Bidirectional Proof Search Procedure for Intuitionistic Modal Logic IS5

Hyungchul Park
Pohang University of Science and Technology (POSTECH), Republic of Korea
December 20, 2013

Contents
1 Abstract 3
2 Introduction 3
  2.1 Preliminaries 3
3 Sequent Calculus $G_{IS5}$ 6
  3.1 Multi-contextual Sequent Calculus $G_{IS5}$ 6
  3.2 Structural Properties and Cut-elimination Theorem 8
  3.3 Soundness and completeness of $G_{IS5}$ with respect to the Kripke model for IS5 11
4 Sequent Calculus $G_{IS5}^F$ for Forward Proof Search 12
  4.1 Signed Subformula Property 13
  4.2 Focus on Principal Formula 14
  4.3 Forward Proof Search in $G_{IS5}^F$ 15
  4.4 Soundness and Completeness $G_{IS5}^F$ with respect to $G_{IS5}$ 15
5 Bidirectional Proof Search in Sequent Calculus 16
  5.1 Focused Thread and Derived Rule 16
  5.2 Backward Proof Search in $G_{IS5}^B$ 18
  5.3 Soundness and Completeness $G_{IS5}^B$ with respect to $G_{IS5}^{FN}$ 20
  5.4 Implementation of Bidirectional Proof Search Procedure 21
6 Efficiency of Bidirectional Proof Search Procedure 21
7 Conclusion 22
1 Abstract

We designed a multi-contextual sequent and its sequent calculus systems for intuitionistic propositional modal logic IS5 (IS5). Based on the systems, we developed a bidirectional proof search procedure. In the procedure, we conduct forward proof search to derive rules specialized for a goal formula. Then, we integrate the derived rules and right-rules into a new system on which we conduct backward proof search. We proved that the procedure is logically sound and complete. We also proved that the procedure terminates in finite time.

2 Introduction

Modal logic is a type of formal logic that includes modal operators of necessity □ and possibility ◊. In modal logic, we assume a set of worlds and the modal operators give locality information to formulas. A formula without modal operator is valid in the current world. □A means that A is true in any accessible world. ◊A means that A is true in some accessible worlds.

We focus on intuitionistic modal logic IS5 (IS5). IS5 is a variant of modal logics that includes logical connectives of intuitionistic propositional logic (IPL) and the modal operators. IS5 is characterized by its equivalence relation as world accessibility. Due to the equivalence relation, worlds accessible from the current world are accessible from another accessible world.

Intuitionistic modal logics have various applications in computer science, for instance, for the verification of computer hardware [1] and for the type systems of staged computation [2] and distributed computing [3]. Proof theory of IS5 and other variants of modal logics has been studied [4, 5]. In contrast, proof search in modal logic variants has not received enough attention. Heilala [6] has proposed bidirectional proof search procedures for IS4.

Based on the bidirectional procedures, we propose sequent calculus systems and a bidirectional procedure for IS5. To maintain accessible worlds, we propose a multi-contextual sequent. We developed proof systems for both forward and backward procedure. The bidirectional procedure runs in backward direction using the partial results from forward procedure.

We mechanized the proof-theoretical study on the sequent calculus. We used the proof assistant Coq and mechanized the proofs of proof-theoretical properties of the sequent calculus. We explain the properties that we proved and the automation techniques that we used in the proof.

2.1 Preliminaries

Intuitionistic Logic Intuitionistic logic (constructive logic) differs from classical logic in the definition of provability. In classical logic, propositional formulas are always
assigned a truth value of true or false regardless of whether we have direct evidence for either case. In contrast, a formula in intuitionistic logic is not assigned a truth value of true or false unless we have proof. The law of excluded middle $A \lor \neg A$ which is accepted as an axiom and considered as true formula in classical logic is not true in intuitionistic logic.

**Modal Logic**  Modal logics are an extension of propositional logic with modal operators. Modal operators give more information on a logical statement. For example, ”John is happy” means that a person named John is usually happy. Here, the meaning of ‘usually’ is one of modalities of the English language. There are many modality kinds in languages including formal languages.

In modal logics, we have two modal operators. Necessity $\Box A$ means that $A$ is always true in any world that is accessible from the current world. Possibility $\Diamond A$ means that $A$ is true in some worlds that are accessible from the current world. But, we have no information about in which world $A$ is true.

In modal logics, we consider a set $G$ of accessible worlds and call it frame. The truth of a formula may differ in each of the possible worlds in $G$. To clarify the meaning of accessible worlds, the accessibility between worlds $R$ needs to be introduced for modal logics. In the binary relation $R$, $wRv$ means that the world $w$ is accessible from the world $v$. The truth relation $w \models A$ means that the formula $A$ is true in the world $w$. The model $<G, R, \models>$ describes a modal logic. In a modal logic, the truth of formula is determined by recursive rules. Following are the rules for modal operators:

- $w \models \Box A$ if and only if for every element $v$ of $G$, if $wRv$ then $v \models A$
- $w \models \Diamond A$ if and only if for some element $v$ of $G$, $wRv$ and $v \models A$

**Modal logic S5**  Modal logic S5 is one of modal logic variants. Modal logic S5 is characterized by its accessibility relation between worlds. By the axiomatic system for S5, the relation is an equivalence relation. That is, the relation is reflexive, symmetric, transitive and Euclidean. For worlds in a partition of the equivalence relation, any world is accessible from another world in the partition. Hence, the definitions of truth relation for the modal operators in S5 are:

- $w \models \Box A$ if and only if for any world $v$ of $G$, $v \models A$
- $w \models \Diamond A$ if and only if for some $v$ of $G$, $v \models A$

where $G$ is the set of worlds that are accessible from the current world.

**Proof Search Procedure**  A proof search procedure for a certain logic determines whether a given formula is provable or not. If so, the procedure build a derivation tree which is considered as proof for the formula. A proof search procedure should be sound and complete with respect to the logic. That is, the procedure must find a derivation
tree for a true formula in the logic and the procedure must conclude that a formula is not provable for a false formula in the logic. Depending on the logic, there may be a deterministic procedure or not. The complexity of the procedure may differ by logics. Various proof search strategies and optimization techniques have been proposed. In this paper, we propose a bidirectional proof search procedure for IS5.

**Bidirectional Proof Search** The bidirectional proof search is based on both backward search and forward search. In the first stage of the procedure, we conduct forward proof search to build a set of derived rules. Derived rules are specialized for the goal formula we are working on. The derived rules substitute for the left-rules of the sequent calculus. We incorporate the derived rules and the right-rules to build a new proof system for backward proof search. Conducting backward proof search in the proof system, we build a derivation tree for a true formula.

The bidirectional procedure takes advantage of both backward and forward proof search. By replacing left-rules with derived rules, the procedure skips repetitive left-rule applications. The derived rules make the structure of derivation tree clear and let us capture the redundancy of sequents during the search. The procedure also allows us to avoid infinite loops by bookkeeping set management. Rather than we implement a loop detection method, the loop prevention by the bookkeeping method is integrated into the proof system.

Our bidirectional procedure for IS5 was based on the bidirectional procedure for IS4 [6]. The main idea of combining forward search and backward search was adopted. But, our procedure handles the multi-contextual sequent which maintains the set of accessible contexts derived from the goal formula. In modal logic S4, the relation between worlds is not symmetric. That is, the relation is directed and sequents do not keep the worlds derived from the goal formula.

**Proof Assistant Coq** Coq is one of the most popular proof assistants. It provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs. Coq is not an automated theorem prover but includes automatic theorem proving tactics and various decision procedures.

To mechanize our proof-theoretical study of the sequent calculus for IS5, we used Coq. From the inductive definitions of formula and sequent to the cut-elimination theorem, we wrote the definitions in the Coq language and mechanized the proofs of lemmas and theorems. In the process of mechanization, we exploited the automatic proving tactics and developed automation `ltac` commands to automate the repetitive process of applying tactics.

**Automated Theorem Prover** An automated theorem prover is an implementation of the proof search procedure. With efficiency in proof systems, optimization techniques in implementation can further reduce the computation time to find a derivation tree.
We implemented the bidirectional procedure in OCaml language. In addition to the implementation of the proof search in the proof systems, we implemented optimization strategies to reduce redundancy of the search space.

3 Sequent Calculus $G_{IS5}$

3.1 Multi-contextual Sequent Calculus $G_{IS5}$

In addition to logical connectives of the intuitionistic propositional logic, the grammar of formulas in IS5 includes modal connectives necessity $\Box$ and possibility $\Diamond$.

\[
\begin{align*}
\text{formula} & : A, B, C ::= p | \top | \bot | A \wedge A | A \lor A | A \supset A | \Box A | \Diamond A \\
\text{context} & : \Gamma, \Delta ::= \{A, B, C\ldots\} \\
\text{frame} & : G ::= \{\Gamma_1; \Gamma_2; \ldots; \Gamma_n\} \\
\text{sequent} & : G \vdash \Delta \vdash \Gamma \vdash C
\end{align*}
\]

$p$ represents atomic formulas which cannot be destructed into subformulas. $\top, \bot, A \wedge A, A \lor A$ and $A \supset A$ are from the intuitionistic propositional logic. The modal formula $\Box A$ means that $A$ is true in every accessible worlds of $G$. The modal formula $\Diamond A$ means that $A$ is true in some accessible worlds of $G$.

A context is a set of formulas that are true in the corresponding world. Especially, the current context $\Gamma$ of a sequent $G \vdash \Delta \vdash \Gamma \vdash C$ stands for the current world. The global context $\Delta$ contains the formulas that are true in every accessible worlds. The necessity operator $\Box$ and the global context $\Delta$ share the meaning of global truth over the accessible worlds. The frame $G$ is a set of contexts that describes the accessible worlds. The frame $G$ is an enumeration of accessible worlds which are known by the analysis of the current world and its formulas. Since the accessibility relation in IS5 is an equivalence relation, the contexts of $G$ has neither directedness nor order. We can treat the contexts of $G$ as a set and assume that a context of $G$ is accessible by another context of $G$. The sequent $G \vdash \Delta \vdash \Gamma \vdash C$ states that $C$ is true in the current world under the assumption of the contexts.
The sequent calculus consists of left-rules and right-rules on logical connectives including modal operators $\Box$ and $\Diamond$. Each inference rule in the sequent calculus is named after the logical connective of the principal formula. The formula of interest on which a inference rule is applied is called principal formula. In the name of each named after the logical connective of the principal formula. The formula of interest including modal operators

Local context $\Gamma$ or another context in frame $G$ is added. Since a formula in the global context $\Delta$ is globally true, we can extract the have subscript a new formula into the local context. The left-rules adding a new formula into the frame inference rules for the modal operators. For the non-modal connectives, the inference rules for the propositional connectives, we introduced the left-rules adding a new formula into the frame in the sequent calculus $G_{1S5}$.

Figure 1: Sequent calculus $G_{1S5}$
context. The right-rule □R moves the local context into the frame $G$ and empties the local context. If we can prove $A$ under no assumption from the local context, then $A$ is globally true in any accessible context. The left-rules for ◇ also show a uniform structure of adding a new context into the frame. Regardless of the context where ◇$A$ is located, ◇$A$ means a context where $A$ is true. Due to the equivalence relation of accessibility, the new context with $A$ is accessible to any contexts in the sequent, including the local context. The right-rules for ◇ select any context and take it as the local context. That is, to prove ◇$A$, we can prove $A$ in any accessible world.

### 3.2 Structural Properties and Cut-elimination Theorem

We conducted proof-theoretical study on the proposed sequent calculus $G_{IS5}$. We stated the cut-elimination theorem for the multi-contextual sequent calculus $G_{IS5}$. To prove the cut-elimination theorem, we proved structural properties and inversion properties. The proofs of the cut-elimination theorem and involving properties are mechanized into Coq proof scripts.

**Definition** *(Size-preserving property)*

Assume a property which states that if $G ⊢ \Delta ⊢ \Gamma ⊢ C$ then $G' ⊢ \Delta' ⊢ \Gamma' ⊢ C'$. Using the usual definition of proof size (Appendix A), we can compare derivation tree sizes of the sequents. If the derivation tree of $G' ⊢ \Delta' ⊢ \Gamma' ⊢ C'$ has a size less than or equal to the size of the derivation tree for $G ⊢ \Delta ⊢ \Gamma ⊢ C$, the property is size-preserving. With proof size notations for sequents, the property reads:

If $n ≃ G ⊢ \Delta ⊢ \Gamma ⊢ C$ then $n ≥ G' ⊢ \Delta' ⊢ \Gamma' ⊢ C'$.

The following structural properties are size-preserving and used in the proof of the cut-elimination theorem. The cut-elimination theorem was proved by induction on the size of the derivation tree. To prove an induction case, we need to prove its conclusion sequent under the condition on proof sizes. For example, some cases say that under the assumption of derivation tree with size $n$, we should prove the conclusion sequent by providing a derivation tree with the size $n + 1$ or less. To prove the conclusion sequent with the size condition, we require the structural properties to be size-preserving.

**Proposition 3.1** *(Weakening property)*

1. If $n ≃ G ⊢ \Delta ⊢ \Gamma ⊢ C$ then $n ≥ G ⊢ \Delta ⊢ \Gamma, A ⊢ C$.
2. If $n ≃ G ⊢ \Delta ⊢ \Gamma ⊢ C$ then $n ≥ G ⊢ \Delta, A ⊢ \Gamma ⊢ C$.
3. If $n ≃ G; \Gamma' ⊢ \Delta ⊢ \Gamma ⊢ C$ then $n ≥ G; \Gamma', A ⊢ \Delta ⊢ \Gamma ⊢ C$.
4. If $n ≃ G ⊢ \Delta ⊢ \Gamma ⊢ C$ then $n ≥ G; \Gamma' ⊢ \Delta ⊢ \Gamma ⊢ C$.

**Proposition 3.2** *(Exchange property)*
If \( n \simeq G \vdash \Delta \vdash \Gamma, A, B, \Gamma_2 \vdash C \) then \( n \geq G \vdash \Delta \vdash \Gamma, B, A, \Gamma_2 \vdash C \).

(2) If \( n \simeq G \vdash \Delta_1, A, B, \Delta_2 \vdash \Gamma \vdash C \) then \( n \geq G \vdash \Delta_1, B, A, \Delta_2 \vdash \Gamma \vdash C \).

(3) If \( n \simeq G; \Gamma_1, A, B, \Gamma_2' \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G; \Gamma_1', B, A, \Gamma_2' \vdash \Delta \vdash \Gamma \vdash C \).

(4) If \( n \simeq G_1; \Gamma_1'; \Gamma_2; G_2 \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G_1; \Gamma_2'; \Gamma_1; G_2 \vdash \Delta \vdash \Gamma \vdash C \).

**Proposition 3.3** (Contraction property)

(1) If \( n \simeq G \vdash \Delta \vdash \Gamma, A, A \vdash C \) then \( n \geq G \vdash \Delta \vdash \Gamma, A \vdash C \).

(2) If \( n \simeq G \vdash \Delta, A \vdash \Gamma, A \vdash C \) then \( n \geq G \vdash \Delta, A \vdash \Gamma \vdash C \).

(3) If \( n \simeq G \vdash \Delta, A, A \vdash C \) then \( n \geq G \vdash \Delta, A \vdash \Gamma \vdash C \).

(4) If \( n \simeq G; \Gamma', A, A \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C \).

(5) If \( n \simeq G \vdash \Delta, A \vdash \Gamma, A \vdash C \) then \( n \geq G \vdash \Delta, A \vdash \Gamma \vdash C \).

(6) If \( n \simeq G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G; \Gamma' \vdash \Delta, A \vdash \Gamma \vdash C \).

(7) If \( n \simeq G; \Gamma \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G; G \vdash \Delta \vdash \Gamma \vdash C \).

(8) If \( n \simeq G_1; \Gamma'; \Gamma' \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G_1; \Gamma'; G_2 \vdash \Delta \vdash \Gamma \vdash C \).

Even though we defined a context as a set of formulas and a frame as a set of contexts, the syntax of the inference rules treats them as lists. That is why we have the structural properties which are not necessary to prove under the definitions as sets. Note that some structural properties are about contexts. We call the structural properties on contexts modal properties. The weakening property (4) is on the frame. If we have a derivation tree for \( n \simeq G \vdash \Delta \vdash \Gamma \vdash C \), then we can build a derivation tree for \( n \geq G, \Gamma' \vdash \Delta \vdash \Gamma \vdash C \) with an arbitrary context \( \Gamma' \) added to the frame. The exchange property (4) states that we can change the order of contexts in the frame. The contraction property (5) and (6) let us remove a redundant formula which appears in the global context and another context. The resulting sequent without redundancy of formula \( A \) has a derivation tree by the properties. The contraction property (7) and (8) are about contexts in the frame. If a context in the frame is equal to the current context, the current world has an access to another world that has exactly the same information of the current world. We can build a derivation tree for a sequent without the redundancy of contexts. In contrast, we cannot remove the current world when we have a redundant context in the frame. The contraction property (8) states that we can contract one of two identical contexts in the frame.

**Proposition 3.4** (Inversion property \( \land \))
\begin{itemize}
\item If \( n \simeq G \vdash \Delta \vdash \Gamma, A \land B \vdash C \) then \( n \geq G \vdash \Delta \vdash \Gamma, A, B \vdash C \).
\item If \( n \simeq G \vdash \Delta, A \land B \vdash \Gamma \vdash C \) then \( n \geq G \vdash \Delta, A, B \vdash \Gamma \vdash C \).
\item If \( n \simeq G; \Gamma', A \land B \vdash \Delta \vdash \Gamma \vdash C \) then \( n \geq G; \Gamma', A, B \vdash \Delta \vdash \Gamma \vdash C \).
\item If \( n \simeq G \vdash \Delta \vdash \Gamma \vdash A \land B \) then \( n \geq G \vdash \Delta \vdash \Gamma \vdash A \) and \( n \geq G \vdash \Delta \vdash \Gamma \vdash B \).
\end{itemize}

An inference rule is invertible if the conclusion sequent implies the premise sequents. Some inference rules in \( G_{155} \) are invertible. In proposition 3.4 is the list of inversion properties on left-rules and right-rules of conjunction \( \land \). In addition to conjunction \( \land \), we have inversion properties on necessity \( \Box \) and possibility \( \Diamond \) (Appendix B). Since the proof of cut-elimination theorem involves the inversion properties, the properties are stated in the size-preserving form.

**Proposition 3.5 (Cut-elimination theorem)**

\begin{itemize}
\item If \( G \vdash \Delta \vdash \Gamma \vdash A \) and \( G \vdash \Delta \vdash \Gamma, A \vdash C \) then \( G \vdash \Delta \vdash \Gamma \vdash C \).
\item If \( G; \Gamma \vdash \Delta \vdash \cdot \vdash A \) and \( G \vdash \Delta, A \vdash \Gamma \vdash C \) then \( G \vdash \Delta \vdash \Gamma \vdash C \).
\item If \( G; \Gamma \vdash \Delta \vdash \Gamma' \vdash A \) and \( G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C \) then \( G \vdash \Delta \vdash \Gamma \vdash C \).
\end{itemize}

The cut-elimination theorem were simultaneously defined and proved by induction on the structure of the cut-formula \( A \) and the proof sizes of the premise sequents. The proof involves the structural properties and the inversion properties. We mechanized the proof of the structural properties, the inversion properties and the cut-elimination theorem using Coq. Three statements in proposition 3.5 can rewritten as inference rules:

\[
\frac{G \vdash \Delta \vdash \Gamma \vdash A \quad G \vdash \Delta \vdash \Gamma, A \vdash C}{G \vdash \Delta \vdash \Gamma \vdash C} \quad \text{cut}^{\text{local}} \quad \frac{G; \Gamma \vdash \Delta \vdash \cdot \vdash A \quad G \vdash \Delta, A \vdash \Gamma \vdash C}{G \vdash \Delta \vdash \Gamma \vdash C} \quad \text{cut}^{\text{global}} \quad \frac{G; \Gamma \vdash \Delta \vdash \Gamma' \vdash A \quad G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C}{G; \Gamma' \vdash \Delta \vdash \Gamma \vdash C} \quad \text{cut}^{\text{frame}}
\]

The cut-elimination theorem states that a derivation tree that makes use of the cut rules also possesses a cut-free derivation tree. The cut-elimination theorem has significant meaning that we need no cut-formula in a derivation tree. In cut-rules, cut-formula \( A \) has no relation with any formula in the conclusion sequent. By the theorem, we can rule out arbitrary formulas that are not related to the goal formula and conduct proof search within the set of related formulas. Formulas related with the goal formula are subformulas which we will discuss later.
3.3 Soundness and completeness of $G_{IS5}$ with respect to the Kripke model for IS5

**Theorem 3.6** (Soundness of $G_{IS5}$ with respect to the Kripke model for IS5)

If $G \vdash \Delta \vdash \Gamma \vdash C$ in $G_{IS5}$ then it is valid in the Kripke model for IS5.

**Theorem 3.7** (Completeness of $G_{IS5}$ with respect to the Kripke model for IS5)

If $A$ is valid in the Kripke model for IS5 then $\cdot \vdash \cdot \vdash \cdot \vdash A$ in $G_{IS5}$.

The soundness and completeness theorem states that the proof system of $G_{IS5}$ is equivalent to IS5. The valid formulas in IS5 has a derivation tree in $G_{IS5}$ and a formula with a derivation tree in $G_{IS5}$ is valid in IS5. To prove the theorem, we rely on an intermediate proof system $G$ from [7], in which the authors proved the soundness and completeness of $G$ with respect to the Kripke model for IS5. We proved soundness and completeness of our $G_{IS5}$ with respect to $G$ (Appendix D). Because the soundness and completeness relation has transitivity, we can complete the proof of theorems 3.6 and 3.7 by incorporating the two steps of the soundness and completeness proofs.
4 Sequent Calculus $G^F_{1S5}$ for Forward Proof Search

Figure 2: Sequent calculus $G^F_{1S5}$
Sequent calculus $G^F_{IS5}$ for forward proof search is based on $G_{IS5}$. Most of the inference rules in $G^F_{IS5}$ are similar to their counterparts in $G_{IS5}$. $G^F_{IS5}$ differs from $G_{IS5}$ in three aspects. First, each formula in a sequent is assigned a sign. The sign is determined by the signed subformula relation. Only the signed subformula of a goal formula may appear in a proof search procedure. Second, the notations for focus on a principal formula were introduced to $G^F_{IS5}$. When a sequent has a focus on its principal formula, we are allowed to apply inference rules only on the principal formula, not on an arbitrary formula in the contexts. Lastly, the contexts in the conclusion sequent of an inference rule is either an empty set or a union of the contexts in the premise sequents. This has a significant meaning that all the formulas in the contexts must be used up exhaustively in the derivation tree.

### 4.1 Signed Subformula Property

\[
\begin{align*}
A_1^+, A_2^+ &\leq (A_1 \land A_2)^+ & A_1^+, A_2^+ &\leq (A_1 \lor A_2)^+ & A_1^+, A_2^+ &\leq (A_1 \supset A_2)^+ \\
A^+ &\leq (\Diamond A)^+ & A^+ &\leq (\lozenge A)^+ & A^- &\leq A^\sim \\
A_1^-, A_2^- &\leq (A_1 \land A_2)^\sim & A_1^-, A_2^- &\leq (A_1 \lor A_2)^\sim & A_1^+, A_2^- &\leq (A_1 \supset A_2)^\sim \\
A^- &\leq (\Diamond A)^\sim & A^- &\leq (\lozenge A)^\sim \\
A_1^\approx, A_2^\approx &\leq (A_1 \land A_2)^\approx & A_1^\approx, A_2^\approx &\leq (A_1 \lor A_2)^\approx & A_1^+, A_2^\approx &\leq (A_1 \supset A_2)^\approx \\
A^- &\leq (\Diamond A)^\approx & A^- &\leq (\lozenge A)^\approx 
\end{align*}
\]

Figure 3: Definition of signed subformula relation ($\leq$)

Signed subformulas of a goal formula include all the formulas that can appear in the backward proof search $G_{IS5}$. A sign of a subformula gives information about in which part the subformula can appear. The signed subformula relation includes five types of signs. ($+$) sign means that its formula can appear as a conclusion formula of a sequent in the derivation tree. A subformula with ($-$) and ($=$) signs can appear in a context. ($\sim$) and ($\approx$) signs are for subformulas as principal formulas. Especially, formulas with double-lined signs ($=$) and ($\approx$) belong to the global context.
By the signed subformula relation, we can enumerate all the signed subformulas which may appear in the derivation tree. For brevity, we allocated numbered labels for each subformula (Fig. 4). In the derivation tree of the goal formula, only the signed subformulas may appear. Thus, during proof search for $L_1$, we need not consider $A^=$ which does not belong to the set of signed subformulas of $L_1$.

**Theorem 4.1** (Signed subformula property)
Every sequent in a $G_{IS5}$ derivation tree of

$$G^\neg \vdash \Delta^= \vdash \Gamma^- \vdash C^+$$

is in the form of

$$\Gamma_1^-; \ldots; \Gamma_n^- \vdash D_1^=; \ldots; D_m^= \vdash E_1^-\ldots E_l^- \vdash F^+$$

where all $D_i^=, E_j^-$, all formulas $H_{ij}$ in $\Gamma_i^-$ and $F^+$ are signed subformulas of $G^\neg, \Delta^=$, $\Gamma^-$ and $C^+$.

By the cut elimination theorem, we need not consider cut-formulas during proof search. In the cut-rules, the cut-formula $A$ introduced to the premise sequents is an arbitrary formula which is not related to any formula in the goal sequent. In contrast, the formulas newly introduced to premise sequents by inference rules are subformulas of the principal formula. In conclusion, derivation trees in $G_{IS5}$ consist of signed subformulas of the goal sequent and do not include an arbitrary formula.

The signed subformula property allows forward proof search for $G_{IS5}^F$. The property restricts the search space to the set of sequents with signed subformulas: $\Gamma_1^-; \ldots; \Gamma_n^- \vdash D_1^-; \ldots; D_m^- \vdash E_1^-; \ldots; E_l^- \vdash F^+$. By the property, the forward proof search is guaranteed to end with search space saturation. Without the property, the forward proof search would generate an infinite number of true sequents.

4.2 Focus on Principal Formula
A sequent in $G_{IS5}^F$ may be equipped with an explicit focus on its principal formula. If a sequent has a focused principal formula, we can apply an inference rule only on
the principal formula. We have two types of focus notations (▷, ▷▷). A focus with
▷ means that the principal formula is originated from the local context or another
context of the frame. In contrast, a focus with ▷▷ means that the principal
formula is originated from the global context. A focus is located either on the local context
or on another context of the frame. The context with the focus is where the principal
formula resides and an inference rule is applied.
Four inference rules ([ch_{local}], [ch_{global}^{a}], [ch_{global}^{b}], [ch_{frame}]) which introduce a focus to
their premise sequent were added to \( G_{I5}^F \). In the forward direction, the four inference
rules ends the focus and add the principal formula to the corresponding context. Some
left-rules have a conclusion sequent with a focus whereas their premise sequents have no
focus (\( \square L_i \), \( \vee L_i \)). The other left-rules have focuses in both of the conclusion
sequent and the premise sequents.
A sequent with a focus formula is restricted in terms of inference rule applications.
One might doubt that \( G_{I5}^F \) with the restriction fails to prove a valid goal formula that
has a derivation tree in \( G_{I5} \). However, we proved that \( G_{I5}^F \) is sound and complete
with respect to \( G_{I5} \).

4.3 Forward Proof Search in \( G_{I5}^F \)

Using \( G_{I5}^F \), we can design a forward proof search procedure. First, we enumerates
all the signed subformulas for the given goal formula. During the forward search, we
only consider formulas in the set of signed subformulas. Second, we generate all the
valid sequents that we can derive with a single terminal inference rule. The set of valid
sequents is our starting point. Third, we expand the set of valid sequents by taking
sequents from the set and applying an inference rule. Since we have valid sequents in
the set, we can take them as premise sequents of another rule application and generate
the conclusion sequent. By repeatedly expanding the set of valid sequents, we reach
the saturation point where all the possible sequents are examined. If the goal sequent
is found during the expansion, we succeed to build a derivation tree. Otherwise, the
goal sequent is impossible to prove and there is no derivation tree.
We may implement a forward proof search procedure based on \( G_{I5}^F \). The focuses of the
inference rules enable the idea of back-chaining. But, in our bidirectional procedure, we
conduct the forward search only for partial results. Instead of expanding the sequent
set until we reach the saturation point, we only generate derived rules that are small
parts of complete derivation trees. In the next section, we will discuss how we run the
forward proof search and generate derived rules specialized for the goal formula.

4.4 Soundness and Completeness \( G_{I5}^F \) with respect to \( G_{I5} \)

Theorem 4.2 (Soundness of \( G_{I5}^F \) with respect to \( G_{I5} \))

1. If \( G^− \vdash \Delta^− \vdash \Gamma^− \rightarrow C^+ \) then \( G \vdash \Delta \vdash \Gamma \vdash C \).
2. If \( G^− \vdash \Delta^= , A^\sim \vdash \Gamma^> \rightarrow C^+ \rightarrow A^\sim \) then \( G \vdash \Delta \vdash \Gamma , A \vdash C \).
3. If $G \vdash \Delta = \Gamma \vdash C^+$ then $G \vdash \Delta, A \vdash \Gamma \vdash C$. 

4. If $G; \Gamma' \vdash C^+ \vdash \Delta = \Gamma \vdash A^\sim$ then $G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C$. 

5. If $G; \Gamma' \vdash \Delta = \Gamma \vdash A \approx C$ then $G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C$.

**Theorem 4.3** (Completeness of $G_{IS5}^F$ with respect to $G_{IS5}$)

If $G \vdash \Delta \vdash \Gamma \vdash C$ then $G' \vdash \Delta' = \Gamma' \vdash C^+$

where $G'$, $\Delta'$ and $\Gamma'$ are subsets of $G$, $\Delta$ and $\Gamma$, respectively.

Together, theorems 4.2 and 4.3 state the logical equivalence between the two systems $G_{IS5}^F$ and $G_{IS5}$. Theorem 4.2 states that any form of sequents in $G_{IS5}^F$ have a derivation tree in $G_{IS5}$. For a focus and its principal formula $A$, we add $A$ to the context of interest. In a derivation tree of $G_{IS5}$, $A$ is implicitly focused as the principal formula. Theorem 4.3 states that if we have a derivation tree in $G_{IS5}$ then we can find a derivation tree in $G_{IS5}^F$ for another sequent. The sequent in $G_{IS5}^F$ has subsets of the corresponding contexts in $G_{IS5}$. Since $G_{IS5}^F$ requires a derivation tree to use all the formulas in contexts exhaustively, the conclusion sequent may have less formulas in the contexts than the sequent in $G_{IS5}$. 

$G_{IS5}^F$ and $G_{IS5}$ differ in two aspects. First, $G_{IS5}^F$ has the focus and its restriction of rule applications. The restriction complicates the proof of completeness. Second, $G_{IS5}^F$ is for forward proof search. The terminal inference rules of $G_{IS5}^F$ have empty contexts and a principal formula. For the non-terminal inference rules, a context in a conclusion sequent of $G_{IS5}^F$ is union of corresponding contexts in premise sequents. For example, in the inference rule $\land R$ of $G_{IS5}^F$, the local context of the conclusion sequent is $\Gamma_1, \Gamma_2$.

To prove theorems 4.2 and 4.3, we introduced intermediate proof systems $G_{IS5}^I$ and $G_{IS5}^{FN}$ (Appendix E). In the first step of the proof, we proved the logical equivalence between $G_{IS5}$ and $G_{IS5}^I$. $G_{IS5}^I$ keep a copy of the principal formula in the premise sequents. $G_{IS5}^F$ selects one among the two formulas of a conjunction connective as we follow up to premise sequents. Then, in the next step, we proved soundness and completeness of $G_{IS5}^{FN}$ with respect to $G_{IS5}^I$. $G_{IS5}^{FN}$ has focuses and the focus conditions on principal formulas. But, $G_{IS5}^{FN}$ is for backward proof search and a sequent may contain formulas that are not used in the derivation tree. In the last step, we prove that $G_{IS5}^{FN}$ and $G_{IS5}^F$ are logically equivalent. By transitivity of soundness and completeness relations, we incorporated the two steps of soundness and completeness proofs and proved the theorems.

## 5 Bidirectional Proof Search in Sequent Calculus

### 5.1 Focused Thread and Derived Rule

**Definition** *(Focused thread)*

A focused thread is a partial derivation tree which starts with an rule application that
introduces a focus at the top and ends with one of [ch] inference rules that has no focus in the conclusion sequent. The inference rules that introduce a focus include [☐L], [◇L] and [∨L].

A focused thread is a partial derivation tree with a series of sequents with focuses. The premise sequents at the top of a focused thread have no derivation tree yet. Except for the top and bottom ends, inference rule applications in a focused thread consist of left-rules that have focuses on both a premise sequent and a conclusion sequent. Thus, a focused thread forms a single thread of sequents with focuses. Premise sequents without a focus may appear in a focused thread by the inference rule [⊃R]. But, they do not interfere with the principal formulas of a focused thread. Note that selecting an inference rule in a focused thread is deterministic. Once we select a label number for a certain formula and introduce a focus to the formula at the top of a focused thread, the principal formulas along the focuses are determined. The principal formulas of the premise sequents are children formulas in the subformula relation and the principal formula of the conclusion sequent is the parent formula. For example, assume that we choose $L_7$ and $L_8$ as the principal formulas at the top (Fig 5). In the next sequent of the focused thread, the principal formula of focus is determined to be $L_6$ which is the parent formula of $L_7$ and $L_8$.

A derived rule substitute for a partial derivation tree in a focused thread form. A derived rule has premise sequents and a goal sequent as a normal inference rule does. Premise sequents of a derived rule correspond to the sequents that appear in the focused thread and the premise sequents are yet to be proved. The idea of derived rules is that we can abstract a sequence of left-rule applications in a focused thread. Using derived rules, we can avoid repeating the same focused thread every time we encounter a sequent with the particular formula in the focus.

\[
\begin{align*}
L_7^+ & \vdash \vdash \vdash M^+ & L_8^+ & \vdash \vdash \vdash M^+ & \forall L^{frame} \\
\vdash L_7^+ & \vdash \vdash \vdash M^+ & \vdash \vdash \vdash L_8^+ & \forall L^{frame} \\
\vdash \vdash \vdash M^+ & \vdash \vdash \vdash L_4^+ & \forall L^{frame} \\
\vdash \vdash \vdash M^+ & \vdash \vdash \vdash \vdash L_6^+ & \forall L^{frame} \\
L_3^+ & \vdash \vdash \vdash M^+ & \forall L^{frame} \\
\vdash \vdash \vdash M^+ & \vdash \vdash \vdash \vdash L_2^+ & \forall L^{frame} \\
\vdash \vdash \vdash M^+ & \vdash \vdash \vdash \vdash L_1^+ & \forall L^{frame} \\
G; \Gamma_1 \vdash \Delta \vdash \Gamma_2, L_3 \vdash L_4 & G; \Gamma_2, L_3, L_7 \vdash \Delta \vdash \Gamma_1 \vdash M & G; \Gamma_2, L_3, L_8 \vdash \Delta \vdash \Gamma_1 \vdash M & \text{(derived rule)}
\end{align*}
\]

**Figure 5:** An example of a focused thread and its derived rule

In (Fig. 5), the inference rule $[\forall L^{frame}]$ at the top introduces a focus to the conclusion sequent. The resulting sequent has a focus on $L_{10}$. Only the inference rule $[\forall L^{frame}]$ can be applied to the conclusion sequent with $L_{10}$ in the focus. Applying proper inference rules on the formulas in the focus, we reach inference rule $[\forall L^{frame}]$.
which generates the final sequent without a focus. The premise sequent with $L_8$ in its conclusion is called a twig. Premise sequents at the top are called trunks. Twig sequents have the same composition of contexts as the final conclusion sequent. Trunk sequents, however, have additional formulas $L_{11}$ and $L_{12}$ in the context of interest. This observation is important in the backward proof search which we discuss in the following section.

### 5.2 Backward Proof Search in $G_{IS5}^B$

To find a derivation tree for a given goal sequent, we conduct backward proof search in the proof system $G_{IS5}^B$ specialized for the goal sequent. $G_{IS5}^B$ consists of derived rules from the forward proof search and right-rules of $G_{IS5}$. The idea is that we replace all the left-rule applications with derived rules.

In $G_{IS5}^B$, we conduct backward proof search. The backward proof search is straightforward. We start with the goal sequent and try to apply inference rules to it. The inference rule application will generate premise sequents which we will try to prove recursively. If a sequent has no premise sequent on its inference rule application, then the sequent is proved. If a conclusion sequent has premise sequents and we recursively found derivation trees for each premise sequent, then the conclusion sequent is also proved.

An example (Fig. 6) shows a derivation tree which was found by the bidirectional proof search. The goal formula and its signed subformulas are labeled with numbers (Fig. 6a). Following the signed subformula relation, we assign a sign for each formula. Each derived rule was annotated with the inference rule type and the label of its principal formula (Fig. 6b). The type of a derived rule is the type of the left-rule at the top of the focused thread. The principal formula of each derived rule is the principal formula of the conclusion sequent at the bottom of the focused thread. For example, the second derived rule is named $\vee L(L_3)$ because the left-rule application at the top is $\vee L_{local}$ type and the principal formula of the left-rule application at the bottom is $L_3$.

The derivation tree consists of derived rules and right-rules (Fig. 6c). The derivation tree was built by the backward proof search in $G_{IS5}^B$. The applications of derived rules were annotated with instance tuples. A derived rule instance indicates the principal formula and the context which the subformulas are added to. For example, in the derivation tree, the rule application $\Diamond L_3(L_2, l)$ is focused on $L_2$ of the conclusion sequent and adds a subformula $L_3$. The principal formula $L_2$ resides in the local context which is represented by $l$. Each derived rule instance represents a derived rule application and is used in the loop prevention mechanism.

One complication of backward proof search is that the recursive procedure may fall into an infinite loop of sequents in a branch of the derivation tree. A well-known method is to check a loop whenever we encounter a new sequent. We trace the branch of the derivation tree from the current sequent to the bottom goal sequent. Tracing back from the current sequent to the conclusion sequent of each rule application, if we
find a sequent which is exactly the same as the current sequent or logically weaker than the current sequent, we abort the proof search of the current sequent. For example, \( A \vdash B \vdash C \) is weaker than \( \vdash A \vdash B \vdash C \). During proof search of a goal sequent, if we meet a stronger sequent than the goal sequent, then building the current derivation tree is pointless. Proving a stronger sequent is unnecessary redundancy which leads to an infinite loop.

We propose a bookkeeping method which substitutes for the loop detection method. Each sequent is coupled with bookkeeping sets. Before we apply an inference rule to a sequent, we consult with the bookkeeping method to decide redundancy. If the method decides that the inference rule application is unnecessary redundancy, we abort the current proof search on the sequent.

Five sets of derived rule instances \( \square \mathcal{L}^R, \Diamond \mathcal{L}^R, \mathcal{L} \mathcal{L}^R, \mathcal{L} \mathcal{L}^R, \mathcal{L} \mathcal{L}^R \) are resource sets for each derived rule type and two twig sets \( \square \mathcal{L}^T, \mathcal{L} \mathcal{T}^T \) are twig sets of \( \square \mathcal{L} \) derived rules and \( \mathcal{L} \mathcal{T} \) derived rules, respectively. A bookkeeping set consists of derived rule instances. A derived rule instance is used as an identification to distinguish a derived rule application on a certain formula in a certain context. The instance is a tuple of a derived rule identification and a context number. A context is given an identification number as the context is introduced to a premise sequent. For example, a formula \( \Diamond A \) in a context can introduce a new context with formula \( A \) into the frame. Having a context number in each derived rule instance, we can distinguish the context on which the derived rule was applied. If the resource set of the current sequent does not contain the derived rule instance, then we avoid the rule application to prevent redundancy.
A derived rule has the structure of the inference rules. It has premise sequents at its top and a goal sequent at the bottom. We can classify the premise sequents into two groups, twig and trunk. The above is an example of a derived rule instance. Derived rule $\lor L(R)$ is applied to $G; \Gamma_2, L_3 \vdash \Delta \vdash \Gamma_1 \vdash C$. The context of interest is $\Gamma_2, L_3$ and the trunk sequents have additional formulas $L_7$ and $L_8$ in the context of interest. Managing bookkeeping sets differs for twig sequents and trunk sequents. When we apply a derived rule to a sequent, we add the derived rule instance to the corresponding resource set. Some types of derived rules empty resource sets or move a twig set into an resource set. The details of how we manage bookkeeping sets for other derived rule types are explained in Appendix F.

5.3 Soundness and Completeness of $G_{IS5}^B$ with respect to $G_{IS5}^{FN}$

**Theorem 5.1** (Soundness of $G_{IS5}^B$ with respect to $G_{IS5}^{FN}$)
If $\vdash \vdash \vdash \vdash A$ in $G_{IS5}^B$, with initial bookkeeping sets generated from $A$, then $\vdash \vdash \vdash \vdash A$ in $G_{IS5}^{FN}$.

**Theorem 5.2** (Completeness of $G_{IS5}^B$ with respect to $G_{IS5}^{FN}$)
If $\vdash \vdash \vdash \vdash A$ in $G_{IS5}^{FN}$ then $\vdash \vdash \vdash \vdash A$ in $G_{IS5}^B$, with initial bookkeeping sets generated from $A$. 
Theorems 5.1 and H.3 state that the backward proof search using derived rules can prove sequents that can be proved in $G_{IS5}^{FN}$ (Appendix H). With all the soundness and completeness theorems between the proof systems, we conclude the logical equivalence relations between IS5 and the backward proof search in $G_{IS5}^{B}$. In conclusion, the bidirectional proof search procedure is a valid procedure to find a proof for a goal sequent in IS5.

**Theorem 5.3** (Termination of Backward Proof Search in $G_{IS5}^{B}$)

*Backward proof search in $G_{IS5}^{B}$ is guaranteed to terminate.*

If the procedure does not terminate for a query sequent, we cannot determine the provability of the sequent. Theorem 5.3 states that the procedure ends in finite time and determine whether the goal sequent is provable. To prove theorem 5.3, we defined a measure vector (Appendix G). For each sequent found in the backward proof search, we can use the bookkeeping sets to calculate the measure vector. The measure vectors have the zero vector as the lower bound and decreases monotonically along branches of derivation trees. The upper bound of branch depth imposed by the measure vector leads to the termination of the procedure.

### 5.4 Implementation of Bidirectional Proof Search Procedure

We demonstrated the bidirectional proof search by developing a theorem prover in OCaml language. The program first conducts forward proof search to find all the derived rules specialized for the goal sequent, then builds the proof system $G_{IS5}^{B}$ for the backward proof search. The program conducts backward proof search using the derived rules and right-rules. The program involves optimization methods other than the search space reduction by bidirectional proof search. One method is to identify the twig region sequents of which share the same context. If the two sequents have the same goal formula, the derivation tree can be shared. Another method is to eliminate duplicated contexts which do not contribute to the proof search. Further optimization of the implementation is still possible.

### 6 Efficiency of Bidirectional Proof Search Procedure

Comparison of the proposed procedure with other proof search procedures for IS5 was impossible because little work has been done to develop proof search procedures for IS5 whereas proof-theoretical studies for the modal logics has been done. In this section, we give explanations of principles we used in our procedure to improve the search efficiency.

To find a derivation tree for the goal sequent, a procedure should traverse the search space for the goal sequent. The number of sequents in the search space depends on
how the procedure applies inference rules. Even though the amount of computation to process a single sequent may differ for procedures and affect the time required to find a derivation tree, we can still compare the sizes of the search spaces to estimate the efficiency of the procedures.

The efficiency improvement of the proposed procedure is mainly related to the following four aspects. First, the focuses of the sequent structure in $G_{FS}$ reduce the search space. A focus in a sequent explicitly indicates the principal formula on which we apply inference rules. In $G_{FS}$, inference rules that have focuses in both premise sequents and goal sequents restrict the inference rules we can apply to the premise sequents. Every time we encounter a sequent during proof search, we try to apply inference rules to find a derivation tree. By the restriction, the number of applicable inference rules and the number of sequents generated by the procedure decreases. Without this restriction, we would consider all the possibilities of rule application on formulas.

Second, the bidirectional proof search with derived rules further reduces the search space. In the bidirectional proof search, we mainly traverse the search space in the backward direction. Backward proof search is inherently goal-oriented. Whereas a forward proof search expands the set of valid sequents to the saturation point, the backward search procedure traverses the sequents related to the goal sequent.

Third, using the derived rules instead of left-rules allows us to skip the intermediate sequents in focused threads. Derived rules from the forward proof search substitute for partial derivation trees of focused threads. By applying a derived rule, we skip the intermediate rule applications, go directly to the top of the partial derivation and continue the proof search on the premise sequents. The backward search with derived rules reduces the size of the search space by the number of intermediate sequents we skip.

Lastly, in addition to the reduction of the search space, the bookkeeping method improves the efficiency of processing a single sequent. The loop detection method involves expensive computation of comparison between sequents. To decide whether the given sequent is redundant, we compare the current sequent with all the sequents on the current branch. The computational complexity is $O(n)$ where $n$ is the depth of the current branch. In the bookkeeping method, we do not trace back the current branch; we only check the bookkeeping sets to detect loops. Managing sets reduces the computational complexity to $O(\log(n))$.

7 Conclusion

In this paper, we contributed to the study of IS5 in two aspects. First, we studied proof theory of IS5. We proposed a multi-contextual sequent and its sequent calculus. We proved cut-elimination of the proposed system. Second, we proposed the bidirectional proof search procedure for IS5. We showed that the bidirectional idea for IS4 [6] can be applied to IS5 with the multi-contextual sequent.

In the previous work on the bidirectional procedure for IS4, the target logic IS4 has no
frame in its sequents. In IS4, we need not consider other contexts that are accessible from the current context. The sequents of IS4 have only the current context of a given sequent. In IS5, the relation between worlds is an equivalence relation. Therefore, for a sequent in IS5, we keep all the contexts introduced to the frame and allow exchange of contexts. Our bidirectional procedure adopted the idea of derived rules and their use in the backward search. But, due to the multi-contextual sequent containing frame, we introduced the notion of context labeling and revised the definition of derived rule instances.

In qualitative comparison with other proof search strategies, the bidirectional procedure traverses a smaller search space. In terms of the time complexity of processing a sequent in the search space, the bidirectional procedure reduces time required for loop prevention by bookkeeping sets.

We have proved that our procedure is valid. The procedure distinguishes provability of a formula and finds a derivation tree for a valid formula in finite time. In conclusion, we designed a bidirectional procedure which is sound and complete with respect to IS5 logic and guaranteed to terminate in finite time.

In the future, we plan to optimize the implementation of the procedure and compare the implementation with other proof search procedures for IS5. We expect to quantitatively compare the efficiency of various proof search strategies including our bidirectional procedure. We may extend our study to the first-order logic of IS5. The introduction of first-order quantifiers will add another dimension to derived rule instances.

8 Coq Mechanization

We mechanized the proof of the cut-elimination theorem using proof assistant Coq. As the first step of mechanization, we wrote the inductive definitions of formula, context, frame and sequent in the formal language of Coq. Then, we mechanized the definition and proof of theorems in the order of dependency. For each theorem, we completed proof using tactics of Coq.

In the Coq programming, we rely heavily on induction strategies. Applying induction tactic on an inductive term allows us to analyze cases of the inductive definition. For each induction case, we used assumptions including inductive hypotheses to complete proof of the subgoals for cases. We used list data type of Coq standard library because we defined context as list of formulas and frame as list of contexts. list terms and operations are used many times for each induction case.

We developed ltac commands to automate repetitive tactic applications and solve the goal sequent. We will discuss the ltac commands we developed and how we exploited the commands to automated Coq programming.
8.1 From Definition of Formula to Cut-elimination Theorem

Formula is defined as a reflexive type because a formula with logical connective contains subformulas. The definition consists of terminal constructors and constructor of logical connectives which takes one or two subformulas. We decided to have nat type for atomic formulas. We label each atomic formula with a nat number and regard atomic formulas as a countable infinite set.

\[
\text{Inductive \text{formula} : \text{Set} :=}
\]
\[
| \text{f_top} : \text{formula} \\
| \text{f_p} : \text{nat} \to \text{formula} \\
| \text{f_bot} : \text{formula} \\
| \text{f_conj} : \text{formula} \to \text{formula} \to \text{formula} \\
| \text{f_disj} : \text{formula} \to \text{formula} \to \text{formula} \\
| \text{f_imp} : \text{formula} \to \text{formula} \to \text{formula} \\
| \text{f_box} : \text{formula} \to \text{formula} \\
| \text{f_dia} : \text{formula} \to \text{formula}.
\]

We defined context as list of formulas and frame as list of contexts. list is provided by the standard library of Coq. list resembles the list data structure which stores elements and keeps the order between the elements.

\[
\text{Definition context} := \text{list formula.}
\]
\[
\text{Definition frame} := \text{list context.}
\]

The inductive definition of sequent consists of the constructors each of which corresponds to an inference rule of G155. A \text{seq} term differs from the definition of sequent we have in the sequent calculus in that a \text{seq} term guarantees the provability. \text{seq} takes frame, global context, local context and a conclusion formula and return a Coq proposition. A \text{seq} term means that we have a derivation tree that consists of constructors of \text{seq}. That is, a \text{seq} term stands for a sequent and its derivation tree.

\[
\text{Inductive seq : frame} \to \text{context} \to \text{context} \to \text{formula} \to \text{Prop} :=
\]
\[
| \text{seq_top} : \forall \text{f:frame} \ (\text{g l:context}), \text{seq f g l f_top} \\
| \text{seq_id1} : \forall \text{f:frame} \ (\text{g l1 l2:context}) \ (\text{p:nat}), \text{seq f g (l1 ++ ((f_p p) :: l2)) (f_p p)} \\
| \text{seq_conj_r} : \forall \text{f:frame} \ (\text{g l:context}) \ (\text{A B:formula}), \text{seq f g l A} \to \text{seq f g l B} \\
| \text{seq_box_l1} : \forall \text{f:frame} \ (\text{g l1 l2:context}) \ (\text{A C:formula}), \text{seq f (A::g) (l1++l2) C} \to \text{seq f g (l1++((f_box A)::l2)) C}
\]

24
Here, we presented some of the constructors. Each constructor play the role of a proposition in Coq. For example, seq_conj_r takes two premise sequents terms and implies the provability of the conclusion sequent. That is what an inference rule implies in sequent calculus systems.

In the definition of constructor seq_imp_l3, the proposition has forall quantifiers and takes two frames, four contexts and three formulas. The constructor takes two frames to split the frame that has the context of interest in the middle. The frame of the conclusion sequent has the context of interest p1++((f_imp A B)::p2) which is sandwiched by two lists f1 and f2. Together, f1 and f2 compose the frame. We split the frame to generalize the inference rule. The rule with a split frame can take any context in the frame as a context of interest. We see the same pattern of splitting lists in contexts of seq terms.

We also have the inductive definition of sequent with size. The constructors of the sized definition correspond to their counterparts in seq definition. The constructors take another argument typed nat which stands for the size of the derivation tree.

```
Inductive seqn : nat -> frame -> context -> context -> formula -> Prop :=
| seqn_top : forall (f:frame) (g l:context),
  seqn 0 f g l f_top
... | seqn_disj_l2b : forall (n1 n2:nat) (f1 f2:frame) (p g1 g2 l:context) (A B C:formula),
  seqn n1 (f1 ++ ((A :: p) :: f2)) (g1 ++ ((f_disj A B)::g2)) l C ->
  seqn n2 (f1 ++ ((B :: p) :: f2)) (g1 ++ ((f_disj A B)::g2)) l C ->
  seqn (n1+n2+1) (f1 ++ (p :: f2)) (g1 ++ ((f_disj A B)::g2)) l C
...```

For the terminal constructors, we have the derivation tree size of zero 0. For the non-terminal constructors, the size of the derivation tree is equal to sum of the sizes of premise sequents plus one.

We have another definition seqle of sequent with size bounded by the nat typed number. A seqle term with n means that the derivation tree for the sequent is less than or equal to n. We mainly use seqn and seqle to define size-preserving theorems.

```
Theorem weakening_seqn_l13:
  forall (n:nat),
  (forall (f:frame) (g l:context) (A C:formula),
   seqn n f g l C -> seqn n f g (A::l) C) /
  (forall (f:frame) (g l p:context) (A C:formula),
   seqn n (p::f) g l C -> seqn n ((A::p)::f) g l C).
Proof.
  eapply complete_nat_ind.
  split; intros.
  inversion H; subst;
  tryall_seqn_constr auto auto;
  contradict_plus_eq_0.
... Qed.

25
Theorem weakening_seqn_l13 is the very first structural property that we proved using Coq. As the definition of theorem implies, a sequent with an additional formula is guaranteed to have a derivation tree provided that we have a derivation tree of the original sequent. \( A \) is the new formula added to either local context or another context in the frame. We defined the theorem to claim the weakening property on both local context and frame simultaneously. The proof of the weakening property on local context depends on the induction hypothesis of the weakening property on a context in frame. Likewise, the proof of the property on frame depends on the induction hypothesis of the property on local context.

In the proof of theorems that we mechanized, we started with induction on the sizes of sequents and then we applied case analysis on the derivation trees. To initiate case analysis on the derivation trees, we used inversion tactic on the assumption of seq. In the usual proof scenario of induction and case analysis, we have the premise sequents of the bottom most rule application. We can use induction hypotheses by the induction on proof sizes. If we have a sequent with a smaller size than the current size \( n \), then we can apply the induction hypotheses on the sequent.

\[
\begin{align*}
H_h : & n = n_1 + n_2 + 1 \\
H_0 : & \text{seqn } n_1 (f_1 ++ l :: f_2) (g_1 ++ f \_imp A_0 B :: g_2) p A_0 \\
H_1 : & \text{seqn } n_2 (f_1 ++ (B :: p) :: f_2) (g_1 ++ f \_imp A_0 B :: g_2) l C \\
IH1 : & \forall (m : \text{nat}) (f : \text{frame}) (g l : \text{context}) (A C : \text{formula}), \\
& m < n_1 + n_2 + 1 \rightarrow \text{seqn } m f g l C \rightarrow \text{seqn } m f g (A :: l) C \\
IH2 : & \forall (m : \text{nat}) (f : \text{frame}) (g l p : \text{context}) (A C : \text{formula}), \\
& m < n_1 + n_2 + 1 \rightarrow \text{seqn } m (p :: f) g l C \rightarrow \text{seqn } m ((A :: p) :: f) g l C \\
H : & \text{seqn } (n_1 + n_2 + 1) (f_1 ++ p :: f_2) (g_1 ++ f \_imp A_0 B :: g_2) l C \\
\end{align*}
\]

The above is the goal window showing the current goal and the available assumptions. At the moment, we try to prove the goal sequent with an extra formula \( A \) on the local context. In the current case, the premise sequent \( H \) has the derivation tree which has the application of seq_imp_l2b at the bottom. \( H_0 \) and \( H_1 \) are the premise sequents of the last rule application. IH1 and IH2 are the induction hypothesis that we can exploit. If we provide a sequent with size less than \( n_1+n_2+1 \), then we get another sequent with an extra formula \( A \) added to the local context.

For the current subcase, we first apply seq_imp_l2b on the goal sequent. Then, we apply induction hypothesis on a context. The resulting goal sequents are equivalent to \( H_0 \) and \( H_1 \), respectively.

Most of the cases follow the similar course of Coq programming, applying inference rules and then applying induction hypotheses. But, depending on associativities of list operations, the proof may require to apply rewrite tactics many times. Since seq has 30 constructors, we have many goals after we apply induction or case analysis. In the next section, we explain ltac automation we developed to reduce the amount of Coq programming.
Theorem cut_elimination:
  forall (A:formula) (n m:nat),
  (forall (f:frame) (g l:context) (C:formula),
   seqle n f g l A -> seqle m f g (A::l) C -> seq f g l C) /
  (forall (f:frame) (g l:context) (C:formula),
   seqle n (l::f) g [] A -> seqle m f (A::g) l C -> seq f g l C) /
  (forall (f:frame) (g l p:context) (C:formula),
   seqle n (l::f) g p A -> seqle m ((A::p)::f) g l C -> seq (p::f) g l C).
Proof.

(* induction on cut-formula A *)
induction A.

Case "f_top".
apply cut_top.
...
Qed.

(* for an induction case where A = A1 disj A2 *)
Theorem cut_disj :
forall (A1 A2:formula),
  (forall n m : nat,
   (forall (f : frame) (g l : context) (C : formula),
    seqle n f g l A1 -> seqle m f g (A1 :: l) C -> seq f g l C) /
   ...)
Proof.

(* induction on the proof size of a premise *)
induction n.
...
SCase "n=S n".
(* nested induction on the proof size of the other premise *)
induction m.
...
SSCase "m=S m".
...
SSSSCase "cut2".

(* case analysis on the derivation tree *)
inversion H0; subst;
repeat list_eq_mismatch_analysis;
try contradict_list_nil;
try solve [try_seq];
try solve [apply_IHall1 IHm1 IHm2 IHm3].
SSSSSSCase "disj_12a".
The cut-elimination theorem consisting of three propositions was defined simultaneously. We started the proof by induction $A$ which analyzes the structure of cut-formula $A$ and provides induction hypotheses. We split the proof of the theorem into multiple theorems each of which correspond to each case of formula constructors. For each case, we apply induction $n$, induction $m$, inversion $H_0$ and inversion $H$ in turns. Using all the induction hypotheses by the applications of induction, we prove each subcase one by one. As we did in the proof of the weakening property, we apply constructors of seq and then apply induction hypotheses to reach sequents of the assumptions that we already have. The process of Coq programming resembles the proof search in that we keep applying rules and hypotheses to complete the proof of branches.

8.2 ltac automation

We developed ltac commands to automate sequences of repetitive programming that appear many times in the proof. The ltac language of Coq provides functionalities such as pattern matching match [term] with, repetition of tatics repeat, trial and backtracking try, solve, first, fail.

Using the pattern matching of match [term or context or goal] with, we can match terms, the current goal or the context of assumptions. We can simulate an if conditional statements using the pattern matching features. The repetition of tatics, trial and backtracking of tatics allow us to devise functions which tries to prove the current goal and fall back when the trial fails. For example, first [tac1; tac2; ...; tacn] applies tac1 and if the application succeed then stop. Otherwise, the command applies tac2 and stops if the application succeed. first stops at the first command that is applied successfully.

We developed list_norm and rewrite_list commands. list_norm transform a list term that is mingled with cons :: and concat ++ into a normalized form. rewrite_list command takes a target term and a scheme. Then, the command change the associativities of operators according to the scheme. For example, assume that we have the following goal and we want to apply rule seqle_disj_l2b. We need to transform the frame into the form where we have the context of interest 1 in the middle. We need to transform the global context to have the principal formula f_disj A B sandwiched by two lists.
First, we apply \texttt{norm_list} to normalize all the list terms. The resulting goal sequent has the list terms with the operations are applied from right to left.

\begin{verbatim}
....
seqle (n1+n2+1) (l::f2) ((f_disj A B :: g1) ++ f_imp A0 B0 :: g0) p C

norm_list.

....
seqle (n1+n2+1) (l::f2) (f_disj A B :: g1 ++ f_imp A0 B0 :: g0) p C
\end{verbatim}

Second, we apply \texttt{rewrite_list} commands to make the target term into the desired form. \([1;0;1]\) means that we want the term in the structure of \texttt{context ++ formula :: context} or \texttt{frame ++ context :: frame}.

\begin{verbatim}
....
seqle (n1+n2+1) (l::f2) (f_disj A B :: g1 ++ f_imp A0 B0 :: g0) p C

rewrite_list (l::f2) [1;0;1].
rewrite_list ([++]++l::f2) ([++] ++ f_disj A B :: g1 ++ f_imp A0 B0 :: g0) [1;0;1].

....
seqle (n1+n2+1) ([++]++l::f2) ([++] ++ f_disj A B :: g1 ++ f_imp A0 B0 :: g0) p C
\end{verbatim}

The resulting sequent has the desired form that we can apply \texttt{seqle_disj_l2b} right away.

We developed \texttt{list_eq_mismatch_analysis} command which analyzes an assumption that two lists are equal. The command enumerates all the possibilities we have to consider. Following is an equality between two lists with terms are concatenated and appended.

\begin{verbatim}
H: l1 ++ a :: l2 = b :: l3 ++ c :: l4

list_eq_mismatch_analysis

.... (l1 ++ a :: l2) ...
\end{verbatim}

Above is one of the subgoals that \texttt{list_eq_mismatch_analysis} generated. Until we analyze each possible structure of the list, we do not have both \texttt{a} and \texttt{b} in the goal
sequent. Now that we have the structure of the target term, we can continue proving using \(a\) and \(b\) in the goal sequent. In another possible structure, we may have assumptions \(a=b\), \(l_1=l_3\) and \(l_2=l_4\). Solving all the subgoals by the command, we can complete the proof.

Based on the \texttt{ltac} commands, we developed advanced commands such as \texttt{try_seqle}, \texttt{apply_structural} and \texttt{contract_list}. \texttt{try_seqle} automatically transforms list structures of a sequent and apply an inference rule on the sequent. \texttt{apply_structural} takes an assumption of sequent to prove the goal sequent. The command applies structural properties of the sequent calculus and make the current goal sequent equal to the designated assumption. \texttt{contract_list} automatically detects absurd assumptions in the current context and applies \texttt{inversion} to prove the current goal.

With the \texttt{ltac} commands, we dramatically reduced the number of lines in Coq scripts. Instead of typing all the repetitive tactics for subgoals, we relied on mechanical trial by the commands.

\begin{verbatim}
H7 : seqle x (f1 ++ (f_disj A B :: p) :: f2) (g1 ++ f_disj A0 B0 :: g0) l C

apply_structural H7

# the goal solved (exchange in frame, weakening in a context of frame)
\end{verbatim}
Appendix A  Sequent Calculus $G_{IS5}$ with Proof Size

![Appendix A Sequent Calculus $G_{IS5}$ with Proof Size](Image of the sequent calculus)

Figure 7: Sequent calculus $G_{IS5}$ with derivation tree size annotated

Appendix B  Inversion Properties of Inference Rules in $G_{IS5}$

**Proposition B.1** (Inversion property $\land$)

- $\forall L_{local}$ If $n \succeq G \vdash \Delta \vdash \Gamma, A \land B \vdash C$ then $n \geq G \vdash \Delta \vdash \Gamma, A, B \vdash C$.
- $\forall L_{global}$ If $n \succeq G \vdash \Delta, A \land B \vdash \Gamma \vdash C$ then $n \geq G \vdash \Delta, A, B \vdash \Gamma \vdash C$.
- $\forall L_{frame}$ If $n \succeq G; \Gamma', A \land B \vdash \Gamma \vdash C$ then $n \geq G; \Gamma', A, B \vdash \Delta \vdash \Gamma \vdash C$.
- $\forall R$ If $n \succeq G \vdash \Delta \vdash \Gamma \vdash A \land B$ then $n \geq G \vdash \Delta \vdash \Gamma \vdash A$ and $n \geq G \vdash \Delta \vdash \Gamma \vdash B$.

**Proposition B.2** (Inversion property $\lor$)
• ($\lor R$) If $n \vdash G \vdash \Delta \vdash \Gamma \vdash A \lor B$ then $n \geq G \vdash \Delta \vdash \Gamma, A \vdash B$.

Proposition B.3 (Inversion property $\square$)

• ($\square_{\text{local}}$) If $n \simeq G \vdash \Delta \vdash \Gamma, \square A \vdash C$ then $n \geq G \vdash \Delta, A \vdash \Gamma \vdash C$.
• ($\square_{\text{global}}$) If $n \simeq G \vdash \Delta, \square A \vdash \Gamma \vdash C$ then $n \geq G \vdash \Delta, A \vdash \Gamma \vdash C$.
• ($\square_{\text{frame}}$) If $n \simeq G; \Gamma', \square A \vdash \Delta \vdash \Gamma \vdash C$ then $n \geq G; \Gamma' \vdash \Delta, A \vdash \Gamma \vdash C$.
• (R) If $n \simeq G \vdash \Delta \vdash \Gamma \vdash \square A$ then $n \geq G; \Gamma \vdash \Delta \vdash \cdot \vdash A$.

Proposition B.4 (Inversion property $\diamond$)

• ($\diamond_{\text{local}}$) If $n \simeq G \vdash \Delta \vdash \Gamma, \diamond A \vdash C$ then $n \geq G; A \vdash \Delta \vdash \Gamma \vdash C$.
• ($\diamond_{\text{global}}$) If $n \simeq G \vdash \Delta, \diamond A \vdash \Gamma \vdash C$ then $n \geq G; A \vdash \Delta \vdash \Gamma \vdash C$.
• ($\diamond_{\text{frame}}$) If $n \simeq G; \Gamma', \diamond A \vdash \Delta \vdash \Gamma \vdash C$ then $n \geq G; \Gamma' \vdash A \vdash \Delta \vdash \Gamma \vdash C$.

Appendix C  Auxiliary Properties of $G_{\text{IS5}}$

Proposition C.1 (Property of $\lor$)

• If $n \simeq G \vdash \Delta \vdash \Gamma, A \lor B \vdash C$ then
  $n \geq G \vdash \Delta \vdash \Gamma, A \vdash \Gamma \vdash C$ and $n \geq G \vdash \Delta \vdash \Gamma, B \vdash \Gamma \vdash C$.
• If $n \simeq G \vdash \Delta, A \lor B \vdash \Gamma \vdash C$ then
  $n \geq G \vdash \Delta, A \vdash \Gamma \vdash C$ and $n \geq G \vdash \Delta, B \vdash \Gamma \vdash C$.
• If $n \simeq G; \Gamma', A \lor B \vdash \Delta \vdash \Gamma \vdash C$ then
  $n \geq G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C$ and $n \geq G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C$.

Proposition C.2 (Property of $\supset$)

• If $n \simeq G \vdash \Delta \vdash \Gamma, A \supset B \vdash C$ then
  $n \geq G \vdash \Delta \vdash \Gamma, B \vdash C$.
• If $n \simeq G; \Gamma', A \supset B \vdash \Delta \vdash \Gamma \vdash C$ then
  $n \geq G; \Gamma', B \vdash \Delta \vdash \Gamma \vdash C$. 

32
Appendix D  Soundness and Completeness of $G_{IS5}$ with respect to $G$

Theorem D.1 (Completeness of $G_{IS5}$ with respect to $G$)
If $G \vdash \Gamma \vdash G C$ in $G$ then $G \vdash \cdot \vdash \Gamma \vdash C$ in $G_{IS5}$.

Proof By induction on the derivation tree in $G$. We present some of the induction cases.

Case $G \vdash \Gamma, p \vdash_G p$  $[Id]_G$
$G \vdash \cdot \vdash \Gamma, p \vdash p$  by $Id_{local}$

Case $G; \Gamma', A, B \vdash \cdot \vdash_G C$  $[\wedge LL]_G$

(1) $G; \Gamma', A, B \vdash \cdot \vdash \Gamma \vdash C$  by IH

(2) $G; \Gamma', A \wedge B \vdash \cdot \vdash \Gamma \vdash C$  by $\wedge frame$ with (1)
Theorem D.2 (Soundness of $G_{1S5}$ with respect to $G$)

If $G \vdash \Delta \vdash \Gamma \vdash C$ in $G_{1S5}$ then $G \vdash \Box \Delta, \Gamma \vdash G C$ in $G$.

(where $\Delta = A_1, A_2, ..., A_i$ and $\Box \Delta = \Box A_1, \Box A_2, ..., \Box A_i$)

Proof By induction on the derivation tree in $G_{1S5}$. We present some of the induction cases.

Case $G; \Gamma', A \vdash G C$  
Case $G; \Gamma', B \vdash G C$  
\[ G; \Gamma', A \lor B \vdash G C \]  
$[\lor LL]_G$

(1) $G; \Gamma', A \vdash \cdot \vdash \Gamma \vdash C$  
by IH

(2) $G; \Gamma', B \vdash \cdot \vdash \Gamma \vdash C$  
by IH

(3) $G; \Gamma', A \lor B \vdash \cdot \vdash \Gamma \vdash C$  
by $\lor L_{frame}$ with (1,2)

Case $G; \Gamma', A \vdash \Box A \vdash G C$  
\[ G; \Gamma' \vdash \Box A \vdash G C \]  
$[\square^2 L]_G$

(1) $G; \Gamma', A \vdash \cdot \vdash \Box A \vdash C$  
by IH

(2) $G; \Gamma', A \vdash A \vdash \Gamma \vdash C$  
by inversion property $\Box L_{local}$ on $A$

(3) $G; \Gamma' \vdash A \vdash \Gamma \vdash C$  
by contraction property on $A$

(4) $G; \Gamma' \vdash A \vdash G C$  
by $\Box L_{local}$ with (3)
Case \( G; \Gamma', A \vdash \Delta, A \lor B \vdash \Gamma \vdash C \quad G; \Gamma', B \vdash \Delta, A \lor B \vdash \Gamma \vdash C \quad \exists L_b^{global} \)

(1) \( G; \Gamma', A \vdash \square \Delta, \square (A \lor B), \Gamma \vdash G C \) \quad by IH

(2) \( G; \Gamma', B \vdash \square \Delta, \square (A \lor B), \Gamma \vdash G C \) \quad by IH

(3) \( G; \Gamma', A \lor B \vdash \square \Delta, \square (A \lor B), \Gamma \vdash G C \) \quad by \([\lor LL]_G\) with (1,2)

(4) \( G; \Gamma' \vdash \square \Delta, \square (A \lor B), \Gamma \vdash G C \) \quad by \([\square L]_G\) with (3)

Case \( G \vdash \Delta, A \supset B \vdash \Gamma \vdash A \quad G \vdash \Delta, A \supset B \vdash \Gamma \vdash C \quad \exists L_a^{global} \)

(1) \( G \vdash \square \Delta, \square (A \supset B), \Gamma \vdash G A \) \quad by IH

(2) \( G \vdash \square \Delta, \square (A \supset B), \Gamma, B \vdash G C \) \quad by IH

(3) \( G \vdash \square \Delta, \square (A \supset B), \Gamma, A \supset B \vdash G A \) \quad by weakening property with (1)

(4) \( G \vdash \square \Delta, \square (A \supset B), \Gamma, A \supset B \vdash G C \) \quad by \([\supset L]_G\) with (2,3)

(5) \( G \vdash \square \Delta, \square (A \supset B), \Gamma \vdash G C \) \quad by \([\square L]_G\) with (4)

Case \( G; \Gamma' \vdash \Delta, A \vdash \Gamma \vdash C \quad \exists L^{frame} \)

(1) \( G; \Gamma' \vdash \square \Delta, \square A, \Gamma \vdash G C \) \quad by IH

(2) \( G; \Gamma', \square A \vdash \square \Delta, \Gamma \vdash G C \) \quad by lemma D.3 with (1)

Case \( G; A \vdash \Delta \vdash \Gamma \vdash C \quad \exists L^{global} \)

(1) \( G; A \vdash \square \Delta, \Gamma \vdash G C \) \quad by IH

(2) \( G \vdash \square \Delta, \Gamma, \diamond A \vdash G C \) \quad by \([\diamond L]_G\) with (1)

(3) \( G \vdash \square \Delta, \square \diamond A, \Gamma \vdash G C \) \quad by axiom (7) of IS5 [7]
Lemma D.3

1. If $G; \Gamma \vdash \Box \Delta \vdash_G C$ then $G; \Gamma ', \Box \Delta \vdash \Gamma \vdash_G C$.
2. If $G; \Gamma ', \Box \Delta \vdash \Gamma \vdash_G C$ then If $G; \Gamma ' \vdash \Gamma , \Box \Delta \vdash_G C$.

Appendix E  Soundness and Completeness of $G_{IS5}^F$ with respect to $G_{IS5}$

![Figure 9: Sequent calculus $G_{IS5}^I$]

We introduce two intermediate sequent calculus to prove soundness and completeness of $G_{IS5}^F$ with respect to $G_{IS5}$. In the first intermediate sequent calculus $G_{IS5}^I$, the principal formula of an inference rule remains in the premise sequents. $[\land L]$ rules of $G_{IS5}^I$ has only one subformula of conjunction in its premise. That is, the left-rules select one of $A$ and $B$ from $A \land B$. We start with the soundness and completeness of $G_{IS5}^I$ with respect to $G_{IS5}$.

**Theorem E.1** (Soundness of $G_{IS5}^I$ with respect to $G_{IS5}$)

If $G \vdash \Delta \vdash \Gamma \vdash_I C$ in $G_{IS5}^I$ then $G \vdash \Delta \vdash \Gamma \vdash C$ in $G_{IS5}$. 

36
Proof By induction on the derivation tree in $G_{IS5}^t$. We present some of the induction cases that are interesting.

Case $G \vdash \Delta \vdash \Gamma, A \land B, A \vdash I C$ by IH

(1) $G \vdash \Delta \vdash \Gamma, A \land B, A \vdash C$

(2) $G \vdash \Delta \vdash \Gamma, A, B, A \vdash C$

(3) $G \vdash \Delta \vdash \Gamma, A, B \vdash C$

(4) $G \vdash \Delta \vdash \Gamma, A \land B \vdash C$

Case $G \vdash \Delta, A \land B \vdash \Gamma, A \vdash I C$ by $\land_L^{local}$

(1) $G \vdash \Delta, A \land B \vdash \Gamma, A \vdash C$

(2) $G \vdash \Delta, A, B \vdash \Gamma, A \vdash C$

(3) $G \vdash \Delta, A, B \vdash \Gamma \vdash C$

(4) $G \vdash \Delta, A \land B \vdash \Gamma \vdash C$

Case $G; \Gamma', A \lor B, A \vdash \Delta \vdash \Gamma \vdash I C$ by $\lor_L^{frame}$

(1) $G; \Gamma', A \lor B, A \vdash \Delta \vdash \Gamma \vdash C$

(2) $G; \Gamma', A, A \vdash \Delta \vdash \Gamma \vdash C$

(3) $G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C$

(4) $G; \Gamma', B \vdash \Delta \vdash \Gamma \vdash C$

(5) $G; \Gamma', A \lor B \vdash \Delta \vdash \Gamma \vdash C$

Case $G \vdash \Delta, A \vdash \Gamma, \Box A \vdash I C$ by $\Box_L^{local}$

(1) $G \vdash \Delta, A \vdash \Gamma, \Box A \vdash C$

(2) $G \vdash \Delta, A, A \vdash \Gamma \vdash C$

by the inversion property of $\Box_L^{local}$
(3) \( G \vdash \Delta, A \vdash \Gamma \vdash C \) by the contraction property on \( A \)

(4) \( G \vdash \Delta \vdash \Gamma, \Box A \vdash C \) by \( \Box L_{\text{local}} \)

**Theorem E.2** (Completeness of \( G_{I55}^l \) with respect to \( G_{I55} \))

If \( G \vdash \Delta \vdash \Gamma \vdash C \) in \( G_{I55} \) then \( G \vdash \Delta \vdash \Gamma \vdash_I C \) in \( G_{I55}^l \).

**Proof** We give a transition function which takes a derivation tree of \( G_{I55} \) and generate a derivaton tree of \( G_{I55}^l \). By the function, we can find a derivation tree for \( G \vdash \Delta \vdash \Gamma \vdash C \) in \( G_{I55} \) whenever we have the derivation tree for \( G \vdash \Delta \vdash \Gamma \vdash_I C \) in \( G_{I55}^l \). We give some of the cases that are interesting.

\[
f \left( \frac{G \vdash \Delta \vdash \Gamma, A, B \vdash C}{G \vdash \Delta \vdash \Gamma, A \land B \vdash C \land L_{\text{local}}} \right) = \frac{f \left( \frac{G \vdash \Delta \vdash \Gamma, A \land B, A, B \vdash C}{G \vdash \Delta \vdash \Gamma, A \land B \vdash \Gamma \vdash C \land L_{\text{local}}} \right)}{f \left( \frac{G \vdash \Delta \vdash \Gamma, A \land B, A, B \vdash C}{G \vdash \Delta \vdash \Gamma, A \land B \vdash \Gamma \vdash C \land L_{\text{local}}} \right)}
\]

\((G \vdash \Delta \vdash \Gamma, A \land B, A, B \vdash C \) by the size-preserving weakening property with \( G \vdash \Delta \vdash \Gamma, A, B \vdash C \))

\[
f \left( \frac{G \vdash \Delta, A, B \vdash \Gamma, A_1 \vdash C}{G \vdash \Delta, A \land B \vdash \Gamma \vdash C \land L_{\text{global}}} \right) = \frac{f \left( \frac{G \vdash \Delta, A, B \vdash \Gamma, A_1 \vdash C}{G \vdash \Delta, A \land B \vdash \Gamma \vdash C \land L_{\text{global}}} \right)}{f \left( \frac{G \vdash \Delta, A, B \vdash \Gamma, A_1 \vdash C}{G \vdash \Delta, A \land B \vdash \Gamma \vdash C \land L_{\text{global}}} \right)}
\]

(Proof size of \( G \vdash \Delta, A \land B \vdash \Gamma, A_1 \vdash C \) is less than proof size of \( G \vdash \Delta, A \land B \vdash \Gamma \vdash C \)).

\[
f \left( \frac{G' \vdash \Delta', A, B \vdash \Gamma', A_1 \vdash C}{G' \vdash \Delta', A, B \vdash \Gamma' \vdash C \land L_{\text{global}}} \right) = \frac{f \left( \frac{G' \vdash \Delta', A, B \vdash \Gamma', A_1 \vdash C'}{G' \vdash \Delta', A, B \vdash \Gamma' \vdash C \land L_{\text{global}}} \right)}{f \left( \frac{G' \vdash \Delta', A, B \vdash \Gamma', A_1 \vdash C'}{G' \vdash \Delta', A, B \vdash \Gamma' \vdash C \land L_{\text{global}}} \right)}
\]

(Proof size of \( G \vdash \Delta, A \land B \vdash \Gamma, A_1 \vdash C \) is less than proof size of \( G \vdash \Delta, A \land B \vdash \Gamma \vdash C \). If there are rule applications between the application of \( \land L_{\text{global}} \) and the left-rule application on \( A \), then the function delays the application of \( \land L_{\text{global}} \).)
\[
f \left( \frac{G \vdash \Delta, A \vdash \Gamma \vdash C}{G \vdash \Delta, \Box A \vdash \Gamma \vdash C} \Box \text{local} \right) = f \left( \frac{G \vdash \Delta, A \vdash \Gamma \vdash C}{G \vdash \Delta, \Box A \vdash \Gamma \vdash C} \Box \text{global} \right)
\]

\((G \vdash \Delta, \Box A, A \vdash \Gamma \vdash C) \text{ by the size-preserving weakening property}\)

Figure 10: Sequent calculus \(G_{F5N}\)
Theorem E.3 (Soundness of $G^{FN}_{I55}$ with respect to $G^{I}_{I55}$)

1. If $G \vdash \Delta \vdash \Gamma \rightarrow C$ in $G^{FN}_{I55}$ then $G \vdash \Delta \vdash \Gamma \vdash I \rightarrow C$ in $G^{I}_{I55}$.

2. If $G \vdash \Delta \vdash \Gamma \triangleright C \rightarrow A$ in $G^{FN}_{I55}$ then $G \vdash \Delta \vdash \Gamma, A \vdash I \rightarrow C$ in $G^{I}_{I55}$.

3. If $G \vdash \Delta \vdash \Gamma \triangleright C \rightarrow A$ in $G^{FN}_{I55}$ then $G \vdash \Delta \vdash \Gamma, A \vdash I \rightarrow C$ in $G^{I}_{I55}$.

4. If $G; \Gamma' \triangleright C \vdash \Delta \vdash \Gamma \rightarrow A$ in $G^{FN}_{I55}$ then $G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash I \rightarrow C$ in $G^{I}_{I55}$.

5. If $G; \Gamma' \triangleright C \vdash \Delta \vdash \Gamma \rightarrow A$ in $G^{FN}_{I55}$ then $G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash I \rightarrow C$ in $G^{I}_{I55}$.

Proof By induction on the derivation tree in $G^{FN}_{I55}$. We present some of the induction cases that are interesting.

Case \( \frac{G \vdash \Delta, A \vdash \Gamma \triangleright C \rightarrow A}{G \vdash \Delta, A \vdash \Gamma \rightarrow C} \) \( \text{ch}_a^{\text{global}} \)

(1) $G \vdash \Delta, A \vdash \Gamma, A \vdash I \rightarrow C$

by IH

(2) $G \vdash \Delta, A \vdash \Gamma \vdash I \rightarrow C$

by the contraction property on $A$

Case \( \frac{G \vdash \Delta \vdash \Gamma \triangleright C \rightarrow A_i}{G \vdash \Delta \vdash \Gamma \vdash C \rightarrow A_1 \wedge A_2} \) \( \land L^{\text{local}} \)

(1) $G \vdash \Delta \vdash \Gamma, A_i \vdash I \rightarrow C$

by IH

(2) $G \vdash \Delta \vdash \Gamma, A_1 \wedge A_2, A_i \vdash I \rightarrow C$

by the weakening property

(3) $G \vdash \Delta \vdash \Gamma, A_1 \wedge A_2 \vdash I \rightarrow C$

by $\land L^{\text{local}}$ with (2)

Case \( \frac{G; \Gamma' \vdash \Delta, A \vdash \Gamma \rightarrow C}{G; \Gamma' \triangleright C \vdash \Delta \vdash \Gamma \rightarrow \Box A} \) \( \Box L^{\text{frame}} \)

(1) $G; \Gamma' \vdash \Delta, A \vdash \Gamma \vdash I \rightarrow C$

by IH

(2) $G; \Gamma', \Box A \vdash \Delta, A \vdash \Gamma \vdash I \rightarrow C$

by the weakening property

(3) $G; \Gamma', \Box A \vdash \Delta \vdash \Gamma \vdash I \rightarrow C$

by $\Box L^{\text{frame}}$ with (2)

Case \( \frac{G; A \vdash \Delta \vdash \Gamma \rightarrow C}{G \vdash \Delta \vdash \Gamma \triangleright C \rightarrow \Diamond A} \) \( \Diamond L^{\text{global}} \)

(1) $G; A \vdash \Delta \vdash \Gamma \vdash I \rightarrow C$

by IH
(2) \( G; A \vdash \Delta \vdash \Gamma, \Diamond A \vdash I \ C \) by the weakening property

(3) \( G \vdash \Delta \vdash \Gamma, \Diamond A \vdash I \ C \) by \( \Diamond L_{local} \) with (2)

**Theorem E.4** (Completeness of \( G_{IS5}^{FN} \) with respect to \( G_{IS5} \))

If \( G \vdash \Delta \vdash \Gamma \vdash I \ C \) in \( G_{IS5} \) then \( G \vdash \Delta \vdash \Gamma \rightarrow C \) in \( G_{IS5}^{FN} \).

**Proof** The intermediate sequent calculus \( G_{IS5}^{FN} \) has focuses in the inference rules whereas \( G_{IS5} \) does not present a focus on the principal formula. Another difference between \( G_{IS5}^{FN} \) and \( G_{IS5} \) is that the left-rules of \( G_{IS5}^{FN} \) preserve the principal formula as we follow up to the premise sequents. The derivation trees on the premise sequents may use the principal formula again on branches.

Since \( G_{IS5}^{FN} \) has the restriction of rule applications by focuses, a derivation tree in \( G_{IS5}^{FN} \) cannot be interpreted as a valid derivation tree in \( G_{IS5}^{FN} \). For example, the following derivation tree in \( G_{IS5}^{FN} \)

\[
\begin{align*}
G \vdash \Delta \vdash \Gamma, A \land B, B \supset C, B \rightarrow B \\
\text{id}_{local} \\
G \vdash \Delta \vdash \Gamma, A \land B, B \supset C, B \rightarrow C \\
\text{id}_{local} \\
G \vdash \Delta \vdash \Gamma, A \land B, B \supset C \\
\text{ch}_{local} (invalid)
\end{align*}
\]

violates the focus condition when we directly interpret the derivation tree in \( G_{IS5}^{FN} \). Transforming the derivation tree of \( G_{IS5} \) into \( G_{IS5}^{FN} \) results in an invalid derivation tree as the following:

\[
\begin{align*}
G \vdash \Delta \vdash \Gamma, A \land B, B \supset C, B \rightarrow B \\
\text{id}_{local} \\
G \vdash \Delta \vdash \Gamma, A \land B, B \supset C, B \rightarrow C \\
\text{ch}_{local} (invalid)
\end{align*}
\]

The invalid rule application ignores the current focus of the conclusion sequent. According to the focus condition, we must take \( B \) as the principal formula instead of \( B \supset C \) at the invalid rule application. Here, we present a function \( f \) which takes a derivation tree of \( G_{IS5} \) and transforms it into a valid derivation tree in \( G_{IS5}^{FN} \).

Before we move on to the definition of function \( f \), we give an intuitive description of the transformation. A violation of the focus condition actually means a broken focused
The focused thread starting with $A_1$ stops at $A_i$ and reassign its focus on $B_1$. Without the invalid change of the focus, the focused thread would apply a rule on $A_i$ and continue the focused thread.

Given a derivation tree, we locate the bottom most rule application that breaks a focused thread. Here, in this example, the rule application (1) is such a rule application. Then, we analyze the partial derivation tree $E$ and locate the bottom most rule application that is applied on $A_i$:

$$
E = \begin{array}{c}
D \\
G' \vdash \Delta' \vdash \Gamma' \rhd C' \to A_{i+1} \qquad (A, i) \\
G' \vdash \Delta' \vdash \Gamma' \rhd C' \to A_i \qquad ch_{\text{local}} (1) \\
G \vdash \Delta \vdash \Gamma \rhd C \to B_2 \qquad (B, 1)
\end{array}
$$

If there is no rule application on $A_i$ in the partial derivation tree $E$, we can simply eliminate the broken focused thread $G$. If a left-rule application focuses on $A_j$ and $j < i$, then we can safely skip such left-rule applications and continue with a rule application on $A_i$. Otherwise, if we locate a rule application on $A_i$, we modify the derivation so that the broken focused thread $G$ starting with $A_1$ is placed right below $G' \vdash \Delta' \vdash \Gamma', A_i \rhd C' \to A_i$ in the branch $B$. That is, we postpone the rule applications of the broken focused thread until we actually need the formula $A_i$. The resulting
derivation tree would be as the following:

\[
\begin{align*}
G' \vdash \Delta' \vdash \Gamma' \triangleright C' &\rightarrow A_{i+1} \\
G' \vdash \Delta' \vdash \Gamma', A_1 \triangleright C' &\rightarrow A_i \ \ (A, i) \\
\vdash G \\
G' \vdash \Delta' \vdash \Gamma', A_1 \triangleright C' &\rightarrow A_1 \\
G' \vdash \Delta' \vdash \Gamma', A_1 \rightarrow C' &\rightarrow C'' \ \ \text{ch}^{\text{local}} \\
\{ & \text{branch B} \\
\} \\
G \vdash \Delta \vdash \Gamma, A_1 \triangleright C &\rightarrow B_2 \\
G \vdash \Delta \vdash \Gamma, A_1 \triangleright C &\rightarrow B_1 \ \ (B, 1) \ \ \text{ch}^{\text{local}} \\
\end{align*}
\]

Since the inference rules of \( G^{FN}_{I5S} \) do not remove formulas from any context, the sequent on the branch \( \mathcal{B} \) contains \( A_1 \) in the context. The context that contains \( A_1 \) may be moved to the frame. But, we can still simulate the rule applications of \( G \) on the context in the frame. Since there is no loss of formulas as we follow up branches, the twig sequents that appear in \( \mathcal{G} \) can be proved by the partial derivation trees in the original tree of \( \mathcal{G} \). By repeatedly applying the transformation, we can remove all the violations of the focus conditions one by one and eventually build a valid derivation tree for \( G \vdash \Delta \vdash \Gamma \rightarrow C \) in \( G^{FN}_{I5S} \).

**Theorem E.5** (Soundness of \( G^{F}_{I5S} \) with respect to \( G^{FN}_{I5S} \))

1. If \( G^+ \vdash \Delta^= \vdash \Gamma^- \rightarrow C^+ \) then \( G \vdash \Delta \vdash \Gamma \rightarrow C \).
2. If \( G^+ \vdash \Delta^=, A^\sim \vdash \Gamma^- \triangleright C^+ \rightarrow A^\sim \rightarrow G \vdash \Delta \vdash \Gamma, A \vdash C \).
3. If \( G^+ \vdash \Delta^= \vdash \Gamma^- \triangleright \triangleright C^+ \rightarrow A^\sim \rightarrow then G \vdash \Delta, A \vdash \Gamma \rightarrow C \).
4. If \( G^-; \Gamma' \triangleright \triangleright \rightarrow C^+ \vdash \Delta^= \vdash \Gamma^- \rightarrow A^\sim \rightarrow then G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C \).
5. If \( G^-; \Gamma' \triangleright \triangleright C^+ \vdash \Delta^= \vdash \Gamma^- \rightarrow A^\sim \rightarrow then G; \Gamma', A \vdash \Delta \vdash \Gamma \vdash C \).

**Proof** We give a function \( f \) which takes a derivation tree in \( G^{F}_{I5S} \) and turn it into a derivation tree in \( G^{FN}_{I5S} \). Since the inference rules of the two systems have one-to-one correspondence, the transformation is straightforward. The difference between the two systems is that the inference rules of \( G^{F}_{I5S} \) copies the contexts of the conclusion sequents to the premise sequents. We can simply copies th contexts to the premise sequents. Since the terminal inference rules of \( G^{FN}_{I5S} \) allow non-empty contexts, the resulting derivation trees are valid in \( G^{FN}_{I5S} \). For an application of \( \triangleright \triangleright \mathcal{R}_b \) that is not included in \( G^{FN}_{I5S} \), the function \( f \) replace the application with an application of \( \triangleright \triangleright \mathcal{R} \).
Theorem E.6 (Completeness of $G_{IS5}^F$ with respect to $G_{IS5}^{FN}$)

1. If $G \vdash \Delta \vdash \Gamma \rightarrow C$ then $G_f \vdash \Delta_f \vdash \Gamma_f \mapsto C$.

2. If $G \vdash \Delta, A \vdash \Gamma \rightarrow C$ then $G_f \vdash \Delta_f \vdash A \mapsto \Gamma_f \mapsto C$.

3. If $G \vdash \Delta \vdash \Gamma \rightarrow C$ then $G_f \vdash \Delta_f \vdash \Gamma_f \mapsto C$.

4. If $G; \Gamma' \vdash \Delta \vdash \Gamma \rightarrow A$ then $G_f; \Gamma'_f \vdash \Delta_f \vdash \Gamma_f \mapsto C \vdash \mapsto A$.

5. If $G; \Gamma' \vdash \Delta \vdash \Gamma \rightarrow A$ then $G_f; \Gamma'_f \vdash \Delta_f \vdash \Gamma_f \mapsto C \vdash \mapsto A$.

where $G_f, \Delta_f, \Gamma_f$ and $\Gamma'_f$ are subsets of $G, \Delta, \Gamma$ and $\Gamma'$, respectively.

Proof

We give a function $f$ which transforms a derivation tree in $G_{IS5}^{FN}$ into a derivation tree in $G_{IS5}^F$. The function starts from the applications of the terminal inference rules. The function eliminates other formulas and leaves the principal formula. The function runs recursively to eliminate formulas that are not used as principal formulas of left-rule applications. The resulting sequent in $G_{IS5}^F$ is valid that the formulas in contexts are exhaustively used. For $\supset R$, the function selects one of $\supset L_{global}$ and $\supset L_{bglobal}$ depending on whether the formula $B$ is used in the derivation tree.

Appendix F  Bookkeeping Method for the Backward Proof Search

1. $\Box L$ derived rules

$$
\begin{array}{c}
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash B_1 \ldots G \vdash \Delta \vdash \Gamma_j \vdash B_n \\
G \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\end{array}
\Box L(i, j)
$$

(a) trunk $\Diamond L^R, \forall L^R, \bot L^R, \top L^R, \leftarrow full(R)$

(b) trunk $\Box L^R \leftarrow \Box L^R \cup \Box L^T \setminus \{(i, j)\}$

(c) trunk $\Box L^T, \Diamond L^T \leftarrow \{\}$

(d) twig $\Box L^R \leftarrow \Box L^R \setminus \{(i, j)\}$

(e) twig $\Box L^T \leftarrow \Box L^T \cup \{(i, j)\}$

2. $\Diamond L$ derived rules

$$
\begin{array}{c}
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash B_1 \ldots G \vdash \Delta \vdash \Gamma_j \vdash B_n \\
G; A \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\end{array}
\Diamond L(i, j)
$$
(a) trunk ∨ \mathcal{L}^R, \perp \mathcal{L}^R, \mathsf{Id}^R \leftarrow \text{full}(.^\mathcal{R})
(b) trunk □ \mathcal{L}^R \leftarrow □ \mathcal{L}^R \cup □ \mathcal{T}
(c) trunk ◊ \mathcal{L}^R \leftarrow ◊ \mathcal{L}^R \cup ◊ \mathcal{T} - \{(i, j)\}
(d) trunk □ \mathcal{T}, ◊ \mathcal{T} \leftarrow \{\}
(e) twig ◊ \mathcal{L}^R \leftarrow □ \mathcal{L}^R - \{(i, j)\}
(f) twig ◊ \mathcal{T} \leftarrow □ \mathcal{T} \cup \{(i, j)\}

3. \forall \mathcal{L} derived rules \((A, B \not\in \Gamma)\)

\[
\begin{array}{c}
\text{twigs} \\
G \vdash \Delta \vdash \Gamma_j \vdash B_1 \ldots G \vdash \Delta \vdash \Gamma_j \vdash B_n \\
G \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\quad
\begin{array}{c}
\text{trunks} \\
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G \vdash \Delta \vdash \Gamma_j \vdash B \vdash C
\end{array}
\quad
\begin{array}{c}
\forall \mathcal{L}(i, j)
\end{array}
\]

(a) trunk \perp \mathcal{L}^R, \mathsf{Id}^R \leftarrow \text{full}(.^\mathcal{R})
(b) trunk □ \mathcal{L}^R \leftarrow □ \mathcal{L}^R \cup □ \mathcal{T}
(c) trunk ◊ \mathcal{L}^R \leftarrow ◊ \mathcal{L}^R \cup ◊ \mathcal{T}
(d) trunk □ \mathcal{T}, ◊ \mathcal{T} \leftarrow \{\}
(e) trunk \forall \mathcal{L}^R \leftarrow \text{full}(\forall \mathcal{L}^R) - \{(i, j)\}
(f) twig \forall \mathcal{L}^R \leftarrow \forall \mathcal{L}^R - \{(i, j)\}

4. \mathsf{Id} derived rules

\[
\begin{array}{c}
\text{twigs} \\
G \vdash \Delta \vdash \Gamma_j \vdash B_1 \ldots G \vdash \Delta \vdash \Gamma_j \vdash B_n \\
G \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\quad
\begin{array}{c}
(i, j)
\end{array}
\]

(a) twig \mathsf{Id}^R \leftarrow \mathsf{Id}^R - \{(i, j)\}

5. \perp \mathcal{L} derived rules

\[
\begin{array}{c}
\text{twigs} \\
G \vdash \Delta \vdash \Gamma_j \vdash B_1 \ldots G \vdash \Delta \vdash \Gamma_j \vdash B_n \\
G \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\quad
\begin{array}{c}
(i, j)
\end{array}
\]

45
(a) twig \( \perp L^R \leftarrow \perp L^R \setminus \{(i, j)\} \)

6. \( \Box R, \Diamond R_b \)

\[
\begin{align*}
G; \Gamma \vdash \Delta \vdash \cdot \vdash A & \quad \Box R \\
\vdash \Delta \vdash \Gamma \vdash \square A & \quad G; \Gamma \vdash \Delta \vdash \Gamma \vdash A \quad \Diamond R_b
\end{align*}
\]

7. \( \exists R_a (A \not\in \Gamma) \)

\[
\begin{align*}
G \vdash \Delta \vdash \Gamma, A \vdash B & \quad \exists R_a \\
\vdash \Delta \vdash \Gamma \vdash A \cup B & \quad \exists R_a
\end{align*}
\]

(a) premise \( \square L^R \leftarrow \square L^R \cup \square L^T \)
(b) premise \( \Diamond L^R \leftarrow \Diamond L^R \cup \Diamond L^T \)
(c) premise \( \square L^T, \Diamond L^T \leftarrow \{} \)
(d) premise \( \forall L^R, \perp L^R, \text{Id}^R \leftarrow \text{full}(\cdot) \)

8. \( \land R, \lor R_l, \lor R_r, \Diamond R_a, \exists R_b (C' \text{ is the direct subformula of } C) \)

\[
\begin{align*}
G \vdash \Delta \vdash \Gamma \vdash C' & \quad R \\
G \vdash \Delta \vdash \Gamma \vdash C
\end{align*}
\]

(a) premise \( \square L^R \leftarrow \square L^R \cup \square L^T \)
(b) premise \( \Diamond L^R \leftarrow \Diamond L^R \cup \Diamond L^T \)
(c) premise \( \square L^T, \Diamond L^T \leftarrow \{} \)
(d) premise \( \forall L^R, \perp L^R, \text{Id}^R \leftarrow \text{full}(\cdot) \)

**Appendix G  Measure Vector by Bookkeeping Sets**

**Theorem G.1** (Termination of Backward Proof Search in \( G_{ISS}^B \))

*Backward proof search in \( G_{ISS}^B \) is guaranteed to terminate.*
Proof We define a tuple as the termination measure for sequents. The following function $f$ generates a tuple for a given sequent with bookkeeping set.

$$f(G \vdash \Delta \vdash \Gamma \vdash C \parallel \square L^R; \Diamond L^R; \land L^R; \perp L^R; \ld R; \Box L^T; \Diamond L^T) = \langle s_1, \ldots, s_9 \rangle$$

where

$$s_1 = |\square L^T \cup \square L^R| \quad s_2 = |\Diamond L^R \cup \Diamond L^T|$$
$$s_3 = \langle |\mathcal{F} - \Gamma_1|, \ldots, |\mathcal{F} - \Gamma_n|, |\mathcal{F} - \Delta| \rangle$$
$$s_4 = |\square L^R| \quad s_5 = |\Diamond L^R|$$
$$s_6 = |\forall L^R| \quad s_7 = |\perp L^R|$$
$$s_8 = |\ld R| \quad s_9 = size(C)$$

and $\mathcal{F}$ is the full set containing all the subformulas of the goal formula. The goal formula means the initial query formula the procedure is working on. Each element in the tuple is greater than or equal to zero. If we compare tuples of sequents in a rule application, the tuple $\langle v_1, \ldots, v_9 \rangle$ for a premise sequent is less than the tuple $\langle w_1, \ldots, w_9 \rangle$ for the conclusion sequent. Thus, the tuple decreases monotonically as the procedure follows up along branches. Since the tuple is lower-bounded by zero tuple $(0, \ldots, 0)$.

The recursive procedure of the backward proof search always terminates in finite depth. The maximum depth is determined by the size of bookkeeping sets and the number of subformulas for the goal formula.

The size function $size(C)$ quantifies the size of formula $C$ in a usual way. The tuple of context sizes

$$\langle |\mathcal{F} - \Gamma_1|, \ldots, |\mathcal{F} - \Gamma_n|, |\mathcal{F} - \Delta| \rangle$$

counts the number distinct formulas in a context and subtract from the full set. To build the tuple of context sizes and compare tuples, we need to enumerate all the contexts. A new context can be introduced to a sequent by rule application of $\square L$, $\Diamond L$ and $\square R$. The number of contexts introduced by these rules can increase as the procedure runs. However, the procedure compares pairs of contexts that have the same label and merged them into one context. The number of contexts in the frame is bounded by the number of subformula compositions. Plus, the number of possible contexts usually remains small in actual execution of the procedure.

Appendix H Soundness and Completeness of $G^B_{IS5}$ with respect to $G^F_{IS5}$

Theorem H.1 (Soundness of $G^B_{IS5}$ with respect to $G^F_{IS5}$)
If $\vdash \vdash \vdash C$ in $G^B_{IS5}$ then $\vdash \vdash \vdash \vdash C$ in $G^F_{IS5}$.

Theorem H.2 (Completeness of $G^B_{IS5}$ with respect to $G^F_{IS5}$)
If $\vdash \vdash \vdash \vdash C$ in $G^F_{IS5}$ then $\vdash \vdash \vdash \vdash C$ in $G^B_{IS5}$.

47
\( G_{IS5}^B \) consists of the derived rules for the goal formula and the right rules of \( G_{IS5} \). \( G_{IS5}^B \) has no focus in left-rules since we replace the left-rules with the derived rules that hide their focused thread. The derived rules that are constructed in \( G_{IS5}^B \) are equally valid in \( G_{IS5}^{FN} \). All the derived rules in \( G_{IS5}^F \) can be simulated by the left-rules of \( G_{IS5}^{FN} \). Any application of left-rules in \( G_{IS5}^F \) forms a focused thread which is found in the construction of derived rules. That is, replacing all the left-rules of \( G_{IS5}^{FN} \) with derived rules is valid and two systems are sound and complete with respect to each other.

**Theorem H.3** (Completeness of \( G_{IS5}^B \) with bookkeeping condition)

*If \( \vdash \vdash \vdash \vdash C \) in \( G_{IS5}^B \), then we can build a derivation tree \( E \) for \( \vdash \vdash \vdash \vdash C \) with initial bookkeeping sets generated from \( C \) (\( E \) conforms with the bookkeeping conditions).*

**Proof** We show that the derivation tree violating the bookkeeping conditions can be transformed into a valid derivation tree that conforms the conditions. The following cases show how to remove a violating rule application in a derivation tree. By repeatedly applying the transformation, we can build a valid tree that has no violation of the conditions.

**Case:** An application of a \( \Box L \) derived rule

\[
\begin{array}{c}
G \vdash \Delta, A \vdash \Gamma_j \vdash C \\
D_1 \\
\vdots \\
D_n \\
G \vdash \Delta \vdash \Gamma_j \vdash C \\
\hline \Box L(i, j)
\end{array}
\]

violates the bookkeeping condition on \( \Box L^R \). Applying a \( \Box L \) derived rule requires the instance \( (i, j) \) in the resource set \( \Box L^R \). Since the application of \( (i, j) \) violates the condition, \( \Box L^R \) does not contain \( (i, j) \) in it. That is, on the current branch from the goal sequent to \( G \vdash \Delta \vdash \Gamma_j \vdash C \), another rule application of \( (i, j) \) removed \( (i, j) \) from \( \Box L^R \). We can locate the closest application of \( (i, j) \) on the current branch.

**Subcase:** The current branch is on the trunk premise of the closest application of \( (i, j) \).

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma' \vdash C' \\
D'_1 \\
\vdots \\
D'_n \\
G' \vdash \Delta' \vdash \Gamma' \vdash C' \\
\hline \Box L(i, j)
\end{array}
\]

Since a formula in the global context never disappears as we follow up along a branch, the global context \( \Delta \) contains \( A \). Thus, the trunk premise of upper rule application \( G \vdash \Delta, A \vdash \Gamma_j \vdash C \) is exactly the same as the conclusion sequent \( G' \vdash \Delta' \vdash \Gamma_j \vdash C \). We can skip the rule application \( (i, j) \) and replace the derivation tree on \( G \vdash \Delta \vdash \Gamma_j \vdash C \) with \( E \). The transformed derivation tree will be:

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma' \vdash C' \\
D'_1 \\
\vdots \\
D'_n \\
G' \vdash \Delta' \vdash \Gamma' \vdash C' \\
\hline \Box L(i, j)
\end{array}
\]
Subcase: The current branch is on one of the twig premises of the closest application of \( (i, j) \).

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma_j' \vdash C' \\
\frac{\mathcal{D}_i}{\mathcal{F}} \quad \frac{\mathcal{D}_j}{\mathcal{F}} \\
G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash C''
\end{array}
\]

Every sequent of branch \( \mathcal{G} \) are twig sequents. Otherwise, \( (i, j) \) in \( \Box L^T \) would be moved to \( \Box L^R \) and the violation would never occur. Thus, the contexts in \( G \vdash \Delta \vdash \Gamma_j \vdash C \) are equivalent to corresponding contexts in \( G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash B_k \). That is, \( G = G'' \), \( \Delta = \Delta'' \) and \( \Gamma = \Gamma_k'' \). Thus, the twig premise of the upper application \( G \vdash \Delta \vdash \Gamma_j \vdash B_k \) is exactly the same as \( G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash B_k \). We can replace the partial derivation tree from \( \mathcal{G} \) with \( \mathcal{D}_k \). The resulting derivation tree will be:

\[
\begin{array}{c}
G' \vdash \Delta' \vdash \Gamma_j' \vdash C' \\
G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash C'' \\
\frac{\mathcal{D}_i}{\mathcal{F}} \quad \frac{\mathcal{D}_j}{\mathcal{F}} \\
\frac{\mathcal{D}_k}{\mathcal{F}} \quad \frac{\mathcal{D}_n}{\mathcal{F}}
\end{array}
\]

Case: An application of \( \Diamond L \) derived rule

\[
\begin{array}{c}
G; \{ A \}_{(i,m)} \vdash \Delta \vdash \Gamma_j \vdash C \\
\frac{\mathcal{D}_1}{\mathcal{F}} \quad \frac{\mathcal{D}_n}{\mathcal{F}}
\end{array}
\]

violates the bookkeeping condition on \( \Diamond L^R \). Applying a \( \Diamond L \) derived rule requires the instance \( (i, j) \) in the resource set \( \Diamond L^R \). Since the application of \( (i, j) \) violates the condition, \( \Diamond L^R \) does not contain \( (i, j) \) in it. That is, on the current branch from the goal sequent to \( G \vdash \Delta \vdash \Gamma_j \vdash C \), another rule application of \( (i, j) \) removed \( (i, j) \) from \( \Diamond L^R \). We can locate the closest application of \( (i, j) \) on the current branch.

Subcase: The current branch is on the trunk premise of the closest application of \( (i, j) \).

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma_j' \vdash C' \\
\frac{\mathcal{D}_i}{\mathcal{F}} \quad \frac{\mathcal{D}_j}{\mathcal{F}} \\
G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash C''
\end{array}
\]

The rule application \( (i, j) \) introduces a new context of \( A \) into the frame of the trunk sequent. The frame \( G \) also contains the context annotated with \( i \) and the context has \( A \) in it. The upper rule application \( (i, j) \) introduce another context with \( i \) annotation. The trunk sequent \( G; \{ A \}_{(i,m')} \vdash \Delta \vdash \Gamma_j \vdash C \) has duplication of contexts in the frame. By the contraction property on contexts, we can build a derivation tree for the sequent without duplication. Assuming that the derivation tree for \( G \vdash \Delta \vdash \Gamma_j \vdash C \) is \( \mathcal{E}' \), the resulting derivation tree will be:

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma_j' \vdash C' \\
\frac{\mathcal{D}_i}{\mathcal{F}} \quad \frac{\mathcal{D}_j}{\mathcal{F}} \\
G'' \vdash \Delta'' \vdash \Gamma_j'' \vdash C''
\end{array}
\]

49
Subcase: The current branch is on one of the twig premises of the closest application of \((i, j)\).

\[
\begin{array}{c}
F = \\
\begin{array}{c}
G' \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma_j \vdash C
\end{array}
\end{array}
\]

A rule application that changes the contexts moves the instances from \(\Diamond L^T\) to \(\Diamond L^R\) or refills \(\Diamond L^R\). Since the upper rule application \((i, j)\) violates the condition that \((i, j) \in \Diamond L^R\), \(G \vdash \Delta \vdash \Gamma_j \vdash C\) has the same contexts as \(G' \vdash \Delta' \vdash \Gamma'_j \vdash B_k\). That is, \(G = G'\), \(\Delta = \Delta'\) and \(\Gamma = \Gamma'_k\). Thus, we can replace the partial derivation tree of \(G\) with \(D_k\).

The resulting derivation tree will be:

\[
\begin{array}{c}
F = \\
\begin{array}{c}
G' \vdash \Delta' \vdash \Gamma'_j \vdash C' \\
G' \vdash \Delta' \vdash \Gamma'_j \vdash C' \end{array}
\end{array}
\]

Case: An application of \(\forall L\) derived rule

\[
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j, A \vdash C \\
G \vdash \Delta \vdash \Gamma_j, B \vdash C
\end{array}
\]

violates the bookkeeping condition on \(\forall L^R\). Applying a \(\forall L\) derived rule requires the instance \((i, j)\) in the resource set \(\forall L^R\). Since the application of \((i, j)\) violates the condition, \(\forall L^R\) does not contain \((i, j)\) in it. That is, on the current branch from the goal sequent to \(G \vdash \Delta \vdash \Gamma_j \vdash C\), another rule application of \((i, j)\) removed \((i, j)\) from \(\forall L^R\). We can locate the closest application of \((i, j)\) on the current branch.

Subcase: The current branch is on the trunk premise of the closest application of \((i, j)\).

\[
\begin{array}{c}
F = \\
\begin{array}{c}
G \vdash \Delta \vdash \Gamma_j \vdash C \\
G \vdash \Delta \vdash \Gamma_j \vdash C
\end{array}
\end{array}
\]

The formula \(A\) added to \(\Gamma'_j\) in the lower application remains in the context \(j\). The context \(j\) of \(G \vdash \Delta \vdash \Gamma_j \vdash C\) also contains the formula \(A\). By the condition of \(\forall L\) derived rule application, applying \((i, j)\) on the sequent was impossible.

Subcase: The current branch is on one of the twig premises of the closest application of \((i, j)\).

\[
\begin{array}{c}
F = \\
\begin{array}{c}
G' \vdash \Delta' \vdash \Gamma_j \vdash C \\
G' \vdash \Delta' \vdash \Gamma_j \vdash C
\end{array}
\end{array}
\]

A rule application that changes the contexts refills \(\forall L^R\). Since the upper rule application \((i, j)\) violates the condition that \((i, j) \in \forall L^R\), \(G \vdash \Delta \vdash \Gamma_j \vdash C\) has the same
contexts as $G' \vdash \Delta' \vdash \Gamma'_j \vdash B_k$. That is, $G = G'$, $\Delta = \Delta'$ and $\Gamma = \Gamma'_k$. Thus, we can replace the partial derivation tree of $G$ with $D_k$. The resulting derivation tree will be:

$$
F = \frac{\mathcal{E}_1' \quad \mathcal{E}_2' \quad \mathcal{D}' \quad \mathcal{D}_k \quad \mathcal{D}'_n}{G' \vdash \Delta' \vdash \Gamma'_j \vdash C'} \quad \forall(i, j)
$$

Case: An application of $\text{Id}$, $\bot$

$$
F = \frac{\mathcal{D}_1 \quad \mathcal{D}_1}{G \vdash \Delta \vdash \Gamma_j \vdash C'} \quad (i, j)
$$

$$
F' = \frac{G \vdash \Delta \vdash \Gamma_j \vdash C'}{G' \vdash \Delta' \vdash \Gamma'_j \vdash C'} \quad (i, j)
$$

A rule application that changes the contexts refills $\text{ld}^R$ and $\bot^R$. Since the upper rule application $(i, j)$ violates the condition that $(i, j) \in \text{ld}^R$ or $\bot^R$, $G \vdash \Delta \vdash \Gamma_j \vdash C$ has the same contexts as $G' \vdash \Delta' \vdash \Gamma'_j \vdash B_k$. That is, $G = G'$, $\Delta = \Delta'$ and $\Gamma = \Gamma'_k$. Thus, we can replace the partial derivation tree of $G$ with $D_k$. The resulting derivation tree will be:

$$
F = \frac{\mathcal{D}' \quad \mathcal{D}_k \quad \mathcal{D}'_n}{G' \vdash \Delta' \vdash \Gamma'_j \vdash C'} \quad (i, j)
$$

References


