depends on natural language word similarity. The words and phrases in user queries must be mapped to a query and facts compatible with our ErgoAI rules R so ErgoAI can work to derive a result.

Definition 14 and Definition 15 serve as a foundation for the next definition.

Definition 16. *Consider the ErgoAI rules and facts R representing the set* LAW *and the compatible ErgoAI user tax question has a query and facts $U[R]$. The query and facts are written as first-order logic sentences in H where $U[R] \models^* H$ since they must be compatible with the rules R. Now a theorem prover such as ErgoAI can determine whether, $R \vdash H$.*

Figure 1 shows a high-level vision of the Neural Tax Networks CTP system. Currently, we are not doing the LLM mapping automatically.

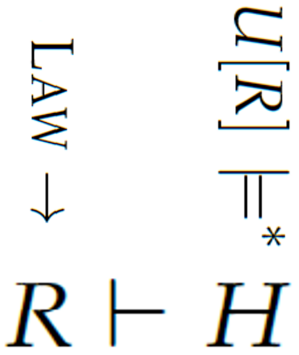

Figure 1. A relationship of syntax and semantics for our multimodal AI system CTP

Figure 1 depicts knowledge authoring as LAW $\rightarrow R$. Since LAW is the set of natural language laws and clarifications, they are based on the semantics of when the said laws were enacted by Assumption 1. Next, natural language facts and a tax question are placed in a set U. We expect the natural langauge in U to be in contemporary jargon. Then U is converted into first-order ErgoAI expressions in a set H which are compatible with R. That is, we will use an LLM to assist constructing H where $U[R] \models^* H$ holds. Finally, a theorem-proving system tries to find a proof $R \vdash H$, if H is provable. A new set H will be computed each time a user asks a new tax question.

The boxes in Figure 2 indicate parts of our processing that may be repeated many times in one user session. Always in the order: a user enters their facts and queries in U and then the system uses an LLM to help compute H so that $U[R] \models^* H$. Next, the system attempts to prove $R \vdash H$. This figure highlights knowledge authoring, the semantics of LLMs, and syntatic provability.

 doi:10.20944/preprints202507.1895.v4

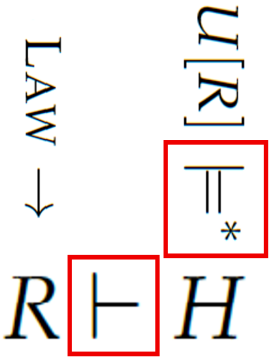

Figure 2. Highlights of the repeated syntactic proofs and semantic LLM models in our multimodal AI CTP system

Suppose H is created so that $U[R] \models H$. Theorem 3 indicates that if $R \vdash H$, then $R \models H$. In other words, if R is valid in the theorem-proving model $\mathcal{M}_{\text{orig}}$ and if H is constructed so that $\mathcal{N}_{\text{llm}} \models H$, then

$$\mathcal{M}_{\text{orig}} \models H \quad \text{and} \quad \mathcal{N}_{\text{llm}} \models H$$

both hold.

To validate our system we have built a number of functional tests. These mimic user entered tax situations where the tax answers are known. As we add LLMs, we will add testing with similar words and measure how the system performs. These measurements will first verify that the proofs, presented on the front end, are correct.

4.2. ErgoAI Examples

This section gives some tax-law related examples of ErgoAI. We use first-order logic capabilities for Horn-clauses in ErgoAI in frame-mode.

If there is an ErgoAI rule or axiom:

$$?X1, \cdots, ?Xn : E \quad \text{:-} \quad ?X1, \cdots, ?Xn : B$$

so that E is the head and B is the body. Where E holds if B is true. Such rules are all in the set R.

The expression $?X1, \cdots, ?Xn : B$ indicates all variables $?Xi$, for $i : n \geq i \geq 1$, are class instances of B.

Consider the rule,

$$F(?X) \quad = \quad \text{?X:liability} \ \textbf{:-} \ \text{?X:forBusiness.} \tag{1}$$

The model $\mathcal{M}$ has domain M, so

$$M \quad = \quad \{ \text{ cloudStorage:forBusiness, virtualMachine:forBusiness } \}.$$

Given $\mathcal{M}$ and $F(?X)$, the next statement is true:

$$\mathcal{M} \quad \models \quad F(?X).$$

That is, the interpretation $\mathcal{M}$ makes *cloudStorage:liability* and *virtualMachine:liability* true. Hence $\mathcal{M}$ is a model for $F(?X)$. Alternatively consider the model $\mathcal{M}'$ and its domain M'

 doi:10.20944/preprints202507.1895.v4

$$M' \quad = \quad \{\, \textit{foodbank:donation}, \textit{politicalParty:donation} \,\},$$

then $\mathcal{M}' \not\models F(?X)$ is true. This is because there is no ?X in the model $\mathcal{M}$ so that ?X:forBusiness in $F(?X)$.

A simplified example for deducting a lunch expense highlights key issues for computing with LLMs. Consider a model $\mathcal{M}$ and its domain M,

$$M \quad = \quad \{\, \textit{salad:Expenditure}, \ \textit{burger:Expenditure} \,\}.$$

This is a model for a lunch expenditure. Features of salads and burgers can include: cost, calories, health-scores, and so on. These food features are in a feature table such as Figure 3. But, a *zillion-dollar-frittata*, costs about 200 times the cost of a salad or burger. So the *zillion-dollar-frittata* is not an ordinary expenditure. For example, the *zillion-dollar-frittata* property or field ordinary disqualifies it from being deductible. All three of these *salad, burger* and *zillion-dollar-frittata* may have some close dimensions, but they are not that similar since their cost dimension (feature) is not in alignment.

5. Löwenheim–Skolem Theorems and Elementary Equivalence

This section gives the main results of this paper. Its final subsection gives highlights of the Neural Tax Networks system.

Our theorem-proving system syntatically proves valid theorems in first-order logic. These results suppose tax law and clarifications are in a set LAW. The set LAW is translated to first-order logic suitable for a legal theorem-proving system, such as ErgoAI.

We assume the semantics of the law has an at most countable set of meanings from when the laws were enacted. This is assuming originalism for the legal rules in a theorem-proving system for first-order logic. The culmination of this section then shows: if new semantics are introduced by LLMs, then a first-order theorem-proving system will be able to prove the same results from the original semantics as well as compatible new semantics introduced by the LLMs. The defeasible reasoning in theorem-proving systems such as ErgoAI, may be able to help when there is a semantic mismatch.

In traditional logic terms, first-order logic cannot differentiate between the original theorem-proving semantics and the new LLM semantics, presuming the new semantics introduced by the LLM are compatible with the semantics for the theorem-proving system. Of course, since Horn-clause logic is a subset of first-order logic it also cannot differentiate between the original theorem-proving semantics and the new LLM semantics.

Theorem 5 (Special-case of the upward Löwenheim–Skolem [43]). *Consider a first-order set of formulas* $\{\, f \,\} \subseteq \mathcal{L} = [L, D, \sigma]$. *If* $\{\, f \,\}$ *has a countably infinite model, then* $\{\, f \,\}$ *has an uncountable model.*

Michel, et al. [53] indicates that English adds about 8,500 new words per year. See also Petersen, et al. [54]. Originalism requires the retention of the meanings of words from when laws are passed. The meanings, from different eras, may not remain in common use, but these meanings remain available for lookup. So, we assume words or meanings do not leave a natural language, rather they may fade from common use. These word meanings all remain a part of natural language given a suitable context. Since we are proposing automating semantic lookup, word meanings must be maintained by context. Particularly, this can be done so we can perform legal reasoning assuming originalism. Also a word may change its meaning in common use while the word remains in common use but just with a different commonly-understood meaning.

A meaning is represented by a set of feature values. For example, Figure 3 shows meanings represented by feature values in each row of a feature table. Each row represents one word, token, or atom. In discussing meanings from a language perspective, we use words for the rows of the feature

 doi:10.20944/preprints202507.1895.v4

table. For LLMs the rows in the feature table are tokens. Feature tables for theorem-proving systems have atoms in their rows.

	...	Feature t_j	Feature t_{j+1}	Feature t_{j+2}	Feature t_{j+3}	...
$\vdots$						
w_{i-2}		$m_{i-2,j}$	$m_{i-2,j+1}$	$m_{i-2,j+2}$	$m_{i-2,j+3}$	
w_{i-1}		$m_{i-1,j}$	$m_{i-1,j+1}$	$m_{i-1,j+2}$	$m_{i-1,j+3}$	
w_i		$m_{i,j}$	$m_{i,j+1}$	$m_{i,j+2}$	$m_{i,j+3}$	
w_{i+1}		$m_{i+1,j}$	$m_{i+1,j+1}$	$m_{i+1,j+2}$	$m_{i+1,j+3}$	
w_{i+2}		$m_{i+2,j}$	$m_{i+2,j+1}$	$m_{i+2,j+2}$	$m_{i+2,j+3}$	
$\vdots$						

Figure 3. A subset of a feature table showing several words and a few features

To keep our discussion simple, we will say each row represents a natural language word. If a word w_i has two meanings, then it will be represented in two rows $w_{i,1}$ and $w_{i,2}$. Each row has a different meaning. Theorem 6 is based on adding new words with new meanings, adding new meanings to current words, or just adding new meanings. There are several ways we can represent new meanings: (1) a new meaning can be represented as a set of features with different values from any existing row, or (2) a new meaning may require new features.

We assume there will always be new meanings that require new features. So, we assume the number of feature columns is countable over all time. Just the same, we assume the number of words an LLM is trained on is countable over all time. In summary, over all time, we assume the number of rows and columns in this figure are both countably infinite. Words that are synonyms have the same feature sets, but these words will be in different rows. There may even be features based on etymology or spelling. Homonyms are each individually listed with their different feature sets since homonyms have different meanings.

The number of meanings is uncountable based on the next argument. Over all time, we can diagonalize over all words and and all features. Figure 3 shows a word w_i whose features must be different from any other word with a different meaning. Therefore, any new word or meaning or an additional meaning for a word is $w' \neq w_i$. So w' is a new word or new meaning and it will never have the same feature values of any of the other words or meanings. There may even be new meanings that are not associated with any word. So, by diagonalization, if there is a countable number of feature columns and a countable number of rows, there must always be additional meanings not listed.

In the case of Neural Tax Networks, some of these new words represent goods or services that are taxable. Ideally, the tax law and clarifications will not have to be changed for such new taxable goods or services. These new word meanings will supply new logical interpretations for tax law.

Assumption 2. *In natural language some words will get new meanings over time. Also new words with new meanings will be added to natural language over time.*

For LLMs, assuming languages always add new words and meanings over time, then it can be argued that natural language has an uncountable model if we take a limit over all time. This is even though this uncountable model may only add a few expressions related to tax law each year. To understand the limits of our multimodal system, our assumption is that such language growth and change goes on forever. Some LLMs currently train on 10^{12} word instances or tokens. This is much larger than the number of atoms used by many theorem-proving systems. Comparing countable and uncountable sets in these contexts may give us insight.

Assumption 1 states the original meaning of words for our theorem-proving system is fixed. In other words, the meaning of the words in law is fixed from when the laws were enacted. We assume originalism since we recognize that the meaning of words can change over time. An example is

 doi:10.20944/preprints202507.1895.v4

provided in Reading Law: The Interpretation of Legal Texts by Scalia and Garner [40]. Under the section entitled "Semantic Canons/Fixed Meaning Canon" Scalia and Garner note that the meaning of words change over time. Furthermore their meanings often change in unpredictable ways. They give as an example the statement attributed to Queen Anne about the architecture at St. Paul's Cathedral was "awful, artificial and amusing." By "awful" she meant "awe inspiring." This contrasts with how the word "awful" is typically used today, in which it does not convey a positive impression of something but instead connotes a negative feeling about the thing being described. Thus, as Scalia and Garner state, it would be quite wrong to ascribe the Queen's 18th century statement about the architecture of St. Paul's Cathedral the 21st century meaning of her words.

Although this is a somewhat extreme example, it clearly shows that to properly determine the meaning of the words of a statute (and thus apply the statue in accordance with its terms) the statute must be interpreted and applied using meaning of the words at the time the statute was written. This is because originalism is the only way to determine what was intended by the legislative body that enacted the statute.

Homonyms can arise over time by having words used for one meaning at a particular point in time take on a second meaning when used in other contexts. The meanings of such words in legal writing can be determined by application of the "whole text" canon discussed by Scalia and Garner[40], which requires the reader to consider the entire text of the document in which the word is used in order to determine its meaning. This is done in conjunction with the "presumption of consistent usage" canon, which states that a word or phrase is presumed to bear the same meaning throughout a text, and that a material variation in terms suggests a variation in the meaning. This is Salia and Garner's "Presumptive of Consistent Use" canon, [40, p. 170]. Taken together, these rules of statutory interpretation and application allow an expert system of the type being developed to use the capabilities of LLMs to determine, based on context, potential dual meanings. This will determine with an extremely high degree of accuracy what meaning should be assigned to a word. By way of example, the sentence "the light poll is bent and must be replaced" can readily be distinguished from the sentence "the voting poll closes at 8 PM." This is only by the use of a different adjective immediately before the word "poll" but also by the use within the same sentence of the word "bent" (in the case of the "light poll") and "closes" (in the case of the voting poll).

5.1. Model theory

Theorem 1 shows context-free grammars can specify a countable infinite domains for theorem proving systems. This is by constructing a countable number of atoms, program-terms or expressions. Using a similarity measure s suppose each word or token vector x has a subset of similar tokens V_x where $|V_x| \leq c$, for a constant integer $c \geq 1$.

Theorem 6. *Taking a limit over all time, LLMs with similarity sets of constant bounded sizes have $\aleph_1$ meanings.*

In some sense, Theorem 6 assumes knowledge will be extended forever. This assumption is based on the idea that as time progresses new meanings will continually be formed. This appears to be tantamount to assuming social constructs, science, engineering, and applied science will never stop evolving.

The next definitions relate different models to each other. The term 'elementary' can be interpreted as 'fundamental.'

Definition 17 (Elementary extensions and substructures [41])**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$. Let $\mathcal{M}$ and $\mathcal{N}$ be models of $\mathcal{L}$ and suppose $\mathcal{M} \subseteq \mathcal{N}$.*

Then $\mathcal{N}$ is an elementary extension *of $\mathcal{M}$ or $\mathcal{M} \preceq \mathcal{N}$ iff every first-order sentence $f \in \mathcal{L}$ is so that*

$$\mathcal{M} \models f(x) \quad \Leftrightarrow \quad \mathcal{N} \models f(x),$$

 doi:10.20944/preprints202507.1895.v4

Also, if $\mathcal{M} \preceq \mathcal{N}$, then $\mathcal{M}$ is an elementary substructure *of $\mathcal{N}$.*

As before, $\mathcal{M} \subseteq \mathcal{N}$ presumes the semantics of $\mathcal{M}$ coincides with the equivalent semantics in $\mathcal{N}$.

Definition 18 (Elementary equivalence [41])**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$. Let $\mathcal{N} \equiv \mathcal{M}$ mean $\mathcal{M}$ and $\mathcal{N}$ are* elementary equivalent *models of $\mathcal{L}$. Then $\mathcal{N} \equiv \mathcal{M}$ iff every first-order sentence $f \in \mathcal{L}$ is so that*

$$\mathcal{M} \models f(x) \quad \Leftrightarrow \quad \mathcal{N} \models f(x).$$

Given a model $\mathcal{N}$ of a first-order language, then $\mathrm{Th}(\mathcal{N})$ is the first-order theory of $\mathcal{N}$. See Definition 12.

Theorem 7 (Elementary and first-order theory equivalence [41])**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ and two of its models $\mathcal{M}$ and $\mathcal{N}$, then*

$$\mathcal{N} \equiv \mathcal{M} \quad \Leftrightarrow \quad Th(\mathcal{N}) = Th(\mathcal{M}).$$

A special case of the Lefschetz Principle of first-order logic [5,57] states: The field $\overline{\mathbb{Q}}$ of solutions of polynomial equations whose coefficients are from $\mathbb{Q}$ and the field of complex numbers $\mathbb{C}$ have an elementary equivalence as described in Theorem 7. The field $\overline{\mathbb{Q}}$ contains the roots of all polynomials with coefficients from $\mathbb{Q}$. So, $\overline{\mathbb{Q}}$ contains algebraic numbers that may be complex. There is a countable infinite number of polynomials with coefficients from $\mathbb{Q}$, hence $|\overline{\mathbb{Q}}| = \aleph_0$. There is an uncountable number of complex numbers $\mathbb{C}$ that are not roots of polynomial equations with coefficients from $\mathbb{Q}$. so $|\mathbb{C}| = \aleph_1$. Recall $\aleph_1 > \aleph_0$.

This means sentences that are true in $\overline{\mathbb{Q}}$ with cardinality $\aleph_0$ are also true in $\mathbb{C}$ with cardinality $\aleph_1$. That is, by Theorem 7, we have

$$\overline{\mathbb{Q}} \equiv \mathbb{C} \quad \Leftrightarrow \quad \mathrm{Th}(\overline{\mathbb{Q}}) = \mathrm{Th}(\mathbb{C}).$$

Theorem 7 indicates if there is an elementary equivalence between $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ with respect to the legal rules and facts, then they both models support the same legal theory. This theorem also illuminates our system. Suppose a user input U is compatible with first-order logic rules and regulations R of our theorem-proving system. The first-order facts, rules, and an intepretation are in the set $U[R]$. LLMs may be applied to help build the ErgoAI set H where $U[R] \models^* H$. The ErgoAI in H is computed with (e.g., cosine) similarity along with any additional logical rules and facts. Assumption 2 indicates there is a countable model for H.

The next version of the Löwenheim–Skolem Theorems is from [41,55,56]. See also [52].

The symbol κ is a cardinal number. Cardinal numbers represent the cardinality of sets. For instance $\kappa = \aleph_0$ and $\kappa = \aleph_1$ are two possible assignments for κ.

Theorem 8 (Löwenheim–Skolem (L-S) Theorems)**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$.*

Upward *The model $\mathcal{K}$ is infinite and $|\mathcal{K}| = \kappa$, then there is a model $\mathcal{J}$ so that $|\mathcal{J}| > \kappa$ and $\mathcal{K} \preceq \mathcal{J}$.*

Downward *The model $\mathcal{K}$ is infinite and $|\mathcal{K}| = \kappa$, then there is a model $\mathcal{J}$ so that $|\mathcal{J}| \leq \kappa$ and $\mathcal{J} \preceq \mathcal{K}$.*

Theorem 8 shows the existence of elementary extensions and elementary substructures. It does not give effective methods to generate these relationships. Therefore this theorem is stated in great generality, not to mention it is expressed using infinite sets.

 doi:10.20944/preprints202507.1895.v4

A first-order language $\mathcal{L} = [L, D, \sigma]$ with an countably infinite model $\mathcal{M}_{\text{orig}}$ can suitably encode tax law and clarifications from the set LAW. This is because tax law is written down. Thus, it must be countable, even considering that over-all time, the set LAW may become of at most countably infinite cardinality.

The next corollaries applies to tax law as well as other areas. There are two cases of interest:

- Both $|\mathcal{M}_{\text{orig}}| = \aleph_0$ and $|\mathcal{N}_{\text{llm}}| = \aleph_0$. This assumes a logic model $\mathcal{M}_{\text{orig}}$ with a countable number of meanings and atoms by Theorem 1. In logic terms, this model's there are a countably infinite number of constant domain elements in $\mathcal{M}_{\text{orig}}$. It also assumes LLMs with only a countably infinite number of meanings and tokens.
- $|\mathcal{M}_{\text{orig}}| = \aleph_1$ and $|\mathcal{N}_{\text{llm}}| = \aleph_1$. This assumes the model $\mathcal{M}_{\text{orig}}$ with an uncountable number of meanings for a countable number of atoms. Likewise, it assumes the model $\mathcal{N}_{\text{llm}}$ for LLMs with an uncountably number of meanings by Theorem 6 for a countable number of tokens.

The type of theorem-proving resolution or SLD-resolution has no impact on the next results. The central focus is on the models.

Corollary 1 (Application of Upward L-S)**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ with a infinite model $\mathcal{M}_{orig}$ for a first-order logic theorem-proving system where $|\mathcal{M}_{orig}| = \kappa$. Then there is a model $\mathcal{N}_{llm}$ so that $\mathcal{M}_{orig} \subseteq \mathcal{N}_{llm}$ and $\mathcal{M}_{orig} \preceq \mathcal{N}_{llm}$ where $|\mathcal{N}| > \kappa$.*

Proof. Apply Theorem 8 (upward) to $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ where $|\mathcal{M}_{\text{orig}}| = \kappa$. Since $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$ we assume both of these models share σ from the language $\mathcal{L}$.

The upward L-S theorem indicates there exists a model $\mathcal{N}_{\text{llm}}$ so that $\mathcal{M}_{\text{orig}} \preceq \mathcal{N}_{\text{llm}}$. □

Applying Corollary 1 requires $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$ so the semantics of $\mathcal{M}_{\text{orig}}$ carries over to $\mathcal{N}_{\text{llm}}$. We express this by saying both $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ share σ from $\mathcal{L}$.

The case of interest to tax law is,

- $|\mathcal{M}_{\text{orig}}| = \aleph_0$ and $|\mathcal{N}_{\text{llm}}| = \aleph_1$ so $|\mathcal{M}_{\text{orig}}| < |\mathcal{N}_{\text{llm}}|$. This assumes logic model with only a countable number of meanings and atoms by Theorem 1. Alternatively, in logic terms, this model's constant domain elements are at most countably infinite. It also assumes LLMs with an uncountably number of meanings by Theorem 6.

Corollary 2 (Application of Downward L-S)**.** *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ and model $\mathcal{N}_{llm}$ so that $|\mathcal{N}_{llm}| = \kappa$ for a LLM passing its output to a first-order theorem proving system. Then there is a model $\mathcal{M}_{orig}$ so that $\mathcal{M}_{orig} \subseteq \mathcal{N}_{llm}$ and $\mathcal{M}_{orig} \preceq \mathcal{N}_{llm}$ where $|\mathcal{M}_{orig}| \leq \kappa$.*

Proof. Apply Theorem 8 (downward) to $\mathcal{N}_{\text{llm}}$ and $\mathcal{M}$ where $|\mathcal{N}_{\text{llm}}| = \kappa$ Since $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$ we assume both of these models share σ from the language $\mathcal{L}$.

Suppose we have an uncountable model $\mathcal{N}_{\text{llm}}$ based on the uncountability of LLM models by Theorem 6. That is,

$$|\mathcal{N}_{\text{llm}}| \quad = \quad \aleph_1.$$

The downward Löwenheim–Skolem Theorem indicates $\mathcal{M}_{\text{orig}} \preceq \mathcal{N}_{\text{llm}}$ where $|\mathcal{M}_{\text{orig}}| < |\mathcal{N}_{\text{llm}}|$ and $|\mathcal{M}_{\text{orig}}| \leq \kappa$. This completes the proof. □

To apply this corollary, we assume $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ share σ from $\mathcal{L}$.

Continuing the case with ($|\mathcal{M}_{\text{orig}}| = \aleph_0$ or $|\mathcal{M}_{\text{orig}}| = \aleph_1$) and $|\mathcal{N}_{\text{llm}}| = \aleph_1$. There is an equivalence between an LLM's uncountable model and a first-order countable or uncountable logic model. This equivalence is based on the logical theory of each of these models. In other words, a consequence of Theorem 7 is next.

 doi:10.20944/preprints202507.1895.v4

Corollary 3. *Consider a language* $\mathcal{L} = [L, D, \sigma]$ *and the models* $\mathcal{M}_{orig}$ *and* $\mathcal{N}_{llm}$*, where* $\mathcal{M}_{orig} \subseteq \mathcal{N}_{llm}$ *and* $\mathcal{M}_{orig}$ *and* $\mathcal{N}_{llm}$ *share* σ *from* $\mathcal{L}$*. Then*

$$\mathcal{N}_{llm} \equiv \mathcal{M}_{orig} \quad \Leftrightarrow \quad Th(\mathcal{N}_{llm}) = Th(\mathcal{M}_{orig}).$$

This is the ideal case where $\mathcal{M}_{\text{orig}}$ and $\mathcal{N}_{\text{llm}}$ support the same theories.

5.2. Certifiable Tax Prover architecture sketch

This subsection gives a basic architectural sketch of our proof-of-technology CTP.

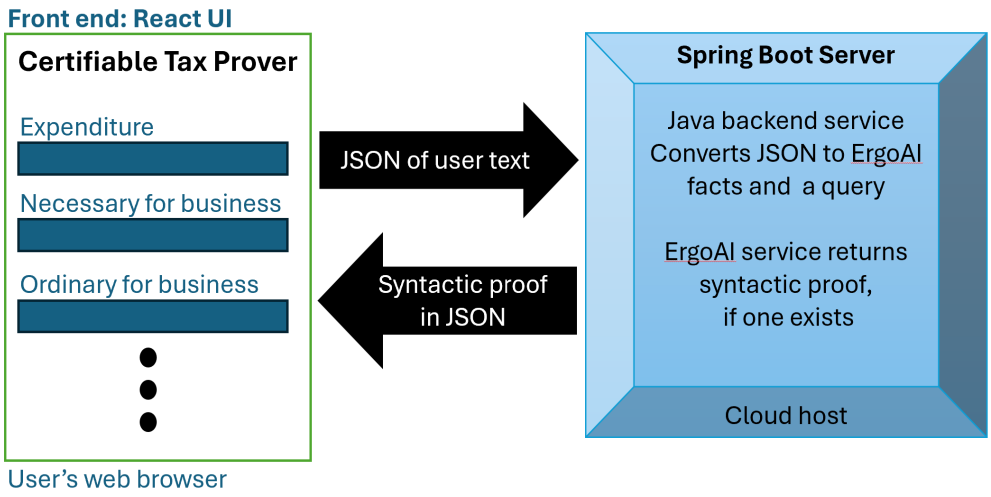

Figure 4. Client-server interaction for the CTP system

CTP is built on a cloud-based client-server architecture. The UI is a thin client since only user data entry and authentication is done on the client side. The computation is all on the server side. The server side is built as a set of microservices. The microservices will include,

1. User roll management
2. LLM AI chat assistant
3. Tax law natural language text server
4. Tax law query builder
5. ErgoAI proof engine

See Figure 5.

Currently, our CTP has a basic highly structured user interface that allows a user to enter facts. The front end translates these facts into JSON. The JSON is sent to the backend which builds queries with the user-entered facts.

The proofs are computed by ErgoAI on our backend. Our backend is in the cloud. Figure 4 provides a sketch the main client-server exchange.

 doi:10.20944/preprints202507.1895.v4

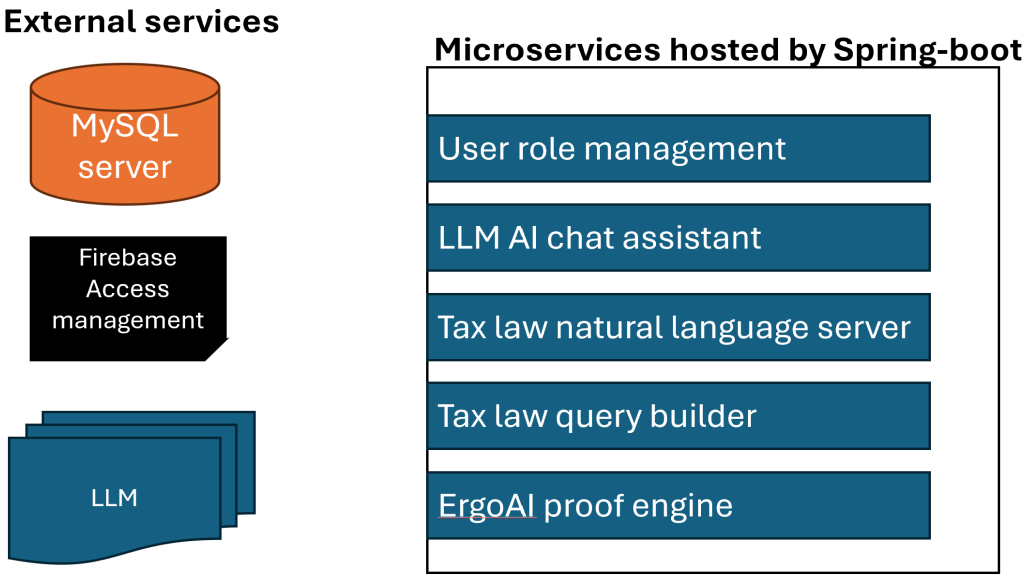

Figure 5. The planned CTP system external services and microservices

The front end is a React UI system. This React UI has users fill out details of their tax questions. We will have a bot that assists the user. This bot will be managed from the backend. The bot will work with the user on the front end to enter contemporary jargon in to the CTP system. This contemporary jargon will be passed to the backend so it can be translated into suitable language for the theorem-prover.

The backend runs Java in a Spring-boot server. This server hosts all of the microservices. This Spring-boot server is running Tomcat. Currently a single service of the Tomcat server translates the JSON from the front end into ErgoAI. Then this same service calls the ErgoAI theorem prover on the inputs.

The Spring-boot server backend receives queries in JSON from the front end. These JSON queries are mapped to ErgoAI and executed by the Java backend. The answers are passed back to the React front end and presented using React. In the future, we will separately build the tax law query in ErgoAI in one service and search for the proof in another service.

Of course, our architecture must deal with the semi-decidability of first-order logic or first-order Horn-clause logic. See Theorem 2. ErgoAI has several features to lower the chances of infinite cycles while trying to prove false sentences. Nonetheless our system must handle any, unplausible situation, gracefully.

6. Discussion

This paper aims to give a better understanding of multimodal AI. Particularly, the two modal systems discussed here are LLMs and first-order logic theorem-proving systems. The first-order theorem-proving systems we are using are from ErgoAI or Prolog. We discuss general features of LLMs.

Often mathematical limits are over infinite domains. For example, $\lim_{x\to\infty} f(x)$, for a mathematical function $f(x)$. In discussing LLMs, our limit is over the infinite time domain. Figure 6 shows this limit assuming an annual linear increase of meanings of about 8,000 new meanings per year. This infinite time domain goes forever forward in time. The purpose of this discussion is to give better understand and reason about the multimodality of our system. This is just as the foundations of computer science has given us a much better understanding of physical computers. Where many times the foundations of computer science assume machines with arbitrarily large or infinite capacity.

 doi:10.20944/preprints202507.1895.v4

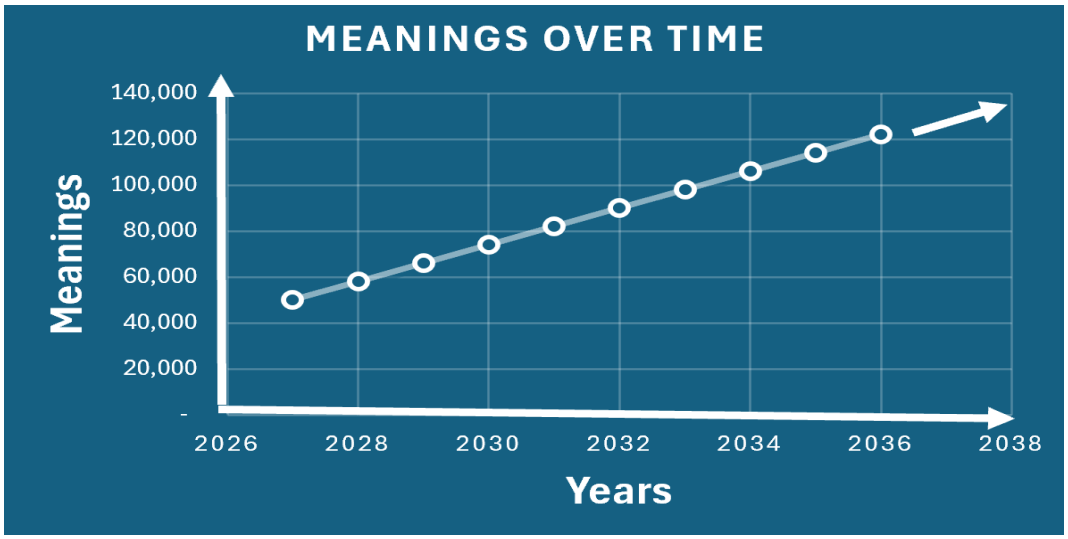

Figure 6. An illustration of the limit of meanings over time

Even though all humans are mortal, these results may give insight to their experience due to the domains of the two modalities of our multimodal system. It is germane, since LLMs train on very large data sets. Currently, these data sets can be as large as 10^{12} word instances or tokens. In contrast, the number of facts and rules in U.S. tax law and clarifications, LAW, presently requires about 3×10^6 word instances.

Consider an uncountable model or number of words or meanings from LLMs by Theorem 6. Corollary 2 indicates such an uncountable model can work with a theorem-proving system using a countably infinite model. This is very interesting in light of originalism for the law in Assumption 1. Furthermore, a few of the rules and principles of tax law are similar to those from 1,900 years ago. This indicates some semantics remains over a long time, even in different languages and in different eras.

Some LLMs have more than ten thousand dimensions for their feature sets. Although the number of dimensions for feature sets has been growing with new LLMs, there is also research to constrain the number of dimensions. Consider this in contrast to our assumption of countably infinite feature sets.

To apply the logical model theory, when the two models are related as $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$, then the semantics of $\mathcal{M}_{\text{orig}}$ must coincide with the semantics of $\mathcal{N}_{\text{llm}}$. Defeasible reasoning may be applied during theorem-proving to essentially enhance semantic compatibility.

One classical interpretation of the upward Löwenheim–Skolem Theorem is that first-order logic cannot distinguish higher levels of infinity. That is, first-order logic cannot differentiate sets with distinct cardinalities of $\aleph_i$, for $i \geq 0$. It is also widely discussed that first-order logic cannot make statements about subsets of its domain [59]. This has impact on differentiating LLMs and theorem-proving systems assuming originalism.

Our use of the Löwenheim–Skolem Theorems and our discussion of Lefschetz Principle sheds light on originalism.

We also mention the classical notion that there is a countable number of algorithms that can be listed. Yet, it is possible to diagonalize over these algorithms showing there must be algorithms not in this list. This discrepancy is rectified by results based on uncomputable numbers such as part 2 of Theorem 2. Meanings, in the sense we discuss here, may not tie into algorithms.

7. Conclusions

We presented a high-level description of a multimodal expert system for tax law to answer tax questions in natural language based on the tax law and clarifications. Already, the results our proof-of-technology system produces are fully explainable, i.e. providing a full proofs of the logical reasoning based on a small subset of tax law. We use the first-order logic theorem-proving system in ErgoAI. Although our use of ErgoAI is restricted to the Horn-clause subset of first-order logic. We do

 doi:10.20944/preprints202507.1895.v4

not see any loss of accuracy in representing laws or their clarifications with this subset of first-order logic.

To effectively achieve a goal, the system will incorporate a database of all tax laws. It will have an LLM enhanced interface to load the data, translate the questions into logical queries and facts, and translate the generated line of reasoning in natural language. So the answers can be presented as proofs. These syntactic proofs are tied together with the semantics of LLMs extending the theorem-proving semantics. Theoretical results on the limits of logical reasoning in such systems are presented.

Acknowledgements

We thank Dmitry and Nadia Udler for detailed discussions and their insightful comments.

Paul Fodor of Stony Brook University and Coherent Knowledge helped a great deal with ErgoAI. His insights on knowledge authoring were also helpful.

Thanks to the referees for their detailed and thoughtful work.

Author Contributions: Conceptualization P.B.; System Conceptualization, H. O.; All authors have read and agreed to the published version of the manuscript.

Funding: The bulk of this research was funded by Henry A. Orphys. We also had partial support from the U.S. National Science Foundation (NSF) through award # 2018873: CAREERS: Cyberteam to Advance Research and Education in Eastern Regional Schools.

Institutional Review Board Statement: There was no need for an Institutional Review Board.

Data Availability Statement: There is no data to report for this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Neural Tax Networks. Available online: https://neuraltaxnetworks.com/ (accessed on 13 April 2025).
2. Orphys, H.A.; Jaworski, W.; Filatova, E.; Bradford, P.G. Determining Correct Answers to Tax and Accounting Issues Arising from Business Transactions and Generating Accounting Entries to Record Those Transactions Using a Computerized Logic Implementation. U.S. Patent 11,295,393 B1, 5 April 2022.
3. Swift, T.; Kifer, M. Multi-paradigm Logic Programming in the ErgoAI System. Proceedings of *Logic Programming and Nonmonotonic Reasoning: 17th International Conference (LPNMR 2024)*, Dallas, TX, USA, October 11–14, Dodaro, C., Gupta, G., Martinez, M. V., Eds.; Springer-Verlag: Berlin, Heidelberg, 2024, 126–139. https://doi.org/10.1007/978-3-031-74209-5_10
4. Egghe, L. Untangling Herdan's law and Heaps' law: Mathematical and informetric arguments. *J. Am. Soc. Inf. Sci.*, 2007 *58*, 702-709. https://doi.org/10.1002/asi.20524
5. Marker, D. *Model Theory: An Introduction*, Springer: New York, USA, 2002. https://doi.org/10.1007/b98860
6. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. Adv. Neural Inf. Process. Syst. 2017, 30, 1–11. https://dl.acm.org/doi/10.5555/3295222.3295349
7. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Akhtar, N.; Barnes, N.; Mian, A. A Comprehensive Overview of Large Language Models. ACM Trans. Intell. Syst. Technol. 2023, 14, 1–38. https://dl.acm.org/doi/10.1145/3744746
8. Minaee, S.; Mikolov, T.; Nikzad, N.; Chenaghlu, M.; Socher, R.; Amatriain, X.; Gao, J. Large Language Models: A Survey. arXiv 2024, arXiv:2402.06196. https://doi.org/10.48550/arXiv.2402.06196
9. Barbierato, E.; Gatti, A.; Incremona, A.; Pozzi, A.; Toti, D. Breaking Away from AI: The Ontological and Ethical Evolution of Machine Learning. *IEEE Access* **2025**, *13*, 55627–55647, https://doi.org/10.1109/ACCESS.2025.3553032
10. Cheng, F.; Li, H.; Liu, F.; van Rooij, R.; Zhang, K.; Lin, Z. Empowering LLMs with Logical Reasoning: A Comprehensive Survey. arXiv 2025, arXiv:2502.15652. Available online: https://doi.org/10.48550/arXiv.2502.15652 (accessed on 21 July 2025).
11. Ben-David, S.; Hrubeš, P.; Moran, S.; Shpilka, A.; Yehudayoff, A. Learnability Can Be Undecidable. Nat. Mach. Intell. 2019, 1, 44–48. https://doi.org/10.1038/s42256-018-0002-3

12. Bradford, P.G.; Wollowski, M. A Formalization of the Turing Test. ACM SIGART Bull. 1995, 6(4), 3–10. https://dl.acm.org/doi/10.1145/222267.222268
13. Coherent Knowledge LLC: http://coherentknowledge.com/ (Accessed 2025-04-13).
14. Leith, P. The Rise and Fall of the Legal Expert System. Eur. J. Law Technol. 2010, 1(1). https://doi.org/10.1080/13600869.2016.1232465
15. Franklin, J. Discussion Paper: How Much of Commonsense and Legal Reasoning Is Formalizable? A Review of Conceptual Obstacles. Law, Probab. Risk 2012, 11(2–3), 225–245, DOI: 10.1093/lpr/mgs007 https://doi.org/10.1093/lpr/mgs007
16. Isozaki, I. Literature Review on AI in Law. Medium, 27 January 2024. Available online: https://isamu-website.medium.com/literature-review-on-ai-in-law-7fe80e352c34 (accessed on 6 September 2025).
17. Thomson Reuters. CoCounsel Essentials: AI Legal Drafting and Analysis Tool; Thomson Reuters: 2025. Available online: https://legal.thomsonreuters.com/en/products/cocounsel-essentials (accessed on 15 August 2025).
18. Lexis+AI https://www.lexisnexis.com/en-us/products/lexis-plus-ai.page (accessed on 16 November 2025).
19. Eve https://www.eve.legal/ (accessed on 16 November 2025).
20. Merigoux, D.; Chataing, N.; Protzenko, J. Catala: A Programming Language for the Law. Proc. ACM Program. Lang. 2021, 5(ICFP), 77, 1–29. https://doi.org/10.1145/3473582
21. Ejjami, R. AI-Driven Justice: Evaluating the Impact of Artificial Intelligence on Legal Systems. Int. J. Multidiscip. Res. (IJFMR) 2024, 6(3), 1-29. https://doi.org/10.36948/ijfmr.2024.v06i03.23969
22. Xu, J.; Fei, H.; Pan, L.; Liu, Q.; Lee, M.-L.; Hsu, W. Faithful Logical Reasoning via Symbolic Chain-of-Thought. arXiv 2024, arXiv:2405.18357v2, https://doi.org/10.48550/arXiv.2405.18357
23. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.H.; Le, Q.V.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* **2022**, *35*, 24824–24837. https://doi.org/10.48550/arXiv.2201.11903
24. Ye, X.; Chen, Q.; Dillig, I.; Durrett, G. SatLM: Satisfiability-Aided Language Models Using Declarative Prompting. Adv. Neural Inf. Process. Syst. 2024, 36, 1–13. https://doi.org/10.48550/arXiv.2305.09656
25. Kirtania, S.; Gupta, P.; Radhakrishna, A. Logic-LM++: Multi-Step Refinement for Symbolic Formulations. arXiv 2024, arXiv:2407.02514 https://doi.org/10.48550/arXiv.2407.02514
26. Goldshmidt, R.; Horovicz, M. TokenSHAP: Interpreting Large Language Models with Monte Carlo Shapley Value Estimation. In Proceedings of the 1st Workshop on NLP for Science (NLP4Science), 2024; pp. 1–8. https://doi.org/10.18653/v1/2024.nlp4science-1.1
27. Lundberg, S.M.; Lee, S.-I. SHAP: SHapley Additive exPlanations. GitHub Repository, 2025. Available online: https://shap.readthedocs.io/en/latest/ (accessed on 30 June 2025).
28. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D.G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. arXiv 2016, arXiv:1603.04467. Available online: https://arxiv.org/abs/1603.04467 https://doi.org/10.48550/arXiv.1603.04467 (accessed on 29 June 2025).
29. Pothukuchi, R.; Morcos, A.; Prasanna, S.; Sundararajan, M.; Varma, R.; Yan, D.; Nair, V.; Nori, H.; Srinivas, S.; Ramachandran, P.; et al. Captum: A Model Interpretability Library for PyTorch. Facebook AI Research 2019. Available online: https://captum.ai/ (accessed on 5 August 2025).
30. Górski, Ł.; Kuźniacki, B.; Almada, M.; Tyliński, K.; Calvo, M.; Asnaghi, P. M.; Almada, L.; Iñiguez, H.; Rubianes, F.; Pera, O.; Nigrelli, J. I. Exploring Explainable AI in the Tax Domain. Artif. Intell. Law 2024, 32, Article Published 07 May 2024, https://doi.org/10.1007/s10506-024-09395-w
31. Kuźniacki, B.; Almada, M.; Tyliński, K.; Górski, Ł.; Winogradska, B.; Zeldenrust, R. Towards eXplainable Artificial Intelligence (XAI) in Tax Law: The Need for a Minimum Legal Standard. 2022. https://doi.org/10.2139/ssrn.4198246
32. Inter-American Center of Tax Administrations. Reviewing the Explainable Artificial Intelligence (XAI) and Its Importance in Tax Administration; CIAT: 18 October 2023. Available online: https://www.ciat.org/reviewing-the-explainable-artificial-intelligence-xai-and-its-importance-in-tax-administration/?lang=en (accessed 2025-06-27)
33. Nay, J.J.; Karamardian, D.; Lawsky, S.B.; Tao, W.; Bhat, M.; Jain, R.; Lee, A.T.; Choi, J.H.; Kasai, J. Large Language Models as Tax Attorneys: A Case Study in Legal Capabilities Emergence. Phil. Trans. R. Soc. A. 2024, 382. http://doi.org/10.1098/rsta.2023.0159

34. Pan, L.; Albalak, A.; Wang, X.; Wang, W.Y. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. arXiv 2023, arXiv:2305.12295 https://doi.org/10.48550/arXiv.2305.12295
35. Kerr D.; Lassila D.; Smith KT.; Smith LM. Historical Development of Taxation From Ancient Times to Modern Day: Implications for the Future. Journal of Accounting and Finance. 2025;25(1):53-76. https://doi.org/10.33423/jaf.v25i1.7510
36. Duncan-Jones, R. Money and Government in the Roman Empire; Cambridge University Press: Cambridge, UK, 1994, https://doi.org/10.1017/CBO9780511552632
37. Lidz, F. How to Evade Taxes in Ancient Rome? A 1,900-Year-Old Papyrus Offers a Guide. The New York Times 2025. Available online: https://www.nytimes.com/ (accessed on 14 April 2025).
38. Lloyd, J.W. Foundations of Logic Programming; Symbolic Computation; Springer: Berlin/Heidelberg, Germany, 1987. https://doi.org/10.1007/978-3-642-83189-8
39. Partlow, J. The necessity of complexity in the tax system. WYo. L. REv.. 2013;13:303. https://doi.org/10.59643/1942-9916.1298
40. Scalia, A.; Garner, B.A. Reading Law: The Interpretation of Legal Texts; Thomson West: St. Paul, MN, USA, 2012.
41. Ebbinghaus, H.-D.; Flum, J.; Thomas, W. Mathematical Logic, 3rd ed.; Springer: Cham, Switzerland, 2021, https://doi.org/10.1007/978-3-030-73839-6
42. Shoenfield, J.R. Mathematical Logic; Association for Symbolic Logic: Storrs, CT, USA; A.K. Peters Ltd.: Natick, MA, USA, 1967.
43. Hodel, R.E. An Introduction to Mathematical Logic; Dover: New York, NY, USA, 1995.
44. Cohen, P.J. Set Theory and the Continuum Hypothesis; Dover: New York, NY, USA, 1994.
45. Bonner, A.J.; Kifer, M. An Overview of Transaction Logic. Theor. Comput. Sci. 1994, 133(2), 205–265, https://doi.org/10.1016/0304-3975(94)00114-9
46. FastText. FastText: Library for Efficient Text Classification and Representation Learning; Facebook AI Research, 2025. Available online: https://fasttext.cc/ (accessed on 15 April 2025).
47. Wolfram, S. What Is ChatGPT Doing and Why Does It Work?; Wolfram Media: Champaign, IL, USA, 2023.
48. Tunstall, L.; Von Werra, L.; Wolf, T. Natural Language Processing with Transformers; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.
49. Dix, J. A Classification Theory of Semantics of Normal Logic Programs: I. Strong Properties. Fundam. Inform. 1995, 22(3), 227–255. https://doi.org/10.3233/FI-1995-2233
50. Gao, T.; Fodor, P.; Kifer, M. Knowledge Authoring for Rule-Based Reasoning. In Proceedings of the ODBASE, OTM Conferences, Valletta, Malta, 22–26 October 2018; pp. 461–480. https://doi.org/10.1007/978-3-030-02671-4_28
51. Legal Information Institute. 26 U.S. Code §162—Trade or Business Expenses; Cornell Law School: Ithaca, NY, USA, 2025. Available online: https://www.law.cornell.edu/uscode/text/26/162 (accessed on 28 April 2025).
52. Kunik, M. On the Downward Löwenheim–Skolem Theorem for Elementary Submodels. arXiv 2024, arXiv:2406.03860. Available online: https://arxiv.org/abs/2406.03860 https://doi.org/10.48550/arXiv.2406.03860 (accessed on 20 March 2025).
53. Michel, J.-B.; Shen, Y.K.; Aiden, A.P.; Veres, A.; Gray, M.K.; The Google Books Team; Pickett, J.P.; Hoiberg, D.; Clancy, D.; Norvig, P.; et al. Quantitative Analysis of Culture Using Millions of Digitized Books. Science 2011, 331, 176–182. https://doi.org/10.1126/science.1199644
54. Petersen, A.M.; Tenenbaum, J.N.; Havlin, S.; Stanley, H.E.; Perc, M. Statistical Laws Governing Fluctuations in Word Use from Word Birth to Word Death. Sci. Rep. 2012, 2, 313. https://doi.org/10.1038/srep00313
55. Van Dalen, D. Logic and Structure, 4th ed.; Springer: Berlin/Heidelberg, Germany, 2004; Universitext, https://doi.org/10.1007/978-3-540-85108-0
56. Ebbinghaus, H.-D. Löwenheim–Skolem Theorems. In Philosophy of Logic; Gabbay, D.M., Guenthner, F., Eds.; Elsevier: Amsterdam, The Netherlands, 2007; pp. 587–614. https://doi.org/10.1016/B978-044451541-4/50018-X
57. Chang, C.C.; Keisler, H.J. Model Theory, 3rd ed.; Dover: Mineola, NY, USA, 2012.
58. Schlipf, J.S. Formalizing a Logic for Logic Programming. Ann. Math. Artif. Intell. 1992, 5(2–4), 279–302. https://doi.org/10.1007/BF01543479

 doi:10.20944/preprints202507.1895.v4

59. Thomas, W. Languages, automata, and logic. In Handbook of Formal Languages: Volume 3 Beyond Words (pp. 389-455). Berlin, Heidelberg: Springer Berlin Heidelberg. 1997. https://doi.org/10.1007/978-3-642-59126-6

Short Biography of Authors

Phillip G. Bradford is a computer scientist with extensive experience in academia and industry. He is on the faculty at the University of Connecticut. Dr. Bradford was awarded an Honoris Causa Doctor of Engineering, from the University of Engineering and Management, Kolkata, India. He worked for General Electric, BlackRock, Reuters Analytics, and he co-founded a firm. He occasionally consults and often works with early-stage startups. He was on the faculty at the University of Alabama School of Engineering and at Rutgers Business School. Phil was a post-doctoral fellow at the Max-Planck-Institut für Informatik. He earned degrees from Rutgers University, the University of Kansas, and Indiana University. Phil has several best-in-class results.

Henry A. Orphys is an experienced tax professional with a JD degree and CPA license who has worked for over 40 years with nationally recognized law firms, accounting firms, the Internal Revenue Service, and global companies ranging in size from $40 billion to $3 billion. His industry experience includes semiconductors, telecommunications, engineering & construction, financial institutions' technology, and software. He is a member of Tax Executives Institute and the Connecticut Society of CPAs. He served as the Chief Tax Officer of two multi-billion dollar, publicly traded companies and served for over a decade as the managing tax counsel for a large, publicly held technology company. Henry has held leadership positions in several national tax organizations and has been a frequent speaker at tax conferences. Since leaving his last full-time public company position Henry has continued to be active in the fields of tax and tax accounting, having worked as a tax and accounting advisor to various companies. He is also the holder of several patents applicable to the use of Artificial Intelligence to U.S. tax and accounting rules.