

Article

Not peer-reviewed version

A Multimodal AI System: Comparing LLMs and Theorem Proving Systems

[Phillip G. Bradford](#)* and [Henry Orphys](#)

Posted Date: 25 December 2025

doi: 10.20944/preprints202507.1895.v3

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



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a [Creative Commons CC BY 4.0 license](#), which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

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

[†] This paper is an extended version of our paper published in *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 where it has an uncountable number of total meanings for its countable number of atoms, and (2) LLMs can have an uncountable number of word, or 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 the syntax of 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 the text of the user's query with the text of the law to enable the first-order logic theorem-proving system to perform its legal reasoning function. An objective 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 there is 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. The semantics of user input may be different from the semantics of the words the laws were encoded in the theorem-proving system. We give results based on logical model theory to give insight into this issue.

Keywords: first-order logic; theorem-proving; ErgoAI; prolog; LLM; large language model; 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 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 the system and then help users understand the system's output or

answer. This application of LLMs combined with a first-order logical theorem-proving systems is the focus on this paper. An ancillary application of LLMs will be to assist the translation of tax law into ErgoAI code. Both of these applications of LLMs are challenging and have 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. 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 requires either the creation of new words or the assignment of new, additional meanings to existing words. These new word meanings differ from previous meanings of the same word. 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, and (3) 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. Atoms are variables or predicates in a theorem proving system. Particularly, these words and word meanings are 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 applies to both LLMs and 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 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 (i.e., the domain of words contained in the tax law when it was enacted) while handling the new words or word meanings that are tokenized by an LLM. To address this in our analysis we assume an uncountable domain of word or token meanings, a subset of which may be supplied to the theorem-proving system by the LLMs. We then apply Löwenheim–Skolem (L-S) Theorems and logical model theory.

A novel application of the Löwenheim–Skolem (L-S) 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 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. 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. 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. Of course, the Neural Tax Networks system assumes all facts entered by a user are true.

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.

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 or Prolog have at most a countably infinite number of atoms that may be substituted for variables. If each of these atoms has a finite number of meanings, then there is a countable number of total meanings or concepts. [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 with their definitions as they are used here.

Table 1. List of Select Symbols and Their Definitions.

Symbol	Definition
R_t	Root of a proof tree
$G = \langle N, \Sigma, P, S \rangle$	A context-free grammar
\aleph_0	Countable infinite cardinality
\aleph_1	Uncountable infinite cardinality
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}_{llm}	Logical model of LLMs
\mathbb{N}	The set of natural numbers $\{1, 2, \dots\}$
\mathbb{Q}	Field of rational numbers
$\overline{\mathbb{Q}}$	Field of all solutions to polynomial equations with coefficients from \mathbb{Q}
\mathbb{R}	The set of real numbers
\mathbb{C}	The set of complex numbers

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 preparation 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 precise 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].

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 is an expert system that 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 specifically designed for the law. Catala is a programming language designed

specifically for programming law [20]. Catala's focus is on US 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 statutes enacted by legislatures. Statutes are essentially legal rules written in precise natural language. In the USA, 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.

Definition 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 the case of 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.

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.

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 operator \neg . They also have variables x_1, x_2, \dots and quantifiers \forall, \exists . First-order logic quantifiers only apply to variables. Functions and predicates in first-order logic describe relationships. It is the variables, functions, and predicates that differentiates first-order logic from more basic logics. The logical connectives, quantifiers, functions, and predicates may all be applied to the variables. We assume all logic expressions are well-formed, finite, and in first-order logic.

Standard logic proofs are syntactic structures in first-order logic. Such proofs are sometimes called Hilbert-style proofs [43]. First-order logic proofs can be laid out as trees. If the proof's goal is to establish an expression R_t is provable, then the proof tree has R_t as its root. The vertices of a proof tree are facts, axioms, and expressions. The directed 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 SLDC-resolution [38]. To apply such inference rules may require substitution or unification. This is a mix of syntactic and semantic proofs, though ErgoAI can output the syntactic proof.

ErgoAI has frame-based syntax which adds structure and includes object oriented features [3]. The ErgoAI or Prolog expression $H \text{ :- } B$ is a rule. This rule indicates that if the body B is true, then conclude the head H is true.

Modern foundations of some tax laws are not all that different from ancient principles of tax law. Certain current US tax laws are analogous to those from past empires and republics [35–37]. These tax laws in the past were written in different natural languages, but their underlying principles are the same. 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 because modern US 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 tax 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). *A law should be interpreted based on the meaning of its words when the law was enacted.*

Of course, tax 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 variables that are expressed as atoms. The syntax of variables in most programming languages are specified by regular expressions. A regular language is all strings that can be generated by a particular regular expression. A context-free language is all strings can be generated by a particular context-free grammar. Regular languages are context-free languages.

Variables can be grouped together using context-free grammars. This holds true for the syntax of atoms in ErgoAI or Prolog. An atom is an identifier that represents a constant, a string, name or value. Atoms are alphanumeric characters provided the first character is a lower-case alphabetic character or the under-score character. Atoms can name predicates, but the focus here is on atoms representing items that are substituted for variables in first-order expressions.

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 since infix notation is commonly used for accounting and finance. So, the text of all inputs for the tax statutes are a subset of certain context free grammars. A context-free grammar can represent the syntax of parenthesized expressions and atoms in ErgoAI. 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 the natural language of law.

An example of “context beyond originalism the supports the meaning in the natural language of law” is, for federal statutes, the committee reports of the House and Senate committees that drafted, in some cases 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.

Definition 2 (Context-free grammar (CFG)). *A context-free grammar $G = \langle N, \Sigma, P, S \rangle$ 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 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.

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 each element of the set 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.*

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 Cohn showed the independence of the continuum hypothesis from a standard set of axioms for set theory [41,42,44].

The next theorem is classical. In our case, this result is useful for applications of the Löwenheim–Skolem (L-S) Theorems. See Theorem 8.

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 $N = \{S, L\}$, a set of terminals $\Sigma = \{0, 1, 2, \dots, 9\}$, a set of productions P , and the start symbol S . Let $\epsilon \notin \Sigma$, be the empty symbol.

The productions P are:

$$\begin{aligned} S &\Longrightarrow L \mid LS \\ L &\Longrightarrow 0 \mid 1 \mid \dots \mid 9 \\ L &\Longrightarrow \epsilon \end{aligned}$$

So, the CFG G can generate a string corresponding to any natural number in \mathbb{N} . All natural numbers form a countably infinite set, completing the proof. \square

The language of regular expression captures the syntax of an atom in ErgoAI or Prolog. The language of well-formed expressions of atoms in ErgoAI or Prolog are in the language of context-free grammars. The language of regular expressions are a proper subset of the language of context-free grammars. So the cardinality of atoms or expressions of atoms in ErgoAI and Prolog is bounded above by \aleph_0 .

In addition to representing all ErgoAI atoms, the proof of Theorem 1 uses a CFG that generates a language corresponding to all integers in \mathbb{N} . In other words, the language of this CFG contains \aleph_0 strings.

Each individual word in the domain of all words used in the set LAW can be uniquely numbered by an integer 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. To represent positive or negative integers the grammar in the proof of Theorem 1 can be extended by adding two terminals $\{+, -\}$ to Σ , two non-terminals S_0 and T to N . Where S_0 is the new start symbol and the additional rules: $T \Longrightarrow + \mid -$ and $S_0 \Longrightarrow TS \mid 0$ to the set P .

The statement $\vdash f$ indicates that the set of formulas f is syntactically provable in the logical system at hand. That is, using a Hilbert-style proof. Therefore, f is a tautology. A tautology is a formula that is always true. For example, consider a Boolean variable X representing when an purchase is for business. So X is over the domain of $\{\text{true}, \text{false}\}$. Then

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

must always be true so f is a tautology. The formula $\neg f = \neg X \wedge X$ is a contradiction. Contradictions are always false.

Suppose g is a first-order logical 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 quantifier 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 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, \dots
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 =

Scoping can be expressed using a context-free grammar.

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 quantifiers, D is the domain, and σ is the signature of the system which is its constants, predicates, and functions.*

The signature σ of a first-order logic language is the set of all non-logical operators and quantifiers of the language [43]. That is, the signature includes constants, functions, and predicates.

A formula f is an expression of an \mathcal{L} -language and we write $f \in \mathcal{L}$. A set of formulas $\{f_i\}_{i \in \theta}$ of an \mathcal{L} -language is written simply 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 one of \forall or \exists .

Definition 6 (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.*

An interpretation defines a domain and semantics to constants, functions, and predicates for formulas or sentences. An interpretation that makes a formula f true is a model. Given an interpretation, a formula may be neither true nor false if the formula has free variables.

Definition 7 (First-order logic interpretation [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. There is a interpretation domain D_I made from the elements in I
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 I of a first-order logic language $\mathcal{L} = [L, D, \sigma]$ is the domain D_I in addition to the semantics of constants, functions, or predicates. The semantics of interpretations are defined with natural language, mathematics, and mathematical examples. Consider the domain D , signature σ , and an interpretation I . The interpretation I can be applied to a sentence f by substituting values from D_I into $F \in D_I$, where F corresponds to $f \in \mathcal{L}$. These substitutions are done while applying the constants, functions, and predicates as expected based on their semantics.

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 8 (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 for all $\{f\} \subset \mathcal{L}$ are so that each $F(\mathcal{M})$ is true, where each $F \in D_I$ corresponds uniquely with a $f \in \{f\}$.*

Models are denoted \mathcal{M} and \mathcal{N} .

The expression $\models f$ indicates all interpretations make the formula f true. If $\{f\} \subset \mathcal{L}$ is a set of formulas of a first-order \mathcal{L} -language and g is a single formula of \mathcal{L} , then $\{f\} \models g$ holds exactly when all models of $\{f\}$ are models of g . For example, if an interpretation I has the positive integers $\mathbb{N} = \{1, 2, \dots\}$ in its domain for the sentence $f = \forall x \in \mathbb{N}[x^2 > 0]$, then $I \models f$ is true. However, if the domain of I is updated to include $\{0\} \cup \mathbb{N}$, then $I \not\models f$. Going further, if $g = \forall x \in \mathbb{N}[x^2 \geq 0]$, then $f \models g$ but $g \not\models f$.

Definition 9 (First-order logic definitions). *Given a first-order logic formulas $f \in \mathcal{L} = [L, D, \sigma]$ and an interpretation I and f 's corresponding $F \in D_I$, then f is:*

Valid if every interpretation I is so that $F(I)$ is true

Inconsistent or Unsatisfiable if F is false under all interpretations I

Consistent or Satisfiable if F is true under at least one interpretation I

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

The next result is classical for syntactic Hilbert-style proofs, see [42,43].

Theorem 2 (First-order logic is semi-decidable). *Consider a set of first-order logic formulas f .*

1. *If f is true, then there is an algorithm that can verify f 's syntactic 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 syntactic falsity in a finite number of steps.*

Suppose $\{h\}$ is a set of first-order logic formulas. If $\{h\}$ is syntactically provable, then it is valid and hence semantically provable since it satisfies all interpretations. We write $\{h\} \vdash c$ when all interpretations of $\{h\}$ are models of $\{h\}$ as well as models of c .

Theorem 3 (Logical soundness). *For any set of one or more first-order logic formulas $\{h\}$ and a first-order formula c : If $\{h\} \vdash c$, then $\{h\} \models c$.*

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

Theorem 4 (Logical completeness). *For any set of one or more first-order logic formulas $\{h\}$ and a first-order logic formula c : If $\{h\} \models c$, then $\{h\} \vdash c$.*

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 US Federal tax statutes and about 2×10^6 word instances in the US Federal case tax law. Of course most of these words are repeated many times, though they may be in different contexts.

3.1. Interpretation I_{orig} and Model \mathcal{M}_{orig}

This subsection discusses interpretations used by the remainder of the paper. These interpretations are for first-order theorem-proving systems and LLMs. The focus here is on interpretations I_{orig} , assuming originalism, for ErgoAI.

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

A observation based on Theorem 2, 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. This extends to the subset of first-order logic found in ErgoAI or Prolog. This is because SLDC-resolution operates on a subset of first-order logic. This subset is first-order logic Horn-clauses.

The focus here is on ErgoAI as a syntactic theorem prover for first-order logic of Horn-clauses. In this case, all ErgoAI statements map to first-order logic Horn-clause sentences. Any first-order statement that is provable in ErgoAI has a valid model, see Theorem 3. Nonetheless, the Horn-clauses of first-order logic ErgoAI and Prolog work in is still semi-decidable, so Theorem 2 applies. Here we sketch a correspondence between ErgoAI statements and logical interpretations.

The interpretation I_{orig} is for first-order logic programs encoding tax law assuming originalism. In any case, the interpretation I_{orig} fixes all word and phrase meanings from when the laws were enacted, see Assumption 1. All first-order logic expressions using I_{orig} are sentences. This is because languages, such as ErgoAI or Prolog, assume universal quantification for any unbound variables. Of course, the ideas here can apply to many other areas besides tax law.

Unification and negation-as-failure add complexity to first-order logic programming interpretations [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].

3.2. Examples in ErgoAI

This subsection illustrates this section's discussion using ErgoAI.

Listing 2 can be expressed in terms of provability. In this regard, consider Listing 1. The \vdash operator is from the logic of syntactic theorem proving.

Listing 1: Applying a rule or axiom in pseudo-ErgoAI as a proof

```
?X: Expenditure
^ ?X[ordinary -> true]
^ ?X[necessary -> true]
^ ?X[forBusiness -> true]
⊢ ?X: Deduction
```

So, if there is an $?X$ that is an expenditure, and it is ordinary, necessary, and forBusiness, then this is accepted as a proof that $?X$ is a deduction. This is because ErgoAI facts are axioms. ErgoAI rules may be axioms or logical inference rules. So, if the first four lines of the rule in Listing 1 are satisfied, then we have a proof that $?X$ is a Deduction.

Listing 2 is an ErgoAI rule for determining if an expenditure is a deduction. The notation $?X$ is that of a variable. The expression $?X:\text{Class}$ indicates the variable $?X$ is an instance of Class. In this listing, the variable $?X$ also has Boolean properties ordinary, necessary, and forBusiness.

Listing 2: 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 2 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.

The ErgoAI code in Listing 3 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 3: 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 3 is in Listing 4. This listing shows three matches for the variable ?X.

Listing 4: 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 4.

4. Theorem Proving, Logic Models, and LLMs

This section focuses on logical model theory as it applies to LLMs using logic programming. This section wraps up discussing how logic programming can illustrate logical semantics using the \models symbol.

In the case of LLMs, tokens are represented by vectors (embeddings) in high-dimensional space. These word or word fragments are tokens. Each feature of a token has a dimension. In many 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 the most recent jargon to enter their legal questions. The most recent jargon may be different from the semantics of the laws when the laws were enacted. See Assumption 1.

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

Given two embeddings x and y both from the set of all embeddings V and a similarity measure $s : V \times V \rightarrow [-1, +1]$. 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. We assume similarity measures stringent enough so all subsets $V_x \subseteq V$ are finite.

Definition 10 (Extended logical semantics for LLMs). *Consider a first-order logic language $\mathcal{L} = [L, D, \sigma]$ and a set of first-order sentences $\{f\} \subset \mathcal{L}$ with a model \mathcal{M} . Then $\mathcal{M} \models^* \{f\}$ iff all pairs $\{a, b\} \subset \mathcal{M}$ have similarity $s(a, b)$ that is above a suitable threshold.*

In Definition 10, 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 1.

Definition 11 (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,

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

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

4.1. ErgoAI Examples

We use first-order logic capabilities of pseudo-ErgoAI in frame-mode.

If there is an ErgoAI rule or axiom:

$$?X1, \dots, ?Xn : H \quad :- \quad ?X1, \dots, ?Xn : B$$

so that H is the head and B is the body. Where H holds if B is true.

Consider the rule,

$$F(?X) = ?X:liability \quad :- \quad ?X:forBusiness. \quad (1)$$

Along with the model,

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

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

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

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

$$\mathcal{M}' = \{ \text{foodbank:donation}, \text{politicalParty:donation} \},$$

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

$$\mathcal{M} = \{ \text{salad:Expenditure}, \text{burger:Expenditure} \}.$$

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 1. 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, Listing 1 indicates that 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.

As before, our theorem-proving systems prove valid theorems in first-order logic. These results suppose tax law and clarifications are in a set LAW and then translated to a first-order logic suitable for a legal theorem-proving system. We assume the semantics of the law is at most countable from when the laws were enacted. This is assuming originalism for the 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 the new semantics introduced by the LLMs. In traditional logic terms, first-order logic cannot differentiate between the original theorem-proving semantics and the new LLM semantics.

LLMs are trained on many natural language sentences or phrases. Currently, several LLMs are trained on greater than 2×10^{12} words instances or tokens.

Theorem 5 (Special-case of the upward Löwenheim–Skolem [43]). *Consider a first-order set of formulas $\{f\} \subset \mathcal{L} = [L, D, \sigma]$. 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 indicates we must retain the meanings of words 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 do not leave a 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 also 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 non-zero feature values. For example, Figure 1 shows feature values for each row. Each row represents one or more words. Theorem 6 is based on adding new words with new meanings, or adding new meanings to current words over time. 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 row, or (2) a new meaning may require new features. See the feature table in Figure 1. 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 non-zero feature sets so they will be counted once. Similarly, synonyms can be on consecutive rows of a feature table. Homonyms are each individually listed with their different feature sets since homonyms have different meanings. Furthermore, individual words may have many feature sets. Each feature set of a single word represents a different meaning.

The number of meanings is uncountable based on the next argument. Given all of these new words and their new features, they diagonalize. Figure 1 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 non-zero feature values of any of the other words or meanings. So, if there is a countable number of feature columns and a countable number of rows, there must always be additional meanings not listed.

	...	Feature t_j	Feature t_{j+1}	Feature t_{j+2}	Feature t_{j+3}	...
⋮						
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}		$m_{i+2,j}$	$m_{i+2,j+1}$	$m_{i+2,j+2}$	$m_{i+2,j+3}$	
⋮						

Figure 1. A subset of an enumeration of words and their features in a feature table.

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 terms 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.

Recall Assumption 1 which 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 provided in Reading Law: The Interpretation of Legal Texts by Scalia and Garner [40]. Under the section entitled “Semantic Cannons/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 statute 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, unrelated but not contradictory, 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. Salia and Garner, “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 is used. This will determine with an extremely high degree of accuracy what meaning should be assigned to that 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).

Theorem 1 shows context-free grammars can approximate countable infinite domains for theorem proving systems. This is by constructing a countable number of atoms or expressions of atoms. So 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 human 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 12 (Elementary extensions and substructures). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$. Let \mathcal{M} and \mathcal{N} be models 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

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

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

The operator \models^* also can form elementary extensions. That is, by forming elementary model embeddings. For example, consider a sentence $f \in \mathcal{L}$ where f has n arguments, then construct f' where each argument is a pair.

This allows us to use a function g so that if $f(x)$ corresponds to $f((x, g(x)))$ and g can be a similarity measure. So, for example, if \mathcal{N}_{llm} and $\mathcal{M}_{\text{orig}}$ we have Since if $\mathcal{M} \subseteq \mathcal{N}$, then we can define \models^* as

$$\mathcal{N}_{\text{llm}} \models f(x_1, \dots, x_n) \Leftrightarrow \mathcal{M}_{\text{orig}} \models f'((x_1, g(x_1)), \dots, (x_n, g(x_n)))$$

for all first-order sentences in f' corresponds to $f \in \mathcal{L}$ and g is a similarity measure.

Definition 13 (Elementary equivalence). *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*

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

Given a model \mathcal{N} of a first-order language, then $\text{Th}(\mathcal{N})$ is the first-order theory of \mathcal{N} . See Definition 11.

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

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

Suppose a user input U is compatible with first-order logic rules and regulations R of our theorem proving system. These facts and rules are in the set $U[R]$. LLMs may help give the semantic expression G where $U[R] \models^* G$. The formulas G are computed with (e.g., cosine) similarity along with any additional logical rules and facts. This also requires Assumption 2 giving a countable model for G .

The next version of the Löwenheim–Skolem (L-S) Theorems is from [41,55]. See also [52].

Theorem 8 (Löwenheim–Skolem (L-S) Theorems). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ with an infinite model \mathcal{M} . Then there is a model \mathcal{N} so that $|\mathcal{N}| = \kappa$ and*

Upward *If $\kappa \geq |\mathcal{M}|$, then \mathcal{N} is an elementary extension of \mathcal{M} , or $\mathcal{M} \preceq \mathcal{N}$,*

Downward *If $\kappa < |\mathcal{M}|$, then \mathcal{N} is an elementary substructure of \mathcal{M} , or $\mathcal{N} \preceq \mathcal{M}$.*

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

The next corollary applies to tax law as well as other areas.

Corollary 1 (Application of Upward L-S). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ with a countably infinite model $\mathcal{M}_{\text{orig}}$ for a first-order logic theorem proving system. Suppose this first-order theorem-proving system has a countably infinite domain from a countably infinite model $\mathcal{M}_{\text{orig}}$ where $|\mathcal{M}_{\text{orig}}| = \kappa = \aleph_0$. Then there is a model \mathcal{N} that is an elementary extension of $\mathcal{M}_{\text{orig}}$ and $|\mathcal{N}| \geq \aleph_0$.*

Proof. There is a countably infinite number of elements in the domain D_I from an interpretation of \mathcal{L} if these atoms can be specified by a regular expression or a context-free grammar by Theorem 1. So, apply Theorem 8 (upward) to the first-order logic theorem proving system for sentences of \mathcal{L} , with a countably infinite model $\mathcal{M}_{\text{orig}}$. The upward L-S theorem indicates there is a countable infinite model $\mathcal{M}_{\text{orig}}$ so that $|\mathcal{M}_{\text{orig}}| = \kappa = \aleph_0$. \square

For the next result, a precondition is the model $\mathcal{M}_{\text{orig}}$ is a subset of \mathcal{N}_{llm} .

Corollary 2 (Application of Downward L-S). *Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ and a countably infinite model $\mathcal{M}_{\text{orig}} \subseteq \mathcal{N}_{\text{llm}}$, for a first-order logic theorem proving system. We assume an LLM with \models^* and a model \mathcal{N}_{llm} so that $|\mathcal{N}_{\text{llm}}| = \kappa = \aleph_1$. By the downward L-S theorem, $\mathcal{M}_{\text{orig}}$ is an elementary substructure of \mathcal{N}_{llm} and $|\mathcal{M}_{\text{orig}}| = \aleph_0$.*

Proof. Consider a first-order language $\mathcal{L} = [L, D, \sigma]$ and a first-order logic theorem-proving system with a countably infinite model $\mathcal{M}_{\text{orig}}$. The elements or atoms of this model can be defined using context-free grammars or a regular language, see Theorem 1. This generates a countably infinite domain for the model so $|\mathcal{M}_{\text{orig}}| = \aleph_0$.

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

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

Then by the downward Löwenheim–Skolem Theorem, there is an elementary substructure \mathcal{M}_{orig} of \mathcal{N}_{llm} where $|\mathcal{M}_{orig}| < |\mathcal{N}_{llm}|$. This completes the proof. \square

To apply this corollary, suppose the originalism-based legal first-order logic theorem proving system has a model whose elements are a subset of an LLM model. Then there is an equivalence between an LLM's uncountable model and a first-order countable 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.

Corollary 3. Consider a language $\mathcal{L} = [L, D, \sigma]$ and two of its models \mathcal{M}_{orig} and \mathcal{N}_{llm} , where $\mathcal{M}_{orig} \subseteq \mathcal{N}_{llm}$, then

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

5.1. Highlights of Our System Design

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, $LAW \rightarrow R$.

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 US 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. Currently, our proof-of-technology has a basic highly structured user interface that allows a user to enter facts. The proof-of-technology also builds simple queries with the facts. Currently, this is done using our standard user interface. Our system is based on a cloud-based client-server architecture where the proofs are computed by ErgoAI on our backend. Our backend is in the cloud. Figure 2 provides a sketch of the system architecture.

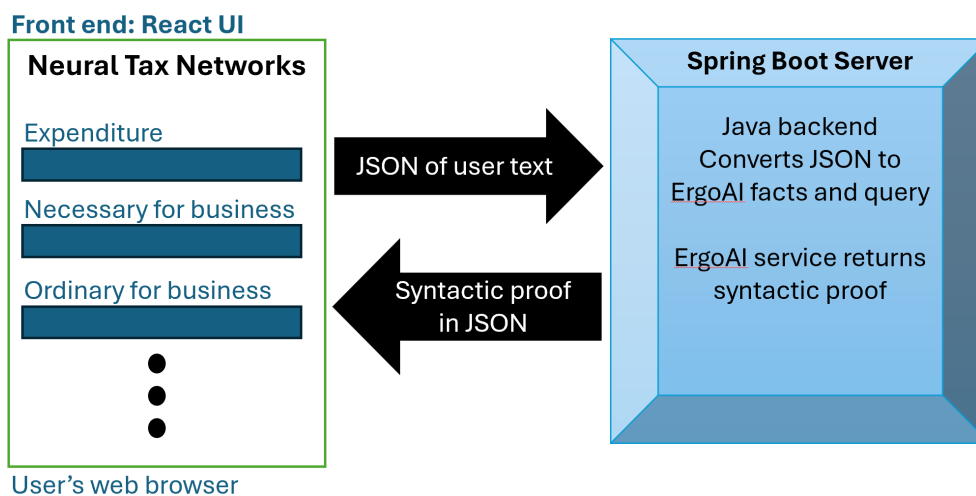


Figure 2. Basic architecture of the Neural Tax network client-server system.

The front end is a React UI system. This React UI has users fill out details of their tax questions. The backend runs Java in a Spring-boot server. 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.

A tax situation is entered by a user through the front end. A tax question is expressed by facts and a query. In response, the proofs are presented on the front end of our system.

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. The user input is the set of natural language expressions U . The set U contains user supplied facts and a tax question. The first-order logic language is in the set R . The set R is based on the set LAW of tax laws and clarifications. The set R was 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 that are compatible with the logic rules and facts R in ErgoAI. We have not yet settled on how to leverage AI to map from user entered natural language tax questions into ErgoAI facts and queries.

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

The expression $U[R] \models^* G$ indicates given $U[R]$, run an LLM to generate G .

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 in our ErgoAI rules R so ErgoAI can work to derive a result.

Definition 14 and Definition 15 can be combined to give the next definition.

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

Figure 3 shows our vision of the Neural Tax Networks system. Currently, we are not doing the LLM mapping automatically.

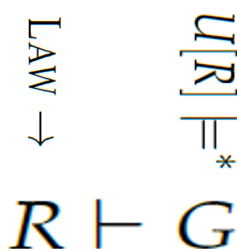


Figure 3. A relationship of syntax and semantics for our multimodal AI system.

Figure 3 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 is placed in a set U . Then U is converted into first-order ErgoAI expressions in a set G which are compatible with R . That is, use an LLM to compute $U[R] \models^* G$. Finally, a theorem-proving system tries to find a proof $R \vdash G$. A set G will be computed for different natural language user input facts and questions U .

The boxes in Figure 4 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 query in U and then the system uses

an LLM to compute $U[R] \models^* G$ giving G . Next, the system attempts to prove $R \vdash G$. This figure highlights knowledge authoring, the semantics of LLMs, and logical deduction.

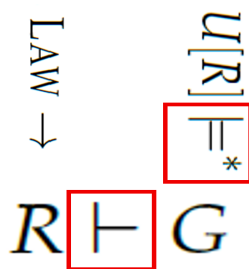


Figure 4. Highlights of the repeated syntactic proofs and semantic LLM models in our multimodal AI system.

We have several example user entered tax situations that we are currently using to test our system. As we add LLMs, we will add testing with similar words and measure how the system performs. These measurements will be based on correct proofs presented on the front end.

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, $\lim_{x \rightarrow \infty} f(x)$, for a mathematical function $f(x)$. In discussing LLMs, our limit is over the infinite time. Figure 5 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.

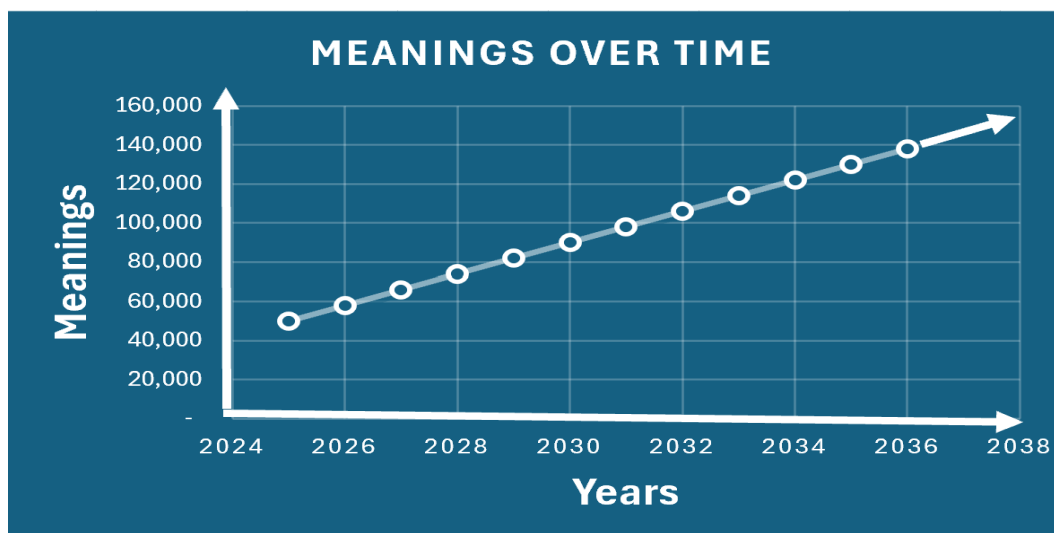


Figure 5. An illustration of the limit of meanings over time.

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 US 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.

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.

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

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

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. 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.

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 NSF 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.

Acknowledgments: 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.

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://isamuwebsite.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](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., Guentner, 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>
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>

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

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, USA. 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.