Automated Proof-Search for Gödel-Löb Provability Logic via Tree Sequents

Omar Y. A. A. Taher

TU Dresden
Technical Report
Supervisor: Dr. Tim S. Lyon

Abstract. This paper introduces an EXPTIME decision procedure for the (in)validity of Gödel-Löb Provability Logic via a syntactic variant of the Tree-Hypersequent System CSGL.

Keywords: Automated Proof Search · Gödel-Löb Provability Logic · Modal Logic · Proof Theory · Tree Sequents .

1 Introduction

Modal Logic is the extension of classical logic that includes modalities, which are operators that capture notions of necessity ($\Box$) and possibility ($\Diamond$). It dates back in modern history to 1912 with the works of C.I. Lewis who proposed the first system. To be more precise, Modal Logic can often times be used to refer to a family of logics that alternates the meaning of the modalities to discuss things like belief (Doxastic Logic), knowledge (Epistemic Logic), time (Temporal Logic), and provability (Provability Logic).The last of which is the one we draw our attention to in this paper. But first, some history.

In 1931, Kurt Gödel made the following two discoveries about systems like Peano's arithmetic (sufficiently powerful formal theories which are sufficiently sound). (1) There are true statements that are undecidable for such systems, i.e., these systems are incomplete and (2) These systems can not prove it's own consistency. These two discoveries are referred to as Gödel's first and second incompleteness theorems. Gödel aimed to capture the provability of mathematical statements in a certain mathematical system (which is usually Peano's arithmetic). He figured out a way to encode these statements as numbers and then created the *Provability Predicate* which was short hand to be able to make Peano's arithmetic speak about its own proofs. It was of the form $\mathsf{Prov}_{PA}(\ulcorner A \urcorner)$ where $\ulcorner A \urcorner$ is a number encoding of a mathematical statement. $\mathsf{Prov}_{PA}(\ulcorner A \urcorner)$ means that A is provable in Peano's arithmetic, i.e., $\mathsf{Prov}_{PA}(\ulcorner A \urcorner) \iff \mathbf{PA} \vdash A$. The heart of Gödel's incompleteness theorems is proving the existence of a statement A such that $\mathbf{PA} \vdash A \iff \neg\mathsf{Prov}_{PA}(\ulcorner A \urcorner)$.

In 1952, Leon Henkin asked the question "What can be said about the sentence asserting its own provability?". Which basically means what can be said about A such that $\mathbf{PA} \vdash A \iff \mathsf{Prov}_{PA}(\ulcorner A \urcorner)$. 3 years later, Martin Löb responded showing that showed $\mathsf{Prov}_{PA}(\ulcorner A \urcorner) \to A$, can be proved in Peano Arithmetic only in the trivial case that Peano Arithmetic already proves A itself. This is formalized as Löb's theorem: $\mathbf{PA} \vdash \mathsf{Prov}_{PA}(\mathsf{Prov}_{PA}(\ulcorner A \urcorner) \to A) \to \mathsf{Prov}_{PA}(A)$.

Provability logics try to capture the provability of mathematical statements in a certain mathematical system (which is usually Peano's arithmetic). It uses propositional variables as stand-ins for formulas of some mathematical system. It then makes statements like $\Box p$ which states that p is a mathematical statement that is provable in some system (going forward we will be using Peano's arithmetic PA), i.e., $\mathbf{PA} \vdash p$. One such logic is Gödel-Löb Provability Logic GL which takes the modal logic **K4** and extends it with Löb's theorem as an axiom where the $\Box$ represents the provability predicate: $\Box(\Box\varphi \to \varphi) \to \Box\varphi$.

Proof theory is the study of the structure and nature of mathematical proofs. In the 1930s, Gerhard Gentzen introduced Sequent systems. In a sequent system, reasoning is expressed as sequents that clearly show what assumptions lead to what conclusions. This formalism makes it easier to achieve things like decidability and to analyze proofs. They can be used to show properties like soundness, completeness, and decidability.

The logic GL enjoys a rich structural proof theory, possessing many cut-free sequent-style systems that share a correspondence between one another such as the Labeled Sequent System $\mathsf{G3GL}$,

the Linear Nested Sequent System LNGL, the Gentzen System $\mathsf{GL}_{\mathsf{seq}}$, and the Tree-Hypersequent System CSGL[4]. The last of which is the system that we built a proof search algorithm using. CSGL, which was provided by Poggiolesi [5], uses so called *tree-hypersequents* in proofs. These tree-hypersequents are trees whose nodes are Gentzen Sequents. In the end of her paper, Poggiolesi states that " It would now be interesting to analyze which other properties this calculus satisfies, such as the decidability property or the interpolation theorem."

This paper picks up where her paper left off. We solved the open decidability question posited at the end of her paper by showing her system decidable.The contributions of the paper can be taken as the following: (1) We created a proof search algorithm that is correct and terminating in EXPTIME, (2) This algorithm also supports counter model extraction using the notion of something called a saturated sequent which we go over in the paper, and (3) we show that something called the diagonal formula which usually makes cut elimination in systems like CSGL difficult to be the reason why our algorithm can terminate.

Outline of Paper. In Section 2, we go over the logical preliminaries needed to discuss this project. They include the syntax of the language, the semantics, and the axioms and inference rules of a sound and complete Hilbert proof system for Gödel-Löb logic, In Section 3, we discuss Poggiolesi's tree sequents dressed up in more readable notation to adequetly introduce her system for for GL, CSGL [5]. Then we go over some important properties and characteristics that CSGL has. In Section 4, we finally introduce the proof search algorithm for CSGL that we devised and discuss results such as correctness and termination. Termination being one of the big contributions of this paper. In Section 5, we discuss the complexity and whether or not our algorithm is optimal. Last, in Section 6, we conclude and discuss future work.

2 Logical Preliminaries

In this section we introduce the syntax and semantics of our language as well as helpful terminology like length of a formula and modal depth of a formula. Then, we introduce a sound and complete Hilbert proof system for GL. Noting the system's axioms and inference rules.

We let $\mathsf{Prop} := \{p, q, r, \ldots\}$ be a countable set of *propositional atoms* and define the language $\mathcal{L}$ to be the set of all formulae generated via the following grammar in BNF:

$$\varphi ::= p \mid \bot \mid \varphi \to \varphi \mid \Box\varphi$$

where p ranges over Prop. We use φ, ψ, χ, ... to denote formulae in $\mathcal{L}$ and define $\neg\varphi := \varphi \to \bot$, $\varphi \wedge \psi := \neg(\varphi \to \neg\psi)$, and $\varphi \vee \psi := \neg\varphi \to \psi$ as usual.

Definition 1 (Length of a Formula). *We define the* length *of a formula as the number of symbols used to write the formula. The symbols are:*

- *Propositional Atoms:* $\{p, q, r, \ldots\}$
- *Logical Connectives:* $\{\to, \Box, \bot\}$
- *Parentheses:* $\{(,)\}$

Let $len(\varphi)$ *be a function that takes a formula* φ *of the language* $\mathcal{L}$ *and returns a natural number denoting the formula's length. It is defined inductively as follows:*

$$len(\varphi) = \begin{cases} 1 & \text{if } \varphi = p, \text{ where } p \in \mathsf{Prop}, \\ 1 & \text{if } \varphi = \bot, \\ len(\varphi) = 1 + len(\varphi_1) + len(\varphi_2) & \text{if } \varphi = \varphi_1 \to \varphi_2, \\ len(\psi) + 1 & \text{if } \varphi = \Box\psi. \end{cases}$$

The parentheses have been ignored in this definition.

Definition 2 (Modal Depth of a Formula). *We define the* modal depth *of a formula as the measure of the level of nestings of the modal operator* $\Box$ *in a formula* φ*, i.e., how far a formula can "see." Let* $depth(\varphi)$ *be a function that takes a formula* φ *of the language* $\mathcal{L}$ *and returns a natural number denoting the formula's depth. It is defined inductively as follows:*

$$depth(\varphi) = \begin{cases} 0 & \text{if } \varphi = p, \text{ where } p \in \mathsf{Prop}, \\ 0 & \text{if } \varphi = \bot, \\ max(depth(\varphi_1), depth(\varphi_2)) & \text{if } \varphi = \varphi_1 \to \varphi_2, \\ 1 + depth(\psi) & \text{if } \varphi = \Box\psi. \end{cases}$$

Definition 3 (Subformulae of a Formula). *We define a* subformula *of a formula as any substring of that formula's symbols that is itself a valid formula in $\mathcal{L}$. For example: $\varphi \to \psi$ is a subformula of $(\varphi \to \psi) \to \omega$ but $\varphi \to$ is not. We define the* set *of subformulae of a formula as all the subformulae of that formula. We introduce $sub(\varphi)$ as a function that takes a formula φ as input and returns the set of all its subformulae. It is defined inductively as follows:*

$$sub(\varphi) = \begin{cases} \{p\} & \text{if } \varphi = p, \text{ where } p \in \mathsf{Prop}, \\ \{\bot\} & \text{if } \varphi = \bot, \\ \{\varphi_1 \to \varphi_2\} \cup sub(\varphi_1) \cup sub(\varphi_2) & \text{if } \varphi = \varphi_1 \to \varphi_2, \\ \{\Box\psi\} \cup sub(\psi) & \text{if } \varphi = \Box\psi. \end{cases}$$

Definition 4 (Model). *We define a* model *to be a tuple $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ such that*

- *$\mathcal{W}$ is a non-empty set of worlds w, u, v, ... (occasionally annotated);*
- *$\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is transitive and conversely-wellfounded;*[1]
- *$\mathcal{V} : \mathsf{Prop} \to 2^{\mathcal{W}}$ is a* valuation function.

Definition 5 (Semantic Clauses). *We define the* satisfaction *of a formula φ in a model $\mathcal{M}$ at world w, written $\mathcal{M}, w \models \varphi$, recursively as follows:*

- *$\mathcal{M}, w \models p$ iff $w \in \mathcal{V}(p)$;*
- *$\mathcal{M}, w \not\models \bot$;*
- *$\mathcal{M}, w \models \varphi \to \psi$ iff $\mathcal{M}, w \not\models \varphi$ or $\mathcal{M}, w \models \psi$;*
- *$\mathcal{M}, w \models \Box\varphi$ iff $\forall u \in \mathcal{W}$, if $(w, u) \in \mathcal{R}$, then $\mathcal{M}, u \models \varphi$;*
- *$\mathcal{M} \models \varphi$ iff $\forall w \in \mathcal{W}$, $\mathcal{M}, w \models \varphi$.*

We write $\models \varphi$ and say that φ is valid *iff for all models $\mathcal{M}$, $\mathcal{M} \models \varphi$.* Gödel-Löb logic (GL) *is defined to be the set $\mathsf{GL} \subset \mathcal{L}$ of all valid formulae.*

A Hilbert-style axiomatization of GL can be obtained by adding the K axiom and Löb's axiom $\Box(\Box\varphi \to \varphi) \to \Box\varphi$ Church's axiom system and the necessitation rule to it's inference rules (cf. Segerberg [6]). According to Boolos [1], this gives a system that is sound and complete for GL.

Axioms

$$\text{(A1)} = \varphi \to (\psi \to \varphi) \quad \text{(A2)} = (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$$

$$\text{(A3)} = (((\varphi \to \bot) \to \bot) \to \varphi) \quad \text{(K)} = \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi) \quad \text{(L)} = \Box(\Box\varphi \to \varphi) \to \Box\varphi$$

Inference Rules

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}\ \text{(mp)} \qquad\qquad \frac{\varphi}{\Box\varphi}\ \text{(n)}$$

3 Tree Sequents

We now introduce *tree sequents*, which are essentially nested sequents [2, 3] (sometimes called *tree-hypersequents*) 'dressed' in labeled notation (cf. [7]).[2] We introduce the syntax using a lot of helpful terminology to be able adequately describe the sequents and their components. Then we introduce the notions of subformulae and modal depth for tree sequent. Then we introduce the semantics of tree sequents.

Definition 6 (Gentzen Sequent). A *sequent* is an expression of the form $\Gamma \vdash \Delta$, *where the antecedent Γ and the succedent Δ are multisets of formulae.*

[1] We note that $\mathcal{R}$ is conversely-wellfounded *iff* it does not contain any infinite ascending $\mathcal{R}$-chains.
[2] We remark that tree-hypersequents are notational variants of nested sequents.

3.1 Labels, Relational Atoms, Labeled Formulas, and Other Notations

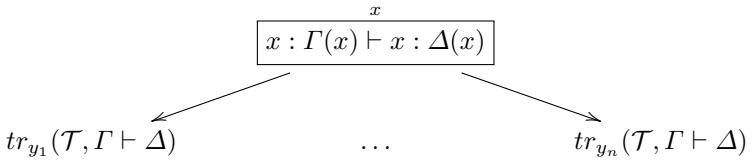
- **Labels:** Let $Lab = \{x, y, z, \ldots\}$ be a countably infinite set of *labels*.
- **Relational Atoms:** We define a *relational atom* to be an expression of the form xRy with $x, y \in Lab$.
- **Labeled Formula:** We define a *labeled formula* to be an expression of the form $x : \varphi$ such that $x \in Lab$ and $\varphi \in \mathcal{L}$.
- As shown in the previous definition, we use upper-case Greek letters $\Gamma, \Delta, \Sigma, \ldots$ to denote finite multisets of labeled formulae.
- **Lab($\cdot$):** For a set $\mathcal{R}$ of relational atoms and a multiset Γ of labeled formulae, we let $Lab(\mathcal{R})$, $Lab(\Gamma)$, and $Lab(\mathcal{R}, \Gamma)$ be the sets of all labels occurring therein.
- **Set Operations:**
  1. $\Gamma(x)$**:** We define the multiset $\Gamma(x) := \{\varphi \mid x : \varphi \in \Gamma\}$.
  2. if $\Gamma = \varphi_1, \ldots, \varphi_n$, then $x : \Gamma := x : \varphi_1, \ldots, x : \varphi_n$.
  3. Γ, Δ denotes the multiset union of the two multisets Γ and Δ.
- **Tree:** A set $\mathcal{T}$ of relational atoms is a *tree iff* the graph $G(\mathcal{T}) = (V, E)$ forms a tree, where $V = \{x \mid x \in Lab(\mathcal{T})\}$ and $E = \{(x, y) \mid xRy \in \mathcal{T}\}$.

Definition 7 (Tree Sequent).
A tree sequent *is defined to be an expression of the form $\mathcal{T}, \Gamma \vdash \Delta$ such that:*

1. *$\mathcal{T}$ is a tree,*
2. *if $\mathcal{T} \neq \emptyset$, then $Lab(\Gamma, \Delta) \subseteq Lab(\mathcal{T})$, and*
3. *if $\mathcal{T} = \emptyset$, then $|Lab(\Gamma, \Delta)| = 1$, i.e. all labeled formulae in Γ, Δ share the same label.*

These conditions ensure that each tree sequent forms a connected graph that is indeed of a tree shape. We use T and annotated versions thereof to denote tree sequents. The root *of a tree sequent $\mathcal{T}, \Gamma \vdash \Delta$ is the label x such that there exists a unique directed path of relational atoms in $\mathcal{T}$ from x to every other label $y \in Lab(\mathcal{T}, \Gamma, \Delta)$; if $\mathcal{T} = \emptyset$, then the root is the single label x shared by all formulae in Γ, Δ.*

Every tree sequent encodes a tree whose vertices are Gentzen sequents. In other words, each tree sequent $T = \mathcal{T}, xRy_1, \ldots, xRy_n, \Gamma \vdash \Delta$ such that x is the root and $y_1, \ldots, y_n$ are all children of x can be graphically depicted as a tree $tr_x(T)$ of the form shown below:



Definition 8 (Subformulae of a Tree Sequent). *We define* the set of subformulae *of a tree sequent $\mathcal{T}, \Gamma \vdash \Delta$ as the union of the sets of subformulae of each labeled formula occurring in the tree sequent. More formally:*

$$sub(\mathcal{T}, \Gamma \vdash \Delta) := \bigcup_{x:\varphi \in \Gamma, \Delta} sub(\varphi)$$

Definition 9 (Modal Depth of a Tree Sequent). *We define* modal depth *of a tree sequent $\mathcal{T}, \Gamma \vdash \Delta$ as the maximum depth among the modal depths of all labeled formulae occurring in the tree sequent More formally:*

$$depth(\mathcal{T}, \Gamma \vdash \Delta) = \max\{depth(\varphi) \mid x : \varphi \in \Gamma, \Delta\}$$

Below, we clarify the interpretation of tree sequents by explaining their evaluation over models.

Definition 10 (Tree Sequent Semantics). *Let $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ be a model. We define an* $\mathcal{M}$-assignment *to be a function $\mu\colon Lab \to \mathcal{W}$.*

Satisfiability

- *A relational atom xRy is* satisfied *on $\mathcal{M}$ with $\mathcal{M}$-assignment μ iff $(\mu(x), \mu(y)) \in \mathcal{R}$*
- *A tree sequent $\mathcal{T}, \Gamma \vdash \Delta$ is* satisfied *on $\mathcal{M}$ with $\mathcal{M}$-assignment μ written $\mathcal{M}, \mu \models \mathcal{T}, \Gamma \vdash \Delta$ iff* ***if***
  - *every $xRy \in \mathcal{T}$ is* satisfied *on $\mathcal{M}$ with $\mathcal{M}$-assignment μ and*
  - *for all $x : \varphi \in \Gamma$, $\mathcal{M}, \mu(x) \models \varphi$*

  ***then** there exists a $y : \psi \in \Delta$ such that $\mathcal{M}, \mu(y) \models \psi$.*

Validity

A labeled sequent is defined to be valid *iff it is satisfied on all models $\mathcal{M}$ with all $\mathcal{M}$-assignments; a labeled sequent is defined to be* invalid *otherwise.*

3.2 The System CSGL for GL

We now introduce Poggiolesi's tree-hypersequent system for GL [5], but adapted to our signature and using the notation of tree sequents as in Lyon [4]. Then we introduce properties of the system that gives us a lot of useful results when we designed our algorithm. The system consists of the following rules:

$$\frac{}{\mathcal{T}, \Gamma, x : p \vdash x : p, \Delta}\ \mathsf{id}_1 \qquad \frac{}{\mathcal{T}, \Gamma, x : \Box\varphi \vdash x : \Box\varphi, \Delta}\ \mathsf{id}_2 \qquad \frac{}{\mathcal{T}, \Gamma, x : \bot \vdash \Delta}\ \bot\mathsf{L}$$

$$\frac{\mathcal{T}, \Gamma, x : \psi \vdash \Delta \qquad \mathcal{T}, \Gamma \vdash x : \varphi, \Delta}{\mathcal{T}, \Gamma, x : \varphi \to \psi \vdash \Delta}\ \to\mathsf{L} \qquad \frac{\mathcal{T}, \Gamma, x : \varphi \vdash x : \psi, \Delta}{\mathcal{T}, \Gamma \vdash x : \varphi \to \psi, \Delta}\ \to\mathsf{R}$$

$$\frac{\mathcal{T}, xRy, \Gamma, x : \Box\varphi, y : \varphi \vdash \Delta}{\mathcal{T}, xRy, \Gamma, x : \Box\varphi \vdash \Delta}\ \Box\mathsf{L} \qquad \frac{\mathcal{T}, xRy, \Gamma, y : \Box\varphi \vdash y : \varphi, \Delta}{\mathcal{T}, \Gamma \vdash x : \Box\varphi, \Delta}\ \Box\mathsf{R}^{\dagger} \qquad \frac{\mathcal{T}, xRy, \Gamma, x : \Box\varphi, y : \Box\varphi \vdash \Delta}{\mathcal{T}, xRy, \Gamma, x : \Box\varphi \vdash \Delta}\ \mathsf{4L}$$

Fig. 1. Tree Sequent Calculus CSGL for GL. The $\Box$R rule is subject to a side condition †, namely, the rule is applicable only if the label y is fresh.

We will now explain inference rules and how they work as well as their classification. Then we will explain how multiple application of these rules form a derivation which can sometimes be considered a proof. Then, we will introduce some useful properties the system has that provide us with interesting results.

Inference Rules: An **inference rule** (in the context of our discussion) has one of the forms:

$$\frac{}{\mathcal{T}, \Gamma \vdash \Delta}\ \mathsf{R}_1 \qquad \frac{\mathcal{T}', \Gamma' \vdash \Delta'}{\mathcal{T}, \Gamma \vdash \Delta}\ \mathsf{R}_2 \qquad \frac{\mathcal{T}, \Gamma_1 \vdash \Delta_1 \qquad \mathcal{T}, \Gamma_2 \vdash \Delta_2}{\mathcal{T}, \Gamma_3 \vdash \Delta_3}\ \mathsf{R}_3$$

Where every tree sequent over the line is called a *premise* and the tree sequent under the line is called the *conclusion* and rules of the form R_1, R_2, and R_3 are called *initial rules*, *unary rules*, and *binary rules*, respectively. It means "If all the premise(s) hold (i.e., are derivable), then the conclusion is also derivable. "It is also worth noting that initial rules give the *axioms* of the system.

Additional Terminology

We use the $\to\mathsf{L}$ rule as an example to introduce the following terminology:

$$\frac{\mathcal{T}, \Gamma, x : \psi \vdash \Delta \qquad \mathcal{T}, \Gamma \vdash x : \varphi, \Delta}{\mathcal{T}, \Gamma, x : \varphi \to \psi \vdash \Delta} \to \mathsf{L}$$

- **Principal Formula:** The formula introduced in the conclusion.
- **Auxiliary Formula(e):** The formula(e) mentioned in the premise(s).
- **Side Formula(e):** Those in Γ and Δ.

For the $\Box\mathsf{R}$ rule we also introduce the notion of a diagonal formula:

$$\frac{\mathcal{T}, xRy, \Gamma, y : \Box\varphi \vdash y : \varphi, \Delta}{\mathcal{T}, \Gamma \vdash x : \Box\varphi, \Delta} \Box\mathsf{R}$$

As mentioned by Poggiolesi [5], the existence of this formula is what makes doing cut elimination in out system quite difficult.

Types of Rules: The rules id_1, id_2, $\bot\mathsf{L}$ are referred to as *initial rules*. The conclusion of an initial rule is called an *initial sequent*. The remaining rules in CSGL are left and right *logical rules*. An example on how to read one of the names would be "Box Left" for $\Box\mathsf{L}$. Note that some rules can be subject to additional constraints. The $\Box\mathsf{R}$, for instance, has the additional constraint of the label y having to be fresh in any application of the rule, i.e., the label y is forbidden to occur in the conclusion. Another important categorization is whether a rule is modal or local.

Definition 11 (Local Rules). Local rules *are rules that operate on the labeled formulae whose main connective is not a modality. They operate on just propositional formulae that hold within a specific world, hence the name* local. *They do not interact with the modal structure encoded in* $\mathcal{T}$. id_1, id_2, $\bot\mathsf{L}$, $\to\mathsf{L}$, *and* $\to\mathsf{R}$ *are all local rules.*

Definition 12 (Modal Rules). Modal rules *are rules that operate on the labeled formulae of the form* $x : \Box\varphi$. *They operate on just modal formulae, hence the name* modal. *They directly interact with the modal structure encoded in* $\mathcal{T}$ *and sometimes even add things to it.* $\Box\mathsf{L}$, $\Box\mathsf{R}$, *and* $\mathsf{4L}$ *are all modal rules.*

Derivations: A *derivation* is a tree such that

1. Every node is a tree sequent;
2. Evert parent node is the conclusion of a rule with its children are the premises.

The *height of a derivation* is the maximum distance between the root (conclusion) of the derivation and one of its leaves.

Definition 13 (Branch of a Derivation). *Let* $\mathfrak{D}$ *be a derivation. A branch* $\mathfrak{B}$ *is a possibly infinite sequence of nodes* $\mathfrak{B} = (\mathfrak{T}_0, \ldots, \mathfrak{T}_n)$ *such that:*

1. $\mathfrak{T}_0$ *is the root of* $\mathfrak{D}$ *(i.e., the conclusion of the derivation).*
2. *For each* $0 < i < n$, $\mathfrak{T}_{i+1}$ *is a premise of* $\mathfrak{T}_i$. *This means that there exists an inference rule* R *where* $\mathfrak{T}_i$ *is the conclusion and* $\mathfrak{T}_{i+1}$ *is (one of) the premise(s).*
3. $\mathfrak{T}_n$ *is a leaf. This means that it is the consequence of an initial rule.*

Here's an example of a branch $\mathfrak{B} = (\mathcal{T}, \Gamma_0 \vdash \Delta_0, \mathcal{T}, \Gamma_1 \vdash \Delta_1, \mathcal{T}, \Gamma_3 \vdash \Delta_3, \mathcal{T}, \Gamma_7 \vdash \Delta_7, \mathcal{T}, \Gamma_{10} \vdash \Delta_{10})$ *visualized:*

$$\dfrac{\dfrac{\mathcal{T}, \Gamma_4 \vdash \Delta_4 \qquad \dfrac{\mathcal{T}, \Gamma_8 \vdash \Delta_8 \qquad \mathcal{T}, \Gamma_9 \vdash \Delta_9}{\mathcal{T}, \Gamma_5 \vdash \Delta_5} R_5}{\mathcal{T}, \Gamma_2 \vdash \Delta_2} R_3 \qquad \dfrac{\mathcal{T}, \Gamma_6 \vdash \Delta_6 \qquad \dfrac{\mathcal{T}, \Gamma_{10} \vdash \Delta_{10}}{\mathcal{T}, \Gamma_7 \vdash \Delta_7} R_6}{\mathcal{T}, \Gamma_3 \vdash \Delta_3} R_4}{\dfrac{\mathcal{T}, \Gamma_1 \vdash \Delta_1}{\mathcal{T}, \Gamma_0 \vdash \Delta_0} R_1} R_2$$

Proofs: A *proof* is a finite derivation such that the leaves are initial sequents. A sequent is provable in a sequent calculus if there exists a proof of that sequent using the rules of the calculus.

The following is an example of a proof of Löb's axiom using the tree sequent system CSGL for GL (best read bottom up).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{xRy, x : \Box(\Box p \to p), y : \Box p \vdash y : p, y : \Box p}\,\mathsf{id}_2 \quad \dfrac{}{xRy, x : \Box(\Box p \to p), y : \Box p, y : p \vdash y : p}\,\mathsf{id}_1}{xRy, x : \Box(\Box p \to p), y : \Box p, y : \Box p \to p \vdash y : p}\,{\to}\mathsf{L}}{xRy, x : \Box(\Box p \to p), y : \Box p \vdash y : p}\,\Box\mathsf{L}}{x : \Box(\Box p \to p) \vdash x : \Box p}\,\Box\mathsf{R}}{\vdash x : \Box(\Box p \to p) \to \Box p}\,{\to}\mathsf{R}$$

The tree sequent $\vdash x : \Box(\Box p \to p) \to \Box p$ is provable in CSGL.

Theorem 1 ([5]). *The tree sequent calculus* CSGL *satisfies the following:*

(1) *Each tree sequent of the form* $\mathcal{T}, \Gamma, x : \varphi \vdash x : \varphi, \Delta$ *is provable in* CSGL*;*
(2) φ *is valid iff* $\vdash x : \varphi$ *is provable in* CSGL*.*

(Hp-)Admissibility and (Hp-)Invertibility

1. A rule R is *admissible*, *iff* if all premises are provable, then so is the conclusion.
2. A rule R is *invertible*, *iff* if its conclusion is provable, then so are all corresponding premises.
3. For a rule R to be *(height preserving) admissible*, it has to be admissible and the proof of the conclusion has to be of height less than or equal to maximum height of the premises' proofs.
4. For a rule R to be *(height preserving) invertible*, it has to be invertible and the proof of the premise(s) has to be of height less than or equal to that of the original proof of the conclusion.

All non-initial rules in CSGL are (hp-)invertible. The following structural rules are (hp-)admissible in CSGL with the exception of cut which is only admissible.

$$\dfrac{\mathcal{T}, \Gamma \vdash \Delta}{\mathcal{T}, \mathcal{T}', \Gamma, \Gamma' \vdash \Delta, \Delta'}\,\mathsf{w} \qquad \dfrac{\mathcal{T}, \Gamma, x : \varphi, x : \varphi \vdash \Delta}{\mathcal{T}, \Gamma, x : \varphi \vdash \Delta}\,\mathsf{cL} \qquad \dfrac{\mathcal{T}, \Gamma \vdash x : \varphi, x : \varphi, \Delta}{\mathcal{T}, \Gamma \vdash x : \varphi, \Delta}\,\mathsf{cR}$$

$$\dfrac{\mathcal{T}, \Gamma \vdash x : \varphi, \Delta \qquad \mathcal{T}, \Gamma, x : \varphi \vdash \Delta}{\mathcal{T}, \Gamma \vdash \Delta}\,\mathsf{cut}$$

Fig. 2. Admissible rules.

Theorem 2 ([5]). *The tree sequent calculus* CSGL *satisfies the following:*

(1) *All non-initial rules are hp-invertible in* CSGL*;*
(2) *The* w*,* cL*, and* cR *rules are hp-admissible in* CSGL*;*
(3) *The* cut *rule is admissible in* CSGL*.*

These results will be used later in our proofs. Specifically the fact that invertibility allows us to make the claim that if the premise is invalid then so is the conclusion and vice-versa. This will be really helpful when it comes time to use our algorithm to extract a counter model. Having contraction as a hp-admissible rule is also helpful as it allows us to take liberties designing the algorithm in the sense that we don't need to simulate the inference rules a hundred percent accurately (this will be explained in detail in Lem. 2). These properties make for a structurally rich proof system that is easy to be automated.

4 Automated Proof-Search

In this section, we finally make our contribution by introducing an algorithm to do the proof search for the system CSGL. The idea is to simply write an algorithm that encodes the rules systematically. The algorithm checks if we can pattern match a conclusion of one of our rules and then applies it backwards by adding the premises. But we cannot simply write an algorithm that applies rules bottom up. Knowing when to stop applying the rules is as important as knowing when to apply them. But when do we stop applying our rules? To answer that question we introduce the notion of a Saturated Sequent.

Definition 14 (Saturated Sequent). *Let $T = \mathcal{T}, \Gamma \vdash \Delta$ be a tree sequent. We define T to be saturated iff T satisfies the following:*

(id) *if $x : \varphi \in \Gamma$, then $x : \varphi \notin \Delta$;*
($\bot$L) *$x : \bot \notin \Gamma$;*
($\to$L) *if $x : \varphi \to \psi \in \Gamma$, then either $x : \psi \in \Gamma$ or $x : \varphi \in \Delta$;*
($\to$R) *if $x : \varphi \to \psi \in \Delta$, then $x : \varphi \in \Gamma$ and $x : \psi \in \Delta$;*
($\Box$L) *if $xRy \in \mathcal{T}$ and $x : \Box\varphi \in \Gamma$, then $y : \Box\varphi, y : \varphi \in \Gamma$;*
($\Box$R) *if $x : \Box\varphi \in \Delta$, then $\exists y \in Lab(T), xRy \in \mathcal{T}, y : \Box\varphi \in \Gamma$, and $y : \varphi \in \Delta$.*

Intuitively, this means that you will not gain any new information by bottom-up applying rules to the sequent.

With the notion of a saturated sequent, we describe a sequent which will not give us any new additional information when having rules applied to it. This means that we should stop. But what else does the saturation tell us? It actually tells us that not only should we stop applying rules but we have effectively reached a termination condition. Specifically we have reached the false termination condition. Since we have all the information we are going to have and none of it is of the form of an initial rule, that tells us that this sequent is not satisfiable. In fact, given that we have hp-invertibility, the saturated sequent could be used to build a counter model for the sequent.

Lemma 1. *Let $T = \mathcal{T}, \Gamma \vdash \Delta$ be a tree sequent. If T is saturated, then it can be transformed into a model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ with $\mathcal{M}$-assignment μ such that $\mathcal{M}, \mu \not\models T$.*

Proof. We define $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ such that:

- $\mathcal{W} = Lab(T)$
- $\mathcal{R} = \{(x, y) \mid xRy \in \mathcal{T}\}^+$
- $\mathcal{V}(p) = \{x \mid x : p \in \Gamma\}$

We show that $\mathcal{M}$ is indeed a model.

1. Transitive: this is trivial since $\mathcal{R}$ is the transitive closure of all edges, and
2. Conversely Well-Founded. This follows from the fact that $\mathcal{T}$ is a finite tree and $\mathcal{R}$ is the transitive closure of all edges which cannot introduce cycles or obvious infinite paths.

Define $\mu : Lab \to \mathcal{W}$ such that $\mu(x) = x$ if $x \in Lab(T)$ and $\mu(x)$ is arbitrary otherwise. Let $xRy \in \mathcal{T}$. By definition of $\mathcal{R}$, $(x, y) \in \mathcal{R}$. By definition of μ, $\mu(x) = x$ and $\mu(y) = y$. So $(\mu(x), \mu(y)) \in \mathcal{R}$. Using mutual induction on the length of φ and ψ, we show that (1) if $x : \varphi \in \Gamma$, then $\mathcal{M}, \mu(x) \models \varphi$ and (2) if $y : \psi \in \Delta$, then $\mathcal{M}, \mu(y) \not\models \psi$.

Base Cases:

- $x : p \in \Gamma$. Then, $x \in \mathcal{V}(p)$, so $\mathcal{M}, \mu(x) \models p$.
- $y : p \in \Delta$. By id_1 in Def. 14, we have $y : p \notin \Gamma$. By definition of $\mathcal{V}$, $y \notin \mathcal{V}(p)$, so $\mathcal{M}, \mu(y) \not\models p$.
- $x : \bot \in \Gamma$. Trivial because of $\bot$L.
- $y : \bot \in \Delta$. Trivial, $\mathcal{M}, \mu(y) \not\models \bot$.

Inductive Steps:

Algorithm 1.1: `prove`

Input: A tree sequent $\mathcal{T}, \Gamma \vdash \Delta$
Output: A Boolean: `true`, `false`

if $x : \varphi \in \Gamma \cap \Delta$ **then**
  return `true`;
end
if $x : \bot \in \Gamma$ **then**
  return `true`;
end
if $\mathcal{T}, \Gamma \vdash \Delta$ *is saturated* **then**
  return `false`;
end
if $x : \varphi \to \psi \in \Gamma, x : \psi \notin \Gamma, x : \varphi \notin \Delta$ **then**
  Set $\Gamma' := \Gamma, x : \psi$;
  Set $\Delta' := \Delta, x : \varphi$;
  return `prove`$(\mathcal{T}, \Gamma' \vdash \Delta)$ `AND` `prove`$(\mathcal{T}, \Gamma \vdash \Delta')$;
end
if $x : \varphi \to \psi \in \Delta$ *and either* $x : \varphi \notin \Gamma$ *or* $x : \psi \notin \Delta$ **then**
  Set $\Gamma' := \Gamma, x : \varphi$;
  Set $\Delta' := \Delta, x : \psi$;
  return `prove`$(\mathcal{T}, \Gamma' \vdash \Delta')$;
end
if $x : \Box\varphi \in \Gamma$ *and* $xRy \in \mathcal{T}$ *but not* $y : \Box\varphi \in \Gamma$ *and* $y : \varphi \in \Gamma$ **then**
  Set $\Gamma' := \Gamma, y : \Box\varphi, y : \varphi$;
  return `prove`$(\mathcal{T}, \Gamma' \vdash \Delta)$;
end
if $x : \Box\varphi \in \Delta$ *but no label* y *such that* $xRy \in \mathcal{T}$, $y : \Box\varphi \in \Gamma$ *and* $y : \varphi \in \Delta$ **then**
  Let $y \in Lab/Lab(\mathcal{T}, \Gamma \vdash \Delta)$
  Set $\mathcal{T}' := \mathcal{T}, xRy$;
  Set $\Gamma' := \Gamma, y : \Box\varphi$;
  Set $\Delta' := \Delta, y : \varphi$;
  return `prove`$(\mathcal{T}', \Gamma' \vdash \Delta')$;
end

- $x : \varphi \to \psi \in \Gamma$. Since T is saturated, then by $\to$L in Def. 14, either $x : \varphi \in \Delta$ or $x : \psi \in \Gamma$. By IH, $\mathcal{M}, \mu(x) \models \psi$ or $\mathcal{M}, \mu(x) \not\models \varphi$. By the semantics of the implication, we have $\mathcal{M}, \mu(x) \models \varphi \to \psi$ regardless of which case is true.
- $y : \varphi \to \psi \in \Delta$. Since T is saturated, then by $\to$R in Def. 14, $y : \varphi \in \Gamma$ and $y : \psi \in \Delta$. By IH, $\mathcal{M}, \mu(y) \models \varphi$ and $\mathcal{M}, \mu(y) \not\models \psi$. By the semantics of the implication, we have $\mathcal{M}, \mu(y) \not\models \varphi \to \psi$.
- $x : \Box\varphi \in \Gamma$. Assume that $y \in \mathcal{W}$ and $(x, y) \in \mathcal{R}$. Note that due to the definition of $\mathcal{R}$ the existence of $(x, y) \in \mathcal{R}$ does not imply the existence of an immediate relational atom xRy but the existence of $z_1, \ldots, z_n \in Lab(T)$ such that $xRz_1, \ldots, z_nRy \in \mathcal{T}$. Since T is saturated, then by ($\Box$L) in Def. 14, we know that $\Box\varphi$ occurs at each label along the chain $xRz_1, \ldots, z_nRy$, and thus $y : \Box\varphi, y : \varphi \in \mathcal{T}$. By IH, $\mathcal{M}, \mu(y) \models \varphi$. Given $\mathcal{M}, \mu(y) \models \varphi$ and $(x, y) \in \mathcal{R}$, then by the semantics of the $\Box$, $\mathcal{M}, \mu(x) \models \Box\varphi$.
- $y : \Box\varphi \in \Delta$. Since T is saturated, then by $\Box$R in Def. 14, $\exists z \in Lab(T)$, $yRz \in \mathcal{T}$, $z : \Box\varphi \in \Gamma$ and $z : \varphi \in \Delta$. By IH, $\mathcal{M}, \mu(z) \not\models \varphi$. By definition of $\mathcal{R}$, $(y, z) \in \mathcal{R}$,and by the semantics of the $\Box$, $\mathcal{M}, \mu(z) \not\models \Box\varphi$.

Now that we have shown the notions of saturated sequents and how they serve as both a false termination condition and means of getting a counter model, we finally show our algorithm which as previously mentioned is a simple recursive simulation of the rules of CSGL in a specific order (more on that later). The algorithm is shown on the next page for readability purposes.

Lemma 2. *Let* $T = \mathcal{T}, \Gamma \vdash \Delta$ *be a tree sequent. If* `prove`$(T) =$ `true`*, then there exists a proof of* T *in* CSGL.

Proof. 1. The only way $\texttt{prove}(T)$ can return $\texttt{true}$ is in one of the base cases, i.e., when we reach a leaf in the proof tree. There are two cases:
   (a) The more trivial case is the return statement at line 5. This is a direct simulation of the $\bot\mathsf{L}$ rule in CSGL.
   (b) The more interesting case is that of the return statement in line 2. It states that if we have $x : \varphi$ in both Γ and Δ we return true. This case might seem to be simulating id_1 or id_2 but it is actually a direct consequence of property (1) in Thm. 1 that states any tree sequent of the form $\mathcal{T}, \Gamma, x : \varphi \vdash x : \varphi, \Delta$ is provable in CSGL.
2. The rest of the code, namely the section with recursive calls, serves to simulate the logical rules. Where the recursive calls at lines 13, 18, 22, and 29 simulate logical rules $\to\mathsf{L}$, $\to\mathsf{R}$, both $\Box\mathsf{L}$ and $\mathsf{4L}$ simultaneously, and $\Box\mathsf{R}$, respectively. However, it is worth noting that they are not actually direct simulations since there are logical rules that remove labeled formulae whereas our code preserves principal formulae bottom-up. So, for example, instead of having a direct application of $\to\mathsf{L}$ we would have $\to\mathsf{L}'$ which is equivalent to applying normal $\to\mathsf{L}$ and cL as follows.

$$\frac{\mathcal{T}, \Gamma, x : \varphi \to \psi, x : \psi \vdash \Delta \qquad \mathcal{T}, \Gamma, x : \varphi \to \psi \vdash x : \varphi, \Delta}{\mathcal{T}, \Gamma, x : \varphi \to \psi \vdash \Delta} \to\mathsf{L}'$$

$$\frac{\dfrac{\mathcal{T}, \Gamma, x : \varphi \to \psi, x : \psi \vdash \Delta \qquad \mathcal{T}, \Gamma, x : \varphi \to \psi \vdash x : \varphi, \Delta}{\mathcal{T}, \Gamma, x : \varphi \to \psi, x : \varphi \to \psi \vdash \Delta} \to\mathsf{L}}{\mathcal{T}, \Gamma, x : \varphi \to \psi \vdash \Delta}\ \mathsf{cL}$$

   Note that since contraction is hp-admissible this is not a problem.

Lemma 3. *Let $T = \mathcal{T}, \Gamma \vdash \Delta$ be a tree sequent. If $\texttt{prove}(T) = \texttt{false}$, then a counter-model M with M-assignment μ can be constructed such that $M, \mu \not\models T$ witnessing that T does not have a proof in CSGL.*

Proof. We know that the algorithm only returns false if one of the leaves of the derivation becomes a saturated sequent. By Lem. 1, know that this tree sequent is invalid. Since all of our rules are invertible, we know that all tree sequents along the path that reaches the saturated sequent in our proof are also invalid. By Def. 13, a path starts at the base of the proof tree and T is that base of the proof tree, we know that T is invalid. By soundness, we know that no invalid sequent has a proof.

The previous two lemmas afford us soundness and completeness. As a direct result of these lemmas, we have correctness.

Theorem 3 (Correctness). *Let $T = \mathcal{T}, \Gamma \vdash \Delta$ be a tree sequent. (1) If $\texttt{prove}(T) = \texttt{true}$, then T has a proof in CSGL. (2) If $\texttt{prove}(T) = \texttt{false}$, then T does not have a proof in CSGL.*

We now move our attention to termination and search bounds of our algorithm. For a moment, let's ignore the existence of the lines of code that simulate the modal rules $\Box\mathsf{R}$, $\Box\mathsf{L}$, and $\mathsf{4L}$. This means that only the local rules remain and that our modal structure will remain unchanged. This means that for the time being the only way for our algorithm to run forever is if there's one or more sequents that we successively apply an infinite amount of non-initial local rules to. But since there are a finite amount of rules in our systems and all our sequents are made up of a finite number of formulae, these infinite applications would have to include repetitions. But since, our rules are designed to only fire once for each pattern match, the only way we can have infinite applications of the same rules is if we are infinitely generating new labeled formulae it can pattern match with. This is also not possible since every time we introduce new labeled formulae, it is of a smaller length and is actually a subformula of an already existing formula in the sequent, and since the number of formulae is finite then the subformulae of the sequent are also finite. Thus, we can not have an infinite number of applications of the same rule which means that we are bound to reach one of the base cases in finitely many steps. It should be fairly obvious that when we reach the parts of the code that don't do recursive calls and returns either `true` or `false` a branch of the proof is terminating.

Now that we have our local rules handled lets reintroduce the modal rules. The argument that the lines simulating $\Box\mathsf{L}$ and $\mathsf{4L}$ don't cause the code to run forever is similar to the ones for the

local rules. This is not hard to see, as they operate very similarly to local rules. Indeed the issue with the modal rules stems from the dastardly $\Box$R rule. $\Box$R can change the modal structure and add a new successor node y to $\mathcal{T}$ and a new labeled formula with y as the label $(y : \Box\varphi)$. This rule could stop our algorithm terminating in the following ways: (1) It could infinitely branch at a node x, and (2) It could add successors infinitely. We show that neither of those are the case below:

Lemma 4 (Branching and Deepening Bounds).

1. *Every label of every tree sequent generated during proof-search has an out-degree bounded by the number of modal subformulae of the input T: This is due to the fact that* $\Box$R *only fires once for every occurrence of the form $x : \Box\varphi$ in Δ and then never again for that occurrence. So the rule applications are bounded by the number times a box formula can appear in Δ. A formula can only show up in Δ if was already there or if it was added there by a logical rule, but in either case it would already be in the set of subformulae of the input sequent and thus we have an upper bound on branching.*
2. *The depth of every tree sequent is bounded by the number of modal subformulae of the input T: This is due to the fact that* $\Box$R *only fires once for every occurrence of the form $x : \Box\varphi$ in Δ along a branch and then never again. At first, our intuition led us to think that the bound would be the modal depth. The thought process was since we are always propagating formulae to successors of lower complexity, the number of times we can apply* $\Box$R *is just the maximum modal depth, i.e., the modal depth of $\mathcal{T}$. This intuition proved to be incorrect when we considered examples like the following: $x : \Box(\Box p \to \bot) \vdash x : \Box\Box r$. Clearly the modal depth of this tree sequent is two. But then consider this derivation (rules and their pattern matches are highlighted for your convenience where purple means that it is a conclusion of one rule and the premise of another at the same time):*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{xRy, yRz, zRw, \ldots, y : \Box\Box r, z : \Box r, w : \Box p \vdash z : r, w : p}\,\Box\mathsf{R}}{xRy, yRz, \ldots, z : \bot, y : \Box\Box r, z : \Box r \vdash z : r, z : \Box p}\,{\to}\mathsf{L}}{xRy, yRz, \ldots, z : \Box(\Box p \to \bot), z : \Box p \to \bot, y : \Box\Box r, z : \Box r \vdash z : r}\,4\mathsf{L}, \Box\mathsf{L}}{xRy, yRz, x : \Box(\Box p \to \bot), y : \Box(\Box p \to \bot), y : \Box p \to \bot, y : \Box\Box r, z : \Box r \vdash z : r}\,4\mathsf{L}, \Box\mathsf{L}}{xRy, yRz, x : \Box(\Box p \to \bot), y : \Box\Box r, z : \Box r \vdash z : r}\,\Box R}{xRy, x : \Box(\Box p \to \bot), y : \Box\Box r \vdash y : \Box r}\,\Box R}{x : \Box(\Box p \to \bot) \vdash x : \Box\Box r}\,\Box R$$

Clearly in the example laid before you, that this derivation has a depth of three even though the modal depth of the tree sequent was two. This comes from the third (bottom-up) application of $\Box$R *which pattern matched with $z : \Box p$. This labeled formula did not start out on the right but resulted from being a subformula in an implication formula. So clearly the bound of our tree depth is not the modal depth. We will now show that the depth is bounded by the number of subformulae of the tree sequent, i.e., that for every occurrence of the form $x : \Box\varphi$ in Δ along a branch we apply* $\Box$R *once and then never again. We show this in the following example by contradiction. So assume that $x : \Box\varphi$ was unpacked once during the derivation. We know that by the order in which our rules are performed in the algorithm that at some point the rule simulating simultaneous* 4L *and* $\Box$L *application that $\Box\varphi$ will be propagated down the path. Given this, if we were to ever encounter another occurrence of $\Box\varphi$ in the Γ down the same path with a different label, we know that at an earlier point the original $\Box\varphi$ was propagated to our current level and thus we actually have $\Box\varphi$ on both sides of the sequent which triggers the true terminating condition. Which means that because of the way we arranged our rules and the diagonal formula we have a bound on deepening that we wouldn't have had otherwise. This is an additional contribution of this paper.*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{T}', xRy_1, \ldots, y_nRz, \Gamma', y_n : \Box\psi \vdash \Delta', z : \Box\varphi}\,\mathsf{id}_2}{\vdots}}{\mathcal{T}', xRy_1, \ldots, y_nRz, \Gamma', y_n : \Box\varphi, z : \Box\varphi, z : \varphi \vdash \Delta'}\,4\mathsf{L}, \Box\mathsf{L}}}{\vdots}}{\mathcal{T}', xRy_1, \Gamma', x : \Box\varphi, y_1 : \Box\varphi, y_1 : \varphi \vdash \Delta, y_1 : \varphi}\,4\mathsf{L}, \Box\mathsf{L}$$

$$\vdots$$

$$\dfrac{\mathcal{T}, xRy_1, \Gamma, x : \Box\varphi \vdash \Delta, y_1 : \varphi}{\mathcal{T}, \Gamma \vdash \Delta, x : \Box\varphi}\,\Box R$$

Theorem 4 (Termination). `prove`(T) *always terminates for any tree sequent* T.

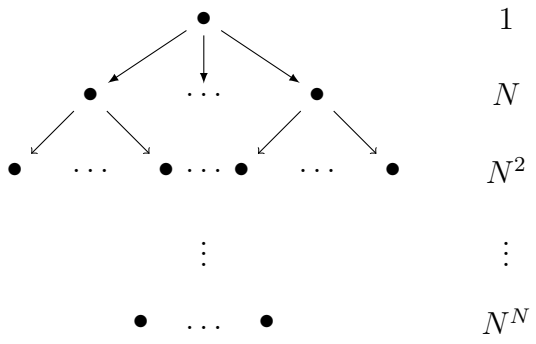
Proof. Lemma 4

With the algorithm shown and the soundness, completeness, and termination results discussed, we have now shown that our contribution is a decision procedure for GL. This was done using system CSGL that Poggiolesi introduced in [5] and in the process answered her open call for a terminating proof search for CSGL. The most interesting result we found was the fact that diagonal formula which made cut elimination so difficult actually lends to the algorithm being terminating.

In the next section we will discuss that while we were able to find a terminating and correct algorithm, we were not able to find one that was optimal with respect to the complexity of the logic.

5 Complexity Analysis

It can be shown that GL is decidable in the computational complexity class PSPACE. Our algorithm is in EXPTIME. This is due to the bounds we showed in Lem. 4. Let T be the input to our `prove`(T), K be the size of our input: $|T| := K$, and N be the number of modal subformulae we have in our input: $|Sub_{\Box}(T)| := N$. (NB. We use $Sub_{\Box}(T)$ to denote all modal subformulae of T, that is, subformuale of the form $\Box\varphi$.) Then it is clear our tree grows in the following way:



The size of our tree can be calculated using $\sum_{i=0}^{N} N^i$. This is a geometric sequence. We then use the following chain of equivalences and inequalities to show that $\sum_{i=0}^{N} N^i \leq 2^{O(K^3)}$ which means the size of our tree is bounded exponentially and thus in $EXPTIME = \bigcup_{i\in\mathbb{N}} 2^{n^i}$:

$$\sum_{i=0}^{N} N^i = \frac{1 - N^{N+1}}{1 - N} \leq N^{O(N^2)} = \left(2^{\log N}\right)^{O(N^2)} = 2^{O(N^2 \log N)} \leq 2^{O(N^3)} \leq 2^{O(K^3)}.$$

6 Conclusion

This paper worked its way up to introducing a proof search algorithm for GL. First, we introduced the logical preliminaries needed to adequately discuss GL. The preliminaries included the syntax

and semantics of modal logic, notions of subformulae, modal depth, and length of a formula, followed by a Hilbert proof system for GL that is sound and complete.

In the next section, we introduced the tree sequents in Poggiolesi [5] in more readable notation. Then, in the new notation we showed the syntax and semantics of tree sequents. We also included a lot helpful terminology to precisely describe tree sequents and notions like modal depth and subformulae of tree sequents for later sections. After, we recalled the system CSGL for GL introduced by Poggiolesi [5], again updated to more readable notation. Then, we explained the inference rules and their classifications and forms again introducing helpful terminology. Finally, we introduced derivations and proofs in the system, and ended the section introducing some properties of CSGL such as admissibility and invertibility.

The fourth section was where our contributions came in. We introduced the notion of a saturated sequent and showed that if the sequent was indeed saturated then it could be used for counter model extraction. After, we answered Poggiolesi's call for a terminating algorithm for CSGL (see [5]). Then we showed our algorithm correct by proving soundness and completeness. After that, we were able to add yet another contribution by way of showing how diagonal formulae in the "Box Right" rule yields termination of our algorithm, thus showing that our algorithm is a decision procedure for CSGL.

In the last section, we went over the complexity of our algorithm and concluded that it was in EXPTIME even though GL is in PSPACE. This leaves open the question on how to optimize our algorithm to make it PSPACE for future work.

References

1. Boolos, G.S.: The Logic of Provability. Cambridge University Press (1994)
2. Bull, R.A.: Cut Elimination for Propositional Dynamic Logic without *. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **38**(2), 85–100 (1992)
3. Kashima, R.: Cut-Free Sequent Calculi for Some Tense Logics. Studia Logica **53**(1), 119–135 (1994)
4. Lyon, T.S.: Unifying Sequent Systems for Gödel-Löb Provability Logic via Syntactic Transformations. In: CSL 2025. LIPIcs, vol. 326, pp. 42:1–42:23. Schloss Dagstuhl (2025). https://doi.org/10.4230/LIPIcs.CSL.2025.42
5. Poggiolesi, F.: A Purely Syntactic and Cut-free Sequent Calculus for the Modal Logic of Provability. The Review of Symbolic Logic **2**(4), 593–611 (2009). https://doi.org/10.1017/S1755020309990244
6. Segerberg, K.: An Essay in Classical Modal Logic. Uppsala: Filosofiska Föreningen och Filosofiska Institutionen vid Uppsala Universitet (1971)
7. Simpson, A.K.: The Proof Theory and Semantics of Intuitionistic Modal Logic. Ph.D. thesis, University of Edinburgh. College of Science and Engineering. School of Informatics (1994)