A Labelled Sequent Calculus for BBI: Proof Theory and Proof Search

Zhé Hóu*[1], Alwen Tiu†[2], and Rajeev Goré‡[1]

[1]Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia
[2]School of Computer Engineering, Nanyang Technological University, Singapore 639798

Abstract

We present a labelled sequent calculus for Boolean BI (BBI), a classical variant of the logic of Bunched Implications. The calculus is simple, sound, complete, and enjoys cut-elimination. We show that all the structural rules in the calculus, i.e., those rules that manipulate labels and ternary relations, can be localised around applications of certain logical rules, thereby localising the handling of these rules in proof search. Based on this, we demonstrate a free variable calculus that deals with the structural rules lazily in a constraint system. We propose a heuristic method to quickly solve certain constraints, and show some experimental results to confirm that our approach is feasible for proof search. Additionally, we show that different semantics for BBI and some axioms in concrete models can be captured modularly simply by adding extra structural rules.

Keywords. labelled sequent calculus, automated reasoning, BBI, free variables, proof theory

1 Introduction

The logic of bunched implications (BI) was introduced to reason about resources using additive connectives $\land$, $\lor$, $\rightarrow$, $\top$, $\bot$, and multiplicative connectives $\top^*$, $*$, $-\!\!*$ [19]. Both parts are intuitionistic so BI is also Intuitionistic logic (IL) plus Lambek multiplicative logic (LM). Changing the additive part to classical logic gives Boolean BI (BBI). Replacing LM by multiplicative classical linear logic gives Classical BI (CBI). BI logics are closely related to separation logic [22], a logic for proving properties of programs. Thus, the semantics and proof theory of BI-logics, particularly for proof search, are important in computer science.

The ternary relational Kripke semantics of BBI-logics come in at least three different flavours: non-deterministic (ND), partial deterministic (PD), and total deterministic (TD) [15]. These semantics give different logics w.r.t. validity, respectively called $BBI_{ND}, BBI_{PD}, BBI_{TD}$, and all are

*Email: zhe.hou@anu.edu.au, Phone: +61 2 6125 1771
†Email: atiu@ntu.edu.sg, Phone: +65 6790 6107
‡Email: rajeev.gore@anu.edu.au, Phone: +61 2 6125 8603

undecidable [4, 15]. The purely syntactic proof theory of BBI also comes in three flavours: Hilbert calculi [21, 10], display calculi [2] and nested sequent calculi [20]. All are sound and complete w.r.t. the ND-semantics [10, 2, 20].

In between the relational semantics and the purely syntactic proof theory are the labelled tableaux of Larchey-Wendling and Galmiche which are sound and complete w.r.t. the PD-semantics [14, 13]. They remark that "the adaptation of this tableaux system to BBI_{TD} should be straightforward (contrary to BBI_{ND})" [16]. We return to these issues in Section 8.

At a very high level, each logical rule introduces some logical connective into the conclusion while a structural rule does not involve any logical connective explicitly. Thus a logical rule is driven by the presence of a logical connective in the conclusion while a structural rule is not. In this setting, effective backward proof search requires that the rules of the calculus have certain properties. First, we prefer as many rules to be invertible as possible since these can be applied backwards as soon as possible without losing derivations. Second, we prefer the logical rules to have the subformula property so that they cannot be applied backwards *ad infinitum.* Third, we prefer as few structural rules as possible since, typically, these can be applied to any sequent causing non-determinism and possibly also non-termination. Fourth, when structural rules are necessary, we would like to drive the applications of structural rules using a systematic proof search procedure (strategy) which is driven by the applications of logical rules, as this may reduce non-determinism.

The display postulates and other structural rules of display calculi, especially the contraction rules on structures, are impractical for backward proof search since the display postulates (merely) shuffle structures in a sequent, whereas the contraction rules copy structures from the conclusion to the premise. Moreover, both can be applied to (almost) every sequent so these rules are applicable at any stage of the proof search, and they can easily generate redundant derivations when it is not clear which inference rule to use. Hence it is hard to give a systematic proof search procedure without controlling these rules. Nested sequents usually face similar problems with the contraction rules and propagation rules, and although Park et al. [20] showed the admissibility of contraction in an improved nested sequent calculus, it contains other rules that explicitly contract structures. Their iterative deepening automated theorem prover for BBI based on nested sequents is terminating and incomplete for bounded depths, but complete and potentially non-terminating for an unbounded depth [20]. The labelled tableaux of Larchey-Wendling and Galmiche compile all structural rules into PD-monoidal constraints, and are cut-free complete for $\mathrm{BBI}_{\mathrm{PD}}$ using a potentially infinite counter-model construction [13]. But effective proof search using a strategy is only a "perspective" and is left as further work [13, page 2].

Surprisingly, many applications of BBI do not directly correspond to its widely used non-deterministic semantics. For example, separation logic models are instances of partial deterministic models [15] while "memory models" for BBI are restricted to have *indivisible units*: "the empty memory cannot be split into non-empty pieces" [4]. Other variants of separation logic employ semantics based on separation algebra [7] or separation theory [6] that are useful in program verification. Our goal is to give a labelled proof system for BBI based upon the ND-semantics which easily extends to the PD-, TD- and other application-driven semantics, while being amenable to effective proof search.

Our labelled sequent calculus LS_{BBI} for BBI adopts some features from existing labelled tableaux for BBI [14] and existing labelled sequent calculi for modal logics [17]. Unlike these calculi, some LS_{BBI}-rules contain substitutions on labels. As in traditional sequent calculi without labels, we are able to absorb the effects of weakening and contraction into some of the logical rules, thereby allowing us to leave out these structural rules by making them admissible. The absorption of contraction makes many logical rules invertible as desired but breaks the subformula property, which means that we lose termination. Some logical rules are still not invertible after absorbing contraction, but the presence of labels allows us to make even these rules invertible, at the price of requiring further non-logical rules to explicitly encode the frame conditions of the underlying (Kripke) semantics. We refer to such rules simply as structural rules.

From a proof-search perspective, labelled calculi are no better than display calculi because these extra structural rules are just as bad as display postulates for proof search since they are typically applicable (backwards) to every sequent. As a step towards our goal, we show that the applications of these extra structural rules are necessary only prior to applying certain logical rules backwards. Thus backward proof search in LS_{BBI} can be structured so that we first apply one fixed set of invertible logical rules, then apply these extra structural rules, and then apply the remaining set of invertible logical rules, leading to a syntax-driven proof search procedure for LS_{BBI} in which all the rule applications in proof search are decided by the logical connectives (the syntax), and redundant derivations caused by structural rules are reduced. In essence, our proof search procedure for a given formula consists of two stages: one where we guess the shape of a candidate derivation tree obtained by applications of introduction rules for logical connectives occurring in the given formula, and the other where we solve certain structural constraints on the relations between labels generated from the applications of these logical rules. The constraint solving part involves only reasoning in the ND-monoidal semantics of BBI and validates that the chosen candidate proof tree does indeed form a valid proof. Technically, this is done by introducing two intermediate proof systems: LS^{e}_{BBI} and LS^{sf}_{BBI} (Section 4) where structural rules are isolated into side conditions of introduction rules, and $FVLS_{BBI}$ (Section 5) where those side conditions are solved lazily by delaying the instantiation of free-variables until we reach zero-premise rules.

Our work is novel from three perspectives. Compared to the labelled tableaux of Larchey-Wendling and Galmiche, we deal with the non-deterministic semantics of BBI, which they have flagged as a difficulty, and we obtain a constructive cut-elimination procedure. Compared to the nested sequent calculus of Park et al., our calculus has much simpler structural rules. Some of Park et al.'s structural rules and traverse rules involve copying structures, and when made to absorb contraction, these rules (especially EA_C) are extraordinarily long and complicated. Our structural rules directly capture the semantics using ternary relational atoms, thus they are very intuitive and easy to read. Note that Park et al. actually gave a labelled variant of their nested sequent calculus, with the same logical rules as ours. However, their structural rules are still just notational variants of the original ones, which are lengthy and do not use ternary relations. We also show that adding certain structural rules to LS_{BBI} allows us to obtain cut-free labelled calculi for all the other semantics mentioned above. Finally, as far as we know, there are no other free-variable calculi for the logics we consider, let alone ones which are amenable to proof search as are our calculi.

The rest of the paper is organized as follows. In Section 2, we present the semantics of BBI, following [15], and our labelled sequent calculus LS_{BBI}. We show that LS_{BBI} is sound with respect to the semantics, and it is complete indirectly via a Hilbert system for BBI [10] which is already known to be complete. In Section 3, we prove some important proof theoretic properties of LS_{BBI}: invertibility of inference rules, admissibility of contraction, and more importantly, cut-elimination. In Section 4, we discuss a permutation result for inference rules, allowing us to isolate applications of structural rules to certain stages of systematic backward proof search. In Section 5, we describe how to reduce proof search to constraint solving in a free-variable sequent calculus. We give a heuristic method for solving the resulting constraint problem in Section 6 and report on experimental results in Section 7. Section 8 concludes the paper. Detailed proofs are available in appendices.

2 The Labelled Sequent Calculus for BBI

The semantics of BBI is in Section 2.1, then we present our labelled calculus in Section 2.2. The soundness proof is outlined in Section 2.3, followed by the completeness proof in Section 2.4, in which the Hilbert system of BBI is used.

2.1 Syntax and Semantics of BBI

BBI formulae are defined inductively as follows, where p is an atomic proposition, $\top^*, *, -\!\!*$ are the multiplicative unit, multiplicative conjunction, and multiplicative implication respectively:

$$A ::= p \mid \top \mid \bot \mid \neg A \mid A \vee A \mid A \wedge A \mid A \to A \mid \top^* \mid A * A \mid A -\!\!* A$$

The labelled sequent calculus for BBI employs a ternary relation of worlds that is based on a non-deterministic monoid structure, à la Galmiche et al. [10].

A non-deterministic monoid is a triple $(\mathcal{M}, \circ, \epsilon)$ where $\mathcal{M}$ is a non-empty set, $\epsilon \in \mathcal{M}$ and $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{P}(\mathcal{M})$. The extension of $\circ$ to $\mathcal{P}(\mathcal{M})$ uses $X \circ Y = \bigcup\{x \circ y : x \in X, y \in Y\}$. The following conditions hold in this monoid:

- *Identity:* $\forall a \in \mathcal{M}.\epsilon \circ a = \{a\}$
- *Commutativity:* $\forall a, b \in \mathcal{M}.a \circ b = b \circ a$
- *Associativity:* $\forall a, b, c \in \mathcal{M}.a \circ (b \circ c) = (a \circ b) \circ c.$

The ternary relation over worlds is defined by $\triangleright \subseteq \mathcal{M} \times \mathcal{M} \times \mathcal{M}$ such that $\triangleright(a, b, c)$ if and only if $c \in a \circ b$. Following Galmiche et al., we write $a, b \triangleright c$ instead of $\triangleright(a, b, c)$. We therefore have the following conditions for all $a, b, c, d \in \mathcal{M}$:

- *Identity:* $\epsilon, a \triangleright b$ iff $a = b$
- *Commutativity:* $a, b \triangleright c$ iff $b, a \triangleright c$
- *Associativity:* If $\exists k$ s.t. $(a, k \triangleright d)$ and $(b, c \triangleright k)$ then $\exists l$ s.t. $(a, b \triangleright l)$ and $(l, c \triangleright d)$.

Therefore we obtain a BBI *relational frame* $(\mathcal{M}, \triangleright, \epsilon)$ from a non-deterministic monoid $(\mathcal{M}, \circ, \epsilon)$ in the obvious way. Intuitively, the relation $x, y \triangleright z$ means that z can be partitioned into two parts: x and y. The identity condition can be read as every world can be partitioned into an empty world and itself. Commutativity captures that partitioning z into x and y is the same as partitioning z into y and x. Finally, associativity means that if z can be partitioned into x and y, and x can further be partitioned into u and v, then all together z consists of u, v and y. Therefore there must exist an element w which is the combination of v and y, such that w and u form z. Note that since we do not restrict this monoid to be cancellative, $(x, y \triangleright x)$ does not imply $y = \epsilon$.

A BBI model $(\mathcal{M}, \triangleright, \epsilon, v)$ consists of a relational frame $(\mathcal{M}, \triangleright, \epsilon)$ and a *valuation* $v : Var \to \mathcal{P}(\mathcal{M})$. A *forcing relation* "$\Vdash$" between elements of $\mathcal{M}$ and BBI-formulae is defined as follows [10]:

$$\begin{array}{ll}
m \Vdash \top^* \text{ iff } m = \epsilon & m \Vdash P \text{ iff } P \in Var \text{ and } m \in v(P) \\
m \Vdash \bot \text{ iff never} & m \Vdash A \vee B \text{ iff } m \Vdash A \text{ or } m \Vdash B \\
m \Vdash \top \text{ iff always} & m \Vdash A \wedge B \text{ iff } m \Vdash A \text{ and } m \Vdash B \\
m \Vdash \neg A \text{ iff } m \not\Vdash A & m \Vdash A \to B \text{ iff } m \not\Vdash A \text{ or } m \Vdash B \\
\multicolumn{2}{l}{m \Vdash A * B \text{ iff } \exists a, b.(a, b \triangleright m \text{ and } a \Vdash A \text{ and } b \Vdash B)} \\
\multicolumn{2}{l}{m \Vdash A -\!\!* B \text{ iff } \forall a, b.((m, a \triangleright b \text{ and } a \Vdash A) \text{ implies } b \Vdash B)}
\end{array}$$

Given a model $(\mathcal{M}, \triangleright, \epsilon, v)$, a formula A is true at $m \in \mathcal{M}$ if $m \Vdash A$. A formula A is *valid* if it is true at every world in every model.

2.2 The Labelled Sequent Calculus

We now give the details of our labelled sequent calculus with some discussion on how it is related to the semantics.

We assume an infinite set $LVAR$ of *label variables*, and a *label constant* ϵ. The latter is a syntactic counterpart of the ϵ world in the semantics. We shall use lower case letters to range over labels, i.e, the set $LVAR \cup \{\epsilon\}$. A *labelled formula* is an expression of the form $x : A$, where x is a label and A is a formula. A *relational atom* is an expression of the form $(x, y \triangleright z)$, where x, y and z are labels. Given a relational frame $(\mathcal{M}, \triangleright, \epsilon)$, the intended interpretations of labels are worlds in $\mathcal{M}$, and the intended interpretation of the symbol $\triangleright$ is the ternary relation $\triangleright$ in the model. The interpretations of labelled formulae and relational atoms in the semantics are dependent on the interpretation of labels. The latter is given by a mapping $\rho : \{\epsilon\} \cup LVAR \to \mathcal{M}$, with $\rho(\epsilon) = \epsilon$. That is, we fix the interpretation of the label constant ϵ to be the world ϵ in the semantics. Given such a ρ, a labelled formula $w : A$ means formula A is true in world $\rho(w)$. A relational atom $(x, y \triangleright z)$ is interpreted as $\rho(x), \rho(y) \triangleright \rho(z)$ in the semantics. That is, a labelled formula $w : A$ is true iff $\rho(w) \Vdash A$, and a relational atom $(x, y \triangleright z)$ is true iff $\rho(x), \rho(y) \triangleright \rho(z)$ holds.

A sequent is of the form $\Gamma \vdash \Delta$, where Γ and Δ are *structures*, defined formally via:

$$\Gamma ::= w : A \mid (x, y \triangleright z) \mid \Gamma; \Gamma \qquad\qquad \Delta ::= w : A \mid \Delta; \Delta$$

In our definition of sequents, the structural connective ";" in the antecedent means (additive) "and" whereas in the succedent it means (additive) "or". We assume implicitly that ';' is associative and commutative. Note that, however, there is a subtlety in this interpretation of ';'. Unlike traditional Gentzen sequents, a labelled sequent $\Gamma \vdash \Delta$ does not in general correspond to a formula $\bigwedge \Gamma \to \bigvee \Delta$. This is because the interpretation of a labelled formula is dependent on the interpretation of the label it is attached to. If, however, all formulas in Γ are attached to the same label, then the Γ corresponds to the formula $\bigwedge \Gamma$.

Our use of ';' as the structural connective in labelled sequents is slightly different from the traditional sequent notation where "," is used as the structural connective. Our notation is consistent with sequent systems for the family of Bunched Implication (BI) logics, where ";" is the additive structural connective, and "," is used to denote the multiplicative structural connective. The multiplicative structural connective is not explicitly presented in our sequent notation, but as we shall see later, it is encoded implicitly in the relational atoms.

The inference rules of our labelled system LS_{BBI} are shown in Figure 1, where we use P as a proposition, A, B are formulae, w, x, y, z are in the set $LVAR$ of labels, ϵ is the label constant. The formula introduced in the conclusion of each rule is the *principal formula*, and the relational atom introduced in the conclusion of each rule is the *principal relational atom*.

Our inference rules are designed to capture the semantics of BBI (cf. end of Section 2.1). For example, reading upward, the left rule for $*$ considers the situation where the formula $A * B$ is true in z, this involves an existential condition in the semantics:

$$\exists a, b \text{ s.t. } (a, b \triangleright z) \text{ holds and } a \Vdash A \text{ and } b \Vdash B.$$

As in the sequent calculus for first-order logic, we create fresh labels for existentially quantified variables. Therefore $*L$ creates a premise containing a new relational atom $(x, y \triangleright z)$ where x, y are fresh, and makes A true in x and B true in y. The $*R$ rule considers the case where the formula $A * B$ is false in z, by negating the semantics of $*$, we obtain that

$$\forall a, b, \text{ if } (a, b \triangleright z) \text{ holds, then } a \not\Vdash A \text{ or } b \not\Vdash B.$$

Therefore $*R$ checks existing relational atoms in the conclusion, and when $(x, y \triangleright z)$ is found for any labels x, y, the rule creates two premises for A being false in x and B being false in y respectively. The rules for $-\!\!*$ are analogous: the $-\!\!*\ L$ rule uses an existing relational atom in the conclusion as $-\!\!*\ L$ involves an universal condition; and the $-\!\!*\ R$ rule creates a new relational atom with fresh labels since it involves an existential condition. Similarly, in rules A, A_C, the label w must be fresh in the premise, as it also encodes an existential condition. Note that the rule A_C is a special case of the rule A where the two relational atoms are the same. A_C is required to guarantee the admissibility of contraction, as will be shown in the proof of Lemma 3.4.

In the rule $\top^* L$, there is an operation of global substitution $[\epsilon/x]$ in the premise. A substitution $\Gamma[y/x]$ is defined in the usual way: replace every occurrence of x in Γ by y. The structural rules Eq_1 and Eq_2 also involve substitutions, both of them are needed since substituting the label constant ϵ is forbidden. That is, if the principal relational atom is $(\epsilon, \epsilon \triangleright w)$, then we can only use the rule Eq_2 to replace every w by ϵ, but we cannot use the Eq_1 rule to replace every ϵ by w. The case where Eq_2 cannot be used is symmetric.

The *additive rules* ($\bot L$, $\top R$, $\wedge L$, $\wedge R$, $\rightarrow L$, $\rightarrow R$) and the *multiplicative rules* ($\top^* L$, $\top^* R$, $*L$, $*R$, $-\!\!*\ L$, $-\!\!*\ R$) respectively deal with the additive/ multiplicative connectives. The *zero-premise rules* are those with no premise (id, $\bot L$, $\top R$, $\top^* R$). Figure 2 shows an example derivation in LS_{BBI}, where we omit the unimportant parts, and use $r \times n$ if a rule r is applied n times.

To prove a formula, we start (at the bottom) by labelling the formula with an arbitrary world w, then try to apply the rules in LS_{BBI} backwards. A sequent is *provable/derivable* if every branch in the backward proof search can be closed by a zero-premise rule application.

The fact that weakening and contraction are forbidden in the multiplicative fragment of BBI is reflected in our calculus as follows. The rules $*L$ and $-\!\!*\ R$ create new relations in moving from conclusions to premises, so $\vdash x : (p * p) \rightarrow p$ is not derivable. In a cut-free derivation, a relation atom of the form $(w, w \triangleright w)$ where $w \neq \epsilon$ can never be created, so $\vdash x : p \rightarrow (p * p)$ is not derivable.

Definition 2.1 (Sequent Validity)**.** A sequent $\Gamma \vdash \Delta$ in LS_{BBI} is valid if for all $(\mathcal{M}, \triangleright, \epsilon)$, v and ρ, if every member of Γ is true then so is some member of Δ.

Note that BBI-validity of a formula A corresponds to the validity of the sequent $\vdash x : A$, where x is an arbitrary label. This correspondence is also adopted in other work for BBI [15, 20] and CBI [3], but is stronger than that used in the sequent calculus LBI for BI [1] [21], in which a formula A is valid iff $\emptyset_m \vdash A$ is provable, where $\emptyset_m$ is the multiplicative structural unit. For example, $\emptyset_m \vdash \top^*$ is provable in LBI, but the sequent $\vdash x : \top^*$ is not provable (although the sequent $\vdash \epsilon : \top^*$ is provable) in LS_{BBI}. Translated to our setting, validity of a formula A in BI would correspond to provability of $\vdash \epsilon : A$.

2.3 Soundness

The soundness proof reasons about the falsifiability of sequents, which is defined as follows.

Definition 2.2 (Sequent Falsifiability)**.** A sequent $\Gamma \vdash \Delta$ in LS_{BBI} is falsifiable iff there exist some $(\mathcal{M}, \triangleright, \epsilon)$, v and ρ, such that every relational atom and labelled formula in Γ is true and every labelled formula in Δ is false, where: (1) $w : A$ is true iff $\rho(w) \Vdash A$; (2) $w : A$ is false iff $\rho(w) \not\Vdash A$; and (3) $(x, y \triangleright z)$ is true iff $\rho(x), \rho(y) \triangleright \rho(z)$ holds.

[1] BI is defined proof theoretically, so the validity of its formulae is defined by sequent validity in LBI.

Identity and Cut:

$$\frac{}{\Gamma; w : P \vdash w : P; \Delta}\ id \qquad \frac{\Gamma \vdash x : A; \Delta \quad \Gamma'; x : A \vdash \Delta'}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\ cut$$

Logical Rules:

$$\frac{}{\Gamma; w : \bot \vdash \Delta}\ \bot L \qquad \frac{\Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]}{\Gamma; w : \top^* \vdash \Delta}\ \top^* L \qquad \frac{}{\Gamma \vdash w : \top; \Delta}\ \top R \qquad \frac{}{\Gamma \vdash \epsilon : \top^*; \Delta}\ \top^* R$$

$$\frac{\Gamma; w : A; w : B \vdash \Delta}{\Gamma; w : A \land B \vdash \Delta}\ \land L \qquad \frac{\Gamma \vdash w : A; \Delta \quad \Gamma \vdash w : B; \Delta}{\Gamma \vdash w : A \land B; \Delta}\ \land R$$

$$\frac{\Gamma \vdash w : A; \Delta \quad \Gamma; w : B \vdash \Delta}{\Gamma; w : A \to B \vdash \Delta}\ \to L \qquad \frac{\Gamma; w : A \vdash w : B; \Delta}{\Gamma \vdash w : A \to B; \Delta}\ \to R$$

$$\frac{(x, y \triangleright z); \Gamma; x : A; y : B \vdash \Delta}{\Gamma; z : A * B \vdash \Delta}\ *L \qquad \frac{(x, y \triangleright z); \Gamma; x : A \vdash z : B; \Delta}{\Gamma \vdash y : A \mathrel{-\!\!*} B; \Delta}\ \mathrel{-\!\!*} R$$

$$\frac{(x, y \triangleright z); \Gamma \vdash x : A; z : A * B; \Delta \quad (x, y \triangleright z); \Gamma \vdash y : B; z : A * B; \Delta}{(x, y \triangleright z); \Gamma \vdash z : A * B; \Delta}\ *R$$

$$\frac{(x, y \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B \vdash x : A; \Delta \quad (x, y \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B; z : B \vdash \Delta}{(x, y \triangleright z); \Gamma; y : A \mathrel{-\!\!*} B \vdash \Delta}\ \mathrel{-\!\!*} L$$

Structural Rules:

$$\frac{(y, x \triangleright z); (x, y \triangleright z); \Gamma \vdash \Delta}{(x, y \triangleright z); \Gamma \vdash \Delta}\ E \qquad \frac{(u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x); \Gamma \vdash \Delta}{(x, y \triangleright z); (u, v \triangleright x); \Gamma \vdash \Delta}\ A$$

$$\frac{(x, \epsilon \triangleright x); \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ U \qquad \frac{(x, w \triangleright x); (y, y \triangleright w); (x, y \triangleright x); \Gamma \vdash \Delta}{(x, y \triangleright x); \Gamma \vdash \Delta}\ A_C$$

$$\frac{(\epsilon, w' \triangleright w'); \Gamma[w'/w] \vdash \Delta[w'/w]}{(\epsilon, w \triangleright w'); \Gamma \vdash \Delta}\ Eq_1 \qquad \frac{(\epsilon, w' \triangleright w'); \Gamma[w'/w] \vdash \Delta[w'/w]}{(\epsilon, w' \triangleright w); \Gamma \vdash \Delta}\ Eq_2$$

Side conditions:
In $\top^* L$, Eq_1 and Eq_2, $w \neq \epsilon$.
In $*L$ and $\mathrel{-\!\!*} R$, the labels x and y do not occur in the conclusion.
In A and A_C, the label w does not occur in the conclusion.

Figure 1: The labelled sequent calculus LS_{BBI} for Boolean BI.

Theorem 2.1 (Soundness)**.** *For any label variable $w \in LVAR$ and any BBI formula F, if $\vdash w : F$ is derivable in LS_{BBI}, then F is valid in BBI_{ND}.*

Proof. The condition $w \in LVAR$ simply says that w cannot be the label constant ϵ but $\rho(w)$ can be the world ϵ. Suppose $\vdash w : F$ is derivable but F is invalid, i.e., there is some model $(\mathcal{M}, \triangleright, \epsilon, v)$ where

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots;w_2:r\vdash w_2:r;\cdots}\,{\scriptstyle id}\qquad\dfrac{}{\cdots;w_1:q\vdash w_1:q;\cdots}\,{\scriptstyle id}}{(w_2,w_1\triangleright w_0);(\epsilon,w_1\triangleright w_1);(w_1,w_2\triangleright w_0);\epsilon:p;\epsilon:\top^*;w_1:q;w_2:r\vdash w_0:r*q}\,{\scriptstyle *R}}{(\epsilon,w_1\triangleright w_1);(w_1,w_2\triangleright w_0);\epsilon:p;\epsilon:\top^*;w_1:q;w_2:r\vdash w_0:r*q}\,{\scriptstyle E}}{(\epsilon,w_4\triangleright w_1);(w_1,w_2\triangleright w_0);\epsilon:p;\epsilon:\top^*;w_4:q;w_2:r\vdash w_0:r*q}\,{\scriptstyle Eq_1}}{(w_3,w_4\triangleright w_1);(w_1,w_2\triangleright w_0);w_3:p;w_3:\top^*;w_4:q;w_2:r\vdash w_0:r*q}\,{\scriptstyle \top^*L}}{(w_3,w_4\triangleright w_1);(w_1,w_2\triangleright w_0);w_3:p\wedge\top^*;w_4:q;w_2:r\vdash w_0:r*q}\,{\scriptstyle \wedge L}}{w_0:((p\wedge\top^*)*q)*r\vdash w_0:r*q}\,{\scriptstyle *L\times 2}}{\vdash w_0:(((p\wedge\top^*)*q)*r)\to(r*q)}\,{\scriptstyle \to R}$$

Figure 2: An example derivation for $(((p \wedge \top^*) * q) * r) \to (r * q)$ in LS_{BBI}.

Axioms

$A \to (\top^* * A)$

$(\top^* * A) \to A$

$(A * B) \to (B * A)$

$(A * (B * C)) \to ((A * B) * C)$

Deduction Rules

$$\dfrac{\vdash A \qquad \vdash A \to B}{\vdash B}\; MP \qquad \dfrac{\vdash A \to C \qquad \vdash B \to D}{\vdash (A * B) \to (C * D)}\; *$$

$$\dfrac{\vdash A \to (B -\!\!* \, C)}{\vdash (A * B) \to C}\; -\!\!*\, 1 \qquad \dfrac{\vdash (A * B) \to C}{\vdash A \to (B -\!\!*\, C)}\; -\!\!*\, 2$$

Figure 3: Some axioms and rules for the Hilbert system for BBI.

$\exists m \in \mathcal{M}$ such that $m \not\Vdash F$. This also means that there is a label mapping ρ, where $\rho(w) = m$, such that $\vdash w : F$ is falsifiable in this model under ρ. We then show that each rule preserves falsifiability upwards. That is, if the conclusion is falsifiable, then at least one of the premises is falsifiable (usually in the same choice of v, ρ, and $\mathcal{M}$). As the rules in LS_{BBI} are designed based on the semantics, this is easy to verify. The details are in Appendix A.1. As $\vdash w : F$ is derivable, each branch must end with a zero-premise rule, the conclusion of which is not falsifiable in any model. This contradicts the assumption that $\vdash w : F$ is falsifiable. Therefore there is no model and no label mapping that falsifies $\vdash w : F$. So given any model, no matter where we map w to, F is always true in that world. Thus F is valid. □

2.4 Completeness

We prove the completeness of LS_{BBI} by showing that every derivation of a formula in the Hilbert system for BBI [10] can be mimicked in LS_{BBI}, possibly using cuts.

The Hilbert system for BBI consists of the axioms and rules for classical propositional logic for the additive fragment and additional axioms and rules for the multiplicative fragment. For the latter, we use the axiomatisation given in [10], and listed in Figure 3. We omit the axioms for classical propositional logic as they are standard, and can be found in, e.g., [24].

Theorem 2.2 (Completeness). *For any label $w \in LVAR$ and any formula F of BBI, if F is valid then $\vdash w : F$ is derivable in LS_{BBI}.*

Proof. Again, it is not possible for the label variable w to be the label constant ϵ. As the Hilbert system for BBI is complete, there is a derivation Π of F in the Hilbert system. We show that one

$$\dfrac{\dfrac{}{\Gamma \vdash w_1 : A}\, id \qquad \dfrac{\dfrac{}{\Gamma; w_1 : B -\!\!* C \vdash w_2 : B}\, id \qquad \dfrac{}{\Gamma; w_1 : B -\!\!* C; w : C \vdash w : C}\, id}{\Gamma; w_1 : B -\!\!* C \vdash w : C}\, -\!\!* L}{\Gamma; w_1 : A \to (B -\!\!* C) \vdash w : C}\, \to L$$

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_1'\\ \vdash w_1 : A \to (B -\!\!* C)\end{matrix} \qquad \begin{matrix}\Pi_2\\ (w_1, w_2 \rhd w); w_1 : A \to (B -\!\!* C); w_1 : A; w_2 : B \vdash w : C\end{matrix}}{(w_1, w_2 \rhd w); w_1 : A; w_2 : B \vdash w : C}\, cut}{w : A * B \vdash w : C}\, *L}{\vdash w : (A * B) \to C}\, \to R$$

Figure 4: A derivation of the rule $-\!\!*$ 1 putting $\Gamma = \{(w_1, w_2 \rhd w); w_1 : A; w_2 : B\})$.

can construct an LS_{BBI} derivation Π' of the sequent $\vdash w : F$, for any label $w \neq \epsilon$. It is enough to show that each axiom and each rule of the Hilbert system can be derived. The derivations of the axioms in LS_{BBI} are straightforward; we show here a non-trivial case in the derivation of the rules of the Hilbert system. Consider the rule $-\!\!*$ 1, suppose Π is the derivation:

$$\dfrac{\begin{matrix}\Pi_1\\ A \to (B -\!\!* C)\end{matrix}}{(A * B) \to C}\, -\!\!* 1$$

The LS_{BBI} derivation Π' is in Figure 4, where Π_1' comes from Π_1 via the induction hypothesis, Π_2 is the upper derivation in Figure 4, and $\Gamma = \{(w_1, w_2 \rhd w); w_1 : A; w_2 : B\})$. ☐

Corollary 2.3 (Formula validity). *For any label variable $w \in LVAR$ and any BBI formula A, the formula A is valid iff $\vdash w : A$ is derivable in LS_{BBI}.*

Proof. Follows from the soundness and completeness proof. Since w is arbitrary, A is true at any world for any valuation v, mapping ρ, and monoid structure $(\mathcal{M}, \rhd, \epsilon)$. ☐

We compare our proofs of soundness and completeness with those of Larchey-Wendling and Galmiche [14, 13] in Section 8.

3 Cut-elimination

This section proves the cut-elimination theorem for our labelled sequent calculus. The general proof outlined here is similar to the cut-elimination proof for labelled systems for modal logic [17], i.e., we start by proving a substitution lemma for labels, followed by proving the invertibility of inference rules, weakening admissibility, and contraction admissibility, before proceeding to the main cut-elimination proof. As there are many case analyses in these proofs, we only outline the important parts here. More details are available in Appendix A

Given a derivation Π, its *height* $ht(\Pi)$ is defined as the length of a longest branch Π. The substitution lemma shows that provability is preserved under arbitrary substitutions of labels. When proving this lemma, we will frequently use the fact that, when applied on a sequent, the series of substitutions $[y/x][z/y]$ have the same effect as $[z/x][z/y]$, for any label z and any non-ϵ labels x, y.

Lemma 3.1 (Substitution)**.** *If Π is an LS_{BBI} derivation for the sequent $\Gamma \vdash \Delta$ then there is an LS_{BBI} derivation Π' of the sequent $\Gamma[y/x] \vdash \Delta[y/x]$ where every occurrence of label x ($x \neq \epsilon$) is replaced by label y, such that $ht(\Pi') \leq ht(\Pi)$.*

Proof. By induction on $ht(\Pi)$.
(Base case) If $ht(\Pi) = 0$, then the only applicable rules are id, $\bot L$, $\top R$ and $\top^* R$. If the label $x \neq \epsilon$ being substituted is not on the principal formula, then the substitution does not affect the original derivation. Since we do not allow to substitute for ϵ, the proof for $\top^* R$ can only be this case. Otherwise we obtain the new derivation by simply replacing the label of the principal formula.
(Inductive case) If $ht(\Pi) > 0$, then consider the last rule applied in the derivation. We consider three main cases.

1. Neither x nor y is the label of the principal formula.

   (a) Suppose the last rule applied is $\top^* L$, and $x \neq w$ and $y \neq w$, and Π runs as below:

   $$\dfrac{\begin{array}{c}\Pi_1 \\ \Gamma'[\epsilon/w] \vdash \Delta[\epsilon/w]\end{array}}{\Gamma'; w : \top^* \vdash \Delta}\ \top^* L$$

   By the induction hypothesis, there is a derivation Π_1' of $\Gamma'[\epsilon/w][y/x] \vdash \Delta[\epsilon/w][y/x]$ with $ht(\Pi_1') \leq ht(\Pi_1)$. Since x and y are different from w, this sequent equals $\Gamma'[y/x][\epsilon/w] \vdash \Delta[y/x][\epsilon/w]$. Therefore Π' is constructed as follows, where $ht(\Pi') \leq ht(\Pi)$.

   $$\dfrac{\begin{array}{c}\Pi_1' \\ \Gamma'[y/x][\epsilon/w] \vdash \Delta[y/x][\epsilon/w]\end{array}}{\Gamma'[y/x]; w : \top^* \vdash \Delta[y/x]}\ \top^* L$$

   (b) If the last rule applied is Eq_1, we distinguish the following cases: x is not w or w'; $x = w$; $x = w'$. The analysis is similar to above, but is more complicated. If the last rule applied is Eq_2, we consider three cases: $x \neq w$ and $y \neq w$; $x = w$; and $y = w$. These are symmetric to the case where the last rule is Eq_1. See Appendix A.2 for details.

   (c) For other rules, we can simply use the induction hypothesis, and apply the corresponding rule to obtain the required derivation.

2. y is the label of the principal formula. Most of the cases follow similarly as above, except for $\top^* L$. In this case the original derivation is as follows.

   $$\dfrac{\begin{array}{c}\Pi_1 \\ \Gamma'[\epsilon/y] \vdash \Delta[\epsilon/y]\end{array}}{\Gamma'; y : \top^* \vdash \Delta}\ \top^* L$$

   Our goal is to derive $\Gamma'[y/x]; y : \top^* \vdash \Delta[y/x]$. Applying $\top^* L$ backwards to it we get

   $$\Gamma'[y/x][\epsilon/y] \vdash \Delta[y/x][\epsilon/y]$$

Note that this sequent is equal to $\Gamma'[\epsilon/y][\epsilon/x] \vdash \Delta[\epsilon/y][\epsilon/x]$, and from the induction hypothesis we know that there is a derivation of this sequent of height less than or equal to $ht(\Pi)$.

3. x is the label of the principal formula.

   (a) Additive rules do not modify labels, so even if the label of the principal formula is replaced by some other label, we can still apply the induction hypothesis on the premise, then use the rule to derive the conclusion.

   (b) For the multiplicative rules that do not produce eigenvariables[2] ($*R, -\!\!* \ L, \top^*L$), we can proceed similarly as in the additive cases, except for the $\top^*L$ rule. For the $\top^*L$ rule, if the label x of the principal formula is replaced by some (other) label y, i.e., Π is

$$\dfrac{\begin{matrix}\Pi_1\\ \Gamma'[\epsilon/x] \vdash \Delta[\epsilon/x]\end{matrix}}{\Gamma'; x : \top^* \vdash \Delta}\ \top^*L$$

   then we need to derive $\Gamma'[y/x]; y : \top^* \vdash \Delta[y/x]$. Using $\top^*L$ rule we have:

$$\dfrac{\Gamma'[y/x][\epsilon/y] \vdash \Delta[y/x][\epsilon/y]}{\Gamma'[y/x]; y : \top^* \vdash \Delta[y/x]}\ \top^*L$$

   Note that the premise now is equal to $\Gamma'[\epsilon/x][\epsilon/y] \vdash \Delta[\epsilon/x][\epsilon/y]$, and can be proved using the induction hypothesis on Π_1.

   If $y = \epsilon$, then Π' is the same as Π, because $\epsilon : \top^*$ in the antecedent can never be used in any rule applications, it will remain unchanged until the branch is closed.

   (c) For the multiplicative rules that have eigenvariables ($*L$ and $-\!\!* \ R$), if the label of the principal formula is replaced by a label other than the newly created labels in the rules, then we proceed similarly as in the additive cases. If the label of the principal formula is replaced by one of the newly created labels, then we just need to create a different new label in the new relation.

   For $*L$, we have the derivation:

$$\dfrac{\begin{matrix}\Pi_1\\ (y, z \triangleright x); \Gamma'; y : A; z : B \vdash \Delta\end{matrix}}{\Gamma'; x : A * B \vdash \Delta}\ *L$$

   If x is substituted by y (the case for substituting by z is symmetric), then we need a derivation of $\Gamma'[y/x]; y : A * B \vdash \Delta[y/x]$. Note that the $*L$ rule requires the relation $(y, z \triangleright x)$ to be fresh, so in the original derivation y and z cannot be in Γ' or Δ. Therefore by the induction hypothesis we must have a derivation Π'_1 for

$$(y', z' \triangleright x); \Gamma'; y' : A; z' : B \vdash \Delta,$$

   where y' and z' are new labels, such that $ht(\Pi'_1) \leq ht(\Pi_1)$. Applying the induction hypothesis again to Π'_1, we have a derivation Π''_1 $(y', z' \triangleright y); \Gamma'[y/x]; y' : A; z' : B \vdash \Delta[y/x]$, with $ht(\Pi''_1) \leq ht(\Pi_1)$. Thus the derivation Π' is constructed as follows.

[2] i.e., those labels that are required to be fresh in the premise.

$$\dfrac{\begin{array}{c}\Pi_1''\\(y', z' \triangleright y); \Gamma'[y/x]; y' : A; z' : B \vdash \Delta[y/x]\end{array}}{\Gamma'[y/x]; y : A * B \vdash \Delta[y/x]} *L$$

The case for $-\!\!* \; R$ is similar. □

Admissibility of weakening is proved by a simple induction on the height of derivations so we state the lemma without proof.

Lemma 3.2 (Weakening admissibility). *If $\Gamma \vdash \Delta$ is derivable in LS_{BBI}, then for all structures Γ' and Δ', the sequent $\Gamma; \Gamma' \vdash \Delta; \Delta'$ is derivable with the same height in LS_{BBI}.*

The converse of weakening admissibility, i.e., a certain 'strengthening' property, also holds for certain formulas. More precisely, if a sequent $\Gamma; w : A \vdash \Delta$ is derivable, and the labelled formula $w : A$ is not principal in any rule application, then $\Gamma \vdash \Delta$ is also derivable by the same derivation. By weakening, the latter implies that $\Gamma; \Gamma' \vdash \Delta$ is also derivable for any Γ'. That is, we can replace an unused labelled formula by an arbitrary structure. This and other supplementary lemmas related to weakening are proved in Appendix A.3.

Lemma 3.3 (Invertibility of rules). *If Π is a cut-free LS_{BBI} derivation of the conclusion of a rule then there is a cut-free LS_{BBI} derivation for each premise, with height at most $ht(\Pi)$.*

Proof. Most of the rules are trivially invertible. The proofs for the additive rules are similar to those for the additive rules from labelled calculi for modal logic or $G3c$ (cf. [18]) since the rules are the same. The slightly non-trivial cases for the rules involving substitutions of labels follow from Lemma 3.1. The proof is detailed in Appendix A.4. □

The proof of the admissibility of contraction on additive formulae is similar to that for classical sequent calculus since the LS_{BBI} rules for these connectives are the same. In the multiplicative rules, the principal formula is retained in the premise, so admissibility of contraction on multiplicative formulae follows trivially. We need to prove that contraction on relational atoms is admissible, as stated in the next lemma.

Lemma 3.4. *For any structures Γ, Δ, and relational atom $(x, y \triangleright z)$: if Π a cut-free LS_{BBI} derivation of $(x, y \triangleright z); (x, y \triangleright z); \Gamma \vdash \Delta$, then there is a cut-free LS_{BBI} derivation Π' of $(x, y \triangleright z); \Gamma \vdash \Delta$ with $ht(\Pi') \leq ht(\Pi)$.*

Proof. (Outline) Let $n = ht(\Pi)$. The proof is by induction on n. Most structural rules only have one principal relational atom, so it is easy to show that contraction can permute through them.

The case for A needs more care, as it involves two principal relational atoms. If the two principal relational atoms are different, then the admissibility of contraction follows similarly as above. But if the principal relational atoms are identical, the situation is a bit tricky:

$$\dfrac{\begin{array}{c}\Pi\\(x, w \triangleright x); (y, y \triangleright w); (x, y \triangleright x); (x, y \triangleright x); \Gamma \vdash \Delta\end{array}}{(x, y \triangleright x); (x, y \triangleright x); \Gamma \vdash \Delta} A$$

There is no obvious way to make this case admissible, and this is the reason we have a special case of the rule A, namely A_C. In the rule A_C, contraction is absorbed so that there is only one principal relational atom. The new derivation is as follows.

$$\dfrac{\begin{array}{c}\Pi' \\ (x, w \triangleright x); (y, y \triangleright w); (x, y \triangleright x); \Gamma \vdash \Delta\end{array}}{(x, y \triangleright x); \Gamma \vdash \Delta}\ A_C$$

For Eq_1 and Eq_2, as the principal relational atom is carried to the premise (although some labels may be changed), so admissibility of contraction on those relational atoms is obvious. □

The admissibility of contraction on formulae are straightforward, the most of cases are analogous to the ones in Negri's labelled calculus for modal logic [17]. For details please see Appendix A.5.

Lemma 3.5 (Contraction admissibility)**.** *If* $\Gamma; \Gamma \vdash \Delta; \Delta$ *is derivable in* LS_{BBI}*, then* $\Gamma \vdash \Delta$ *is derivable with the same height in* LS_{BBI}*.*

Cut Elimination Theorem

We define the *cut rank* of an application of the *cut* rule as the pair $(|f|, ht(\Pi_1) + ht(\Pi_2))$, where $|f|$ denotes the size of the cut formula (i.e., the number of connectives in the formula), and $ht(\Pi_1)$, $ht(\Pi_2)$ are the heights of the derivations above the *cut* rule, the sum of them is call the *cut height*. Cut ranks are ordered lexicographically, where each component of the ranks are ordered according to the ordering $>$ on natural numbers.

Theorem 3.6 (Cut-elimination)**.** *If* $\Gamma \vdash \Delta$ *is derivable in* LS_{BBI}*, then it is also derivable without using the cut rule.*

Proof. By induction on the cut ranks of the proof in LS_{BBI}. We show that each application of *cut* can either be eliminated, or be replaced by one or more *cut* rules of smaller ranks. The argument for termination is similar to the cut-elimination proof for $G3ip$ [18]. We start to eliminate the topmost *cut* first, and repeat this procedure until there is no *cut* in the derivation. We first show that *cut* can be eliminated when the *cut height* is the lowest, i.e., at least one premise is of height 1. Then we show that the *cut height* is reduced in all cases in which the cut formula is not principal in both premises of cut. If the cut formula is principal in both premises, then the *cut* is reduced to one or more *cut*s on smaller formulae or shorter derivations. Since atoms cannot be principal in logical rules, finally we can either reduce all *cut*s to the case where the cut formula is not principal in both premises, or reduce those *cut*s on compound formulae until their *cut height*s are minimal and then eliminate those *cut*s.

We show here some non-trivial cut elimination steps for the inductive cases; the full proof can be found in Appendix A.6.

Suppose neither premise of the cut is an instance of id, $\bot L$, $\top R$, or $\top^* R$. We distinguish three cases here: the cut formula is not principal in the left premises; the cut formula is only principal in the left premise; and the cut formula is principal in both premises.

1. The cut formula is not principal in the left premise, which ends with a rule r.

   (a) If r is $\top^* L$, w.l.o.g. we assume the label of the principal formula is y (which might be equal to x). The original derivation is as follows.

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1 \\ \Gamma[\epsilon/y] \vdash x : A; \Delta[\epsilon/y]\end{array}}{\Gamma; y : \top^* \vdash x : A; \Delta}\ \top^* L \qquad \begin{array}{c}\Pi_2 \\ \Gamma'; x : A \vdash \Delta'\end{array}}{\Gamma; \Gamma'; y : \top^* \vdash \Delta; \Delta'}\ cut$$

By the Substitution lemma, there is a derivation Π'_2 of $\Gamma'[\epsilon/y]; x : A \vdash \Delta[\epsilon/y]$. Thus we can transform the derivation into the following:

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma[\epsilon/y] \vdash x : A; \Delta[\epsilon/y]\end{matrix} \qquad \begin{matrix}\Pi'_2\\ \Gamma'[\epsilon/y]; x : A \vdash \Delta'[\epsilon/y]\end{matrix}}{\Gamma[\epsilon/y]; \Gamma'[\epsilon/y] \vdash \Delta[\epsilon/y]; \Delta'[\epsilon/y]}\,\mathit{cut}}{\Gamma; \Gamma'; y : \top^* \vdash \Delta; \Delta'}\,\top^* L$$

If $x = y$ in the original derivation, then the new derivation cuts on $\epsilon : A$ instead. As substitution is height preserving, the cut height in this case is reduced as well.

(b) If r is Eq_1, and the label x of the principal formula is not equal to w', the original derivation is as follows.

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ (\epsilon, w \triangleright w); \Gamma[w/w'] \vdash x : A; \Delta[w/w']\end{matrix}}{(\epsilon, w' \triangleright w); \Gamma \vdash x : A; \Delta}\,Eq_1 \qquad \begin{matrix}\Pi_2\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{(\epsilon, w' \triangleright w); \Gamma; \Gamma' \vdash \Delta; \Delta'}\,\mathit{cut}$$

This *cut* is reduced in the same way as the $\top^* L$ case, where we get Π'_2 from the Substitution Lemma:

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ (\epsilon, w \triangleright w); \Gamma[w/w'] \vdash x : A; \Delta[w/w']\end{matrix} \qquad \begin{matrix}\Pi'_2\\ \Gamma'[w/w']; x : A \vdash \Delta'[w/w']\end{matrix}}{(\epsilon, w \triangleright w); \Gamma[w/w']; \Gamma'[w/w'] \vdash \Delta[w/w']; \Delta'[w/w']}\,\mathit{cut}}{(\epsilon, w' \triangleright w); \Gamma; \Gamma' \vdash \Delta; \Delta'}\,Eq_1$$

If $x = w'$, then we cut on $w : A$ instead in the reduced version.

(c) If r is Eq_2, the procedure follows similarly as the case for Eq_1 above.

(d) If r is a unary rule except for $\top^* L$, Eq_1, and Eq_2, the proof is transformed as follows:

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma_1 \vdash x : A; \Delta_1\end{matrix}}{\Gamma \vdash x : A; \Delta}\,r \qquad \begin{matrix}\Pi_2\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,\mathit{cut} \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma_1 \vdash x : A; \Delta_1\end{matrix} \qquad \begin{matrix}\Pi_2\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{\Gamma_1; \Gamma' \vdash \Delta_1; \Delta'}\,\mathit{cut}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,r$$

Note that as all our rules except $\top^* L$, Eq_1, and Eq_2 do not modify side structures, Γ' and Δ' in the premise of r are not changed. The cut rank of the original *cut* is $(|x : A|, |\Pi_1| + 1 + |\Pi_2|)$, whereas the cut rank of the new *cut* is $(|x : A|, |\Pi_1| + |\Pi_2|)$, so the *cut height* reduces.

(e) If r is a binary inference, we can transform the derivation similarly.

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma_1 \vdash x : A; \Delta_1\end{matrix} \qquad \begin{matrix}\Pi_2\\ \Gamma_2 \vdash x : A; \Delta_2\end{matrix}}{\Gamma \vdash x : A; \Delta}\,r \qquad \begin{matrix}\Pi_3\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,\mathit{cut} \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma_1 \vdash x : A; \Delta_1\end{matrix} \qquad \begin{matrix}\Pi_3\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{\Gamma_1; \Gamma' \vdash \Delta_1; \Delta'}\,\mathit{cut} \qquad \dfrac{\begin{matrix}\Pi_2\\ \Gamma_2 \vdash x : A; \Delta_2\end{matrix} \qquad \begin{matrix}\Pi_3\\ \Gamma'; x : A \vdash \Delta'\end{matrix}}{\Gamma_2; \Gamma' \vdash \Delta_2; \Delta'}\,\mathit{cut}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,r$$

The cut rank of the original *cut* is $(|x : A|, max(|\Pi_1|, |\Pi_2|) + 1 + |\Pi_3|)$, and that of the new two *cut*s are $(|x : A|, |\Pi_1| + |\Pi_3|)$ and $(|x : A|, |\Pi_2| + |\Pi_3|)$ respectively. Thus the cut heights are reduced.

2. The cut formula is only principal in the left premise. We only consider the last rule in the right branch. The proof of this case is symmetric to those in Case 1.

3. The cut formula is principal in both premises. We do a case analysis on the main connective of the cut formula. If the main connective is additive, then there is no need to substitute any labels. We only show the case for $\wedge$ here.

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma \vdash x : A; \Delta\end{matrix} \qquad \begin{matrix}\Pi_2\\ \Gamma \vdash x : B; \Delta\end{matrix}}{\Gamma \vdash x : A \wedge B; \Delta}\wedge R \qquad \dfrac{\begin{matrix}\Pi_3\\ \Gamma'; x : A; x : B \vdash \Delta'\end{matrix}}{\Gamma'; x : A \wedge B \vdash \Delta'}\wedge L \rightsquigarrow}{\Gamma; \Gamma' \vdash \Delta; \Delta'}cut$$

$$\dfrac{\dfrac{\begin{matrix}\Pi_1\\ \Gamma \vdash x : A; \Delta\end{matrix} \qquad \dfrac{\begin{matrix}\Pi_2\\ \Gamma \vdash x : B; \Delta\end{matrix} \qquad \begin{matrix}\Pi_3\\ \Gamma'; x : A; x : B \vdash \Delta'\end{matrix}}{\Gamma; \Gamma'; x : A \vdash \Delta; \Delta'}cut}{\Gamma; \Gamma; \Gamma' \vdash \Delta; \Delta; \Delta'}cut}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\text{Lemma 3.5}$$

In this case, *cut* is reduced to applications on smaller formulae, therefore the cut rank of the *cut* rule reduces.

There is an asymmetry in the rules for $\top^*$. That is, the left rule for $\top^*$ requires that the label w of $\top^*$ cannot be ϵ, whereas the right rule for $\top^*$ restricts the label of $\top^*$ to be ϵ only. As a consequence, when the cut formula is $\top^*$, it cannot be the principal formula of both premises at the same time. Therefore the cases for $\top^*$ are covered in the proof above.

When the main connective of the cut formula is $*$ or $-\!\!*$, the case is more complicated. For $*$, we have the following two derivations as the premises of the *cut* rule:

$$\dfrac{\begin{matrix}\Pi_1\\ (x, y \triangleright z); \Gamma \vdash x : A; z : A * B; \Delta\end{matrix} \qquad \begin{matrix}\Pi_2\\ (x, y \triangleright z); \Gamma \vdash y : B; z : A * B; \Delta\end{matrix}}{(x, y \triangleright z); \Gamma \vdash z : A * B; \Delta} {\scriptstyle *R}$$

and

$$\dfrac{\begin{matrix}\Pi_3\\ (x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'\end{matrix}}{\Gamma'; z : A * B \vdash \Delta'} {\scriptstyle *L}$$

And the *cut* rule gives the end sequent $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$. The cut rank of this *cut* is $(|A * B|, max(|\Pi_1|, |\Pi_2|) + 1 + |\Pi_3| + 1)$.

We use several *cut*s with smaller ranks to derive $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$ as follows. Firstly,

$$\dfrac{\dfrac{\Pi_1}{(x, y \triangleright z); \Gamma \vdash x : A; z : A * B; \Delta} \quad \dfrac{\dfrac{\Pi_3}{(x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'}}{\Gamma'; z : A * B \vdash \Delta'} {*L}}{(x, y \triangleright z); \Gamma; \Gamma' \vdash x : A; \Delta; \Delta'} \; cut$$

The cut rank of this *cut* is $(|A * B|, |\Pi_1| + |\Pi_3| + 1))$, thus is less than the original *cut*.

The second *cut* works similarly.

$$\dfrac{\dfrac{\Pi_2}{(x, y \triangleright z); \Gamma \vdash y : B; z : A * B; \Delta} \quad \dfrac{\dfrac{\Pi_3}{(x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'}}{\Gamma'; z : A * B \vdash \Delta'} {*L}}{(x, y \triangleright z); \Gamma; \Gamma' \vdash y : B; \Delta; \Delta'} \; cut$$

The third *cut* works on a smaller formula.

$$\dfrac{(x, y \triangleright z); \Gamma; \Gamma' \vdash x : A; \Delta; \Delta' \quad \dfrac{\Pi_3'}{(x, y \triangleright z); \Gamma'; x : A; y : B \vdash \Delta'}}{(x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma'; \Gamma'; y : B \vdash \Delta; \Delta'; \Delta'} \; cut$$

The cut formula is $x : A$, thus the cut rank of this *cut* is less regardless of the height of the derivations. Note that in the Π_3 branch, the $*L$ rule requires that the relation $(x', y' \triangleright z)$ is newly created, so x' and y' cannot be ϵ and they cannot be in Γ' or Δ'. Therefore we are allowed to use the substitution lemma to get a derivation Π_3' of $(x, y \triangleright z); \Gamma'; x : A; y : B \vdash \Delta'$ by just substituting x' for x and y' for y.

Finally we cut on another smaller formula $y : B$.

$$\dfrac{(x, y \triangleright z); \Gamma; \Gamma' \vdash y : B; \Delta; \Delta' \quad (x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma'; \Gamma'; y : B \vdash \Delta; \Delta'; \Delta'}{(x, y \triangleright z); (x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma; \Gamma'; \Gamma'; \Gamma' \vdash \Delta; \Delta; \Delta'; \Delta'; \Delta'} \; cut$$

The cut rank of this *cut* is less than the original *cut*. We then apply the admissibility of contraction to derive $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$. The case for $-\!\!*$ is similar. □

4 Localising structural rules

Structural rules such as E, A, U can be applied to any sequent that contains suitable relational atoms, just as weakening and contraction in LK can be applied to any sequent that contains at least one formula. Thus each structural rule creates many non-deterministic choices in proof search. One way to reduce this non-determinism is to separate the structural rules into a constraint system as outlined in Section 5. We proceed via the intermediate calculi detailed in this section.

As a first step towards designing an effective proof search procedure for LS_{BBI}, we need to restrict the use of these structural rules so that their applications in proof search are driven by logical rules, thereby reducing the non-determinism in proof search.

We note the fact that the structural rules in LS_{BBI} can permute upwards through all other rules except for *id*, $\top^* R$, $*R$, and $-\!\!* L$. We refer to these four rules as *positive rules*, and refer to the other logical rules in LS_{BBI} as *negative rules*. The main reason is, all negative rules do not rely on relational atoms. This is formalised in the following lemma, and proved in Appendix A.7.

Lemma 4.1. *The structural rules in LS_{BBI} can permute upwards through negative rules in LS_{BBI}.*

Now we design a more compact proof system where applications of structural rules are separated into a special entailment relation for relational atoms and equivalence of labels. Such a special entailment can be seen as a "condition" in a rule application. To be specific, since all the structural rule applications can permute upwards until they meet positive rule applications, we would apply negative rules, backwards, as much as possible in the proof search, and only apply structural rules, backwards, when they are required by positive rules. In this sense, structural rule applications are encapsulated in positive rule applications as "conditions" of the form "to apply this positive rule, some structural rules have to be applied first". We shall see in the next section that proof search in this proof system can be separated into two phases: guessing the shape of the proof tree, and deriving the relational atoms needed. The latter will be phrased using a constraint system.

In this section we localise the structural rules in two steps: we first deal with Eq_1 and Eq_2, and then the other structural rules.

4.1 Localising Eq_1 and Eq_2

Allowing substitutions in a proof rule simplifies the cut-elimination proof for LS_{BBI}. However, for proof search, this creates a problem as Eq_1 and Eq_2 do not permute over certain rules that require matching of two labels (e.g., $*R$ or $-\!\!* \, L$). Our first intermediate proof system LS^e_{BBI} aims to remove substitutions from LS_{BBI}. Instead, the equality between labels is captured via a special entailment relation. To define its inference rules, we first need a few preliminary definitions.

A structural rule r can be seen as defining a relation $\mathcal{R}$ as follows: $(\mathcal{G}_1, \theta, \mathcal{V}, \mathcal{G}_2) \in \mathcal{R}$ iff there is an instance of r such that

- the set of principal relational atoms (cf. Section 2.2 for the definition of principal relational atoms) of the instance is $\mathcal{G}_1$;
- the substitution applied to the premise of the instance is θ;
- the set of fresh labels created in the premise of the instance is $\mathcal{V}$; and
- the set of new relational atoms in the premise of the instance is $\mathcal{G}_2$.

In the following, we shall write $r(\mathcal{G}_1, \theta, \mathcal{V}, \mathcal{G}_2)$ to denote that $(\mathcal{G}_1, \theta, \mathcal{V}, \mathcal{G}_2) \in \mathcal{R}$ as defined above. For example, we have both $U(\{\}, [\,], \{x\}, \{(x, \epsilon \triangleright x)\})$ and $U(\{\}, [\,], \{\}, \{(x, \epsilon \triangleright x)\})$, which are justified respectively by the following instances of U:

$$\frac{(x, \epsilon \triangleright x); (a, b \triangleright c) \vdash a : F}{(a, b \triangleright c) \vdash a : F}\, U \qquad \frac{(x, \epsilon \triangleright x); (x, y \triangleright z) \vdash y : G}{(x, y \triangleright z) \vdash y : G}\, U$$

Note that the rule U does not restrict x to be among the labels occuring in the conclusion, so one can introduce a new label. Similarly, we have $A(\{(x, y \triangleright z), (u, v \triangleright x)\}, [\,], \{w\}, \{(u, w \triangleright z), (y, v \triangleright w)\})$ which is justified by, e.g., the following instance of A:

$$\frac{(x, y \triangleright z); (u, v \triangleright x); (u, w \triangleright z), (y, v \triangleright w); x : F \vdash y : G}{(x, y \triangleright z); (u, v \triangleright x); x : F \vdash y : G}\, A$$

and $Eq_1(\{(e, w \triangleright w')\}, [w'/w], \{\ \}, \{\ \})$, which is justified by, e.g.,

$$\frac{(\epsilon, w' \triangleright w'); w' : F \vdash w' : G}{(\epsilon, w \triangleright w'); w : F \vdash w' : G}\, Eq_1$$

$$\frac{\mathcal{G} \vdash_E (w_1 = w_2)}{\mathcal{G}||\Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle id} \qquad \frac{(\epsilon, w \triangleright \epsilon); \mathcal{G} \vdash \Delta}{\mathcal{G}||\Gamma; w : \top^* \vdash \Delta}\ {\scriptstyle \top^* L} \qquad \frac{\mathcal{G} \vdash_E (w = \epsilon)}{\mathcal{G}||\Gamma \vdash w : \top^*; \Delta}\ {\scriptstyle \top^* R}$$

$$\frac{(x, y \triangleright z'); \mathcal{G}'||\Gamma \vdash x : A; z : A * B; \Delta \qquad (x, y \triangleright z'); \mathcal{G}'||\Gamma \vdash y : B; z : A * B; \Delta \qquad \mathcal{G} \vdash_E (z = z')}{(x, y \triangleright z'); \mathcal{G}'||\Gamma \vdash z : A * B; \Delta}\ {\scriptstyle *R}$$

$$\frac{(x, y' \triangleright z); \mathcal{G}'||\Gamma; y : A \mathrel{-\!\!*} B \vdash x : A; \Delta \qquad (x, y' \triangleright z); \mathcal{G}'||\Gamma; y : A \mathrel{-\!\!*} B; z : B \vdash \Delta \qquad \mathcal{G} \vdash_E (y = y')}{(x, y' \triangleright z); \mathcal{G}'||\Gamma; y : A \mathrel{-\!\!*} B \vdash \Delta}\ {\scriptstyle -\!* L}$$

$$\frac{(u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \mathcal{G}'||\Gamma \vdash \Delta \qquad \mathcal{G} \vdash_E (x = x')}{(x, y \triangleright z); (u, v \triangleright x'); \mathcal{G}'||\Gamma \vdash \Delta}\ {\scriptstyle A}$$

$$\frac{(x, w \triangleright x'); (y, y \triangleright w); (x, y \triangleright x'); \mathcal{G}'||\Gamma \vdash \Delta \qquad \mathcal{G} \vdash_E (x = x')}{(x, y \triangleright x'); \mathcal{G}'||\Gamma \vdash \Delta}\ {\scriptstyle A_C}$$

$\mathcal{G}$ is the multiset of relational atoms in the left hand side of the conclusion sequent.
In each rule, the entailment $\vdash_E$ is not a premise, but a condition on the rule.

Figure 5: The changed rules in LS^e_{BBI}.

We call $r(\mathcal{G}_1, \theta, \mathcal{V}, \mathcal{G}_2)$ an *abstract instance* of the rule r. The set of abstract instances of structural rules are ranged over by $\mathbf{r}$.

Given a set of relational atoms $\mathcal{G}$, we denote with $LV(\mathcal{G})$ the set of label variables in $\mathcal{G}$. Let σ be a sequence (list) of abstract instances of structural rules $[\mathbf{r}_1; \cdots; \mathbf{r}_n]$. Given a set of relational atoms $\mathcal{G}$, the result of the application of σ to $\mathcal{G}$, denoted by $\mathcal{S}(\mathcal{G}, \sigma)$, is defined inductively as follows where @ stands for sequence (list) concatenation:

$$\mathcal{S}(\mathcal{G}, \sigma) \quad = \quad \begin{cases} \mathcal{G} & \text{if} \quad \sigma = [\,] \\ \mathcal{S}(\mathcal{G}\theta \cup \mathcal{G}_2, \sigma') & \text{if} \quad \mathcal{G}_1 \subseteq \mathcal{G}, \sigma = [r(\mathcal{G}_1, \theta, \mathcal{V}, \mathcal{G}_2)]@\sigma' \text{ and } LV(\mathcal{G}) \cap \mathcal{V} = \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

Given a sequence of structural rule applications $\sigma = [r_1(\mathcal{G}_1, \theta_1, \mathcal{V}_1, \mathcal{G}'_1); \cdots; r_n(\mathcal{G}_n, \theta_n, \mathcal{V}_n, \mathcal{G}'_n)]$, we denote with $\mathit{subst}(\sigma)$ the composite substitution $\theta_1 \circ \cdots \circ \theta_n$, where $t(\theta_1 \circ \theta_2)$ means $(t\theta_1)\theta_2$.

Definition 4.1. Let $\mathcal{G}$ be a set of relational atoms. The entailment relation $\mathcal{G} \vdash_E (u = v)$ holds iff there exists a sequence σ of abstract instances of Eq_1 or Eq_2 such that $\mathcal{S}(\mathcal{G}, \sigma)$ is defined, and $u\theta = v\theta$, where $\theta = \mathit{subst}(\sigma)$.

We now define the proof system LS^e_{BBI} as $LS_{BBI} \setminus \{Eq_1, Eq_2\}$ (i.e., LS_{BBI} without rules Eq_1, Eq_2) where certain rules are modified according to Figure 5. We write $\mathcal{G}||\Gamma \vdash \Delta$ to emphasize that the left hand side of a sequent is partitioned into two parts: $\mathcal{G}$, which contains only relational atoms, and Γ, which contains only labelled formulae. Note that the new $\top^* L$ rule does not modify any labels, instead, the relational atom $(\epsilon, w \triangleright \epsilon)$ in the premise ensures that the derivability of $(w = \epsilon)$ is preserved. The point of this intermediate step is to avoid label substitutions in the proof system.

Theorem 4.2. *A sequent $\Gamma \vdash \Delta$ is derivable in LS_{BBI} if and only if it is derivable in LS^e_{BBI}.*

Proof. (Outline) One direction, from LS^e_{BBI} to LS_{BBI} is straightforward, as $\vdash_E$ is essentially just a sequence of applications of Eq_1 and Eq_2. The other direction can be proved by showing that Eq_1

$$\frac{\mathcal{G} \vdash_R (w_1 = w_2)}{\mathcal{G}||\Gamma; w_1 : P \vdash w_2 : P; \Delta}\ id \qquad \frac{\mathcal{G} \vdash_R (w = \epsilon)}{\mathcal{G}||\Gamma \vdash w : \top^*; \Delta}\ \top^* R$$

$$\frac{\mathcal{S}(\mathcal{G},\sigma)||\Gamma \vdash x : A; w : A * B; \Delta \quad \mathcal{S}(\mathcal{G},\sigma)||\Gamma \vdash y : B; w : A * B; \Delta \quad \dfrac{\sigma}{\mathcal{G} \vdash_R (x, y \triangleright w)}}{\mathcal{G}||\Gamma \vdash w : A * B; \Delta}\ *R^{\dagger}$$

$$\frac{\mathcal{S}(\mathcal{G},\sigma)||\Gamma; w : A \mathrel{-\!\!*} B \vdash x : A; \Delta \quad \mathcal{S}(\mathcal{G},\sigma)||\Gamma; w : A \mathrel{-\!\!*} B; z : B \vdash \Delta \quad \dfrac{\sigma}{\mathcal{G} \vdash_R (x, w \triangleright z)}}{\mathcal{G}||\Gamma; w : A \mathrel{-\!\!*} B \vdash \Delta}\ \mathrel{-\!\!*} L^{\ddagger}$$

The entailment $\vdash_R$ is a condition rather than a premise, and some rules require its derivation σ.

Figure 6: Changed rules in LS^{sf}_{BBI}.

and Eq_2 are admissible in LS^{e}_{BBI}. Detailed proofs are given in Appendix A.8, A.9, for soundness and completeness respectively. $\square$

4.2 Localising the rest of the structural rules

As a second step, we isolate the other structural rules into a separate entailment relation, as we did with Eq_1 and Eq_2, giving another intermediate system LS^{sf}_{BBI}.

Definition 4.2 (Relation Entailment $\vdash_R$)**.** The entailment relation $\vdash_R$ has the following two forms:

1. $\mathcal{G} \vdash_R (w_1 = w_2)$ is true iff there is a sequence σ of abstract instances of E, U, A, A_C so that $\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w_1 = w_2)$.

2. $\mathcal{G} \vdash_R (w_1, w_2 \triangleright w_3)$ is true iff there is a sequence σ of abstract instances of E, U, A, A_C so that $(w_1', w_2' \triangleright w_3') \in \mathcal{S}(\mathcal{G},\sigma)$ and the following hold: $\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w_1 = w_1')$, $\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w_2 = w_2')$, and $\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w_3 = w_3')$.

In each case, we say that σ is a derivation of the entailment relation involved.

The entailment $\vdash_R$ is stronger than $\vdash_E$. For example, if $\mathcal{G}$ only contains $(x, \epsilon \triangleright y)$, then $\mathcal{G} \not\vdash_E (x = y)$; but $\mathcal{G} \vdash_R (x = y)$ by applying E to obtain $(\epsilon, x \triangleright y)$, then apply Eq_1 or Eq_2 on the new relational atom.

The changed rules in the second intermediate system LS^{sf}_{BBI} are given in Figure 6.

The following is an immediate result, the proof is divided in two parts for soundness and completeness, detailed in Appendix A.11 and A.12 respectively.

Theorem 4.3. *A sequent $\Gamma \vdash \Delta$ is derivable in LS^{e}_{BBI} if and only if it is derivable in LS^{sf}_{BBI}.*

As a consequence of Theorem 4.2 and 4.3, we can also obtain the equivalence between LS_{BBI} and LS^{sf}_{BBI}, the latter will be used in the next section. Figure 7 shows example derivations in the intermediate systems LS^{e}_{BBI} and LS^{sf}_{BBI} respectively in contrast to the derivation in Figure 2.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\cdots \vdash_E (w_2 = w_2)}{\cdots ; w_2 : r \vdash w_2 : r; \cdots}\,{\scriptstyle id} \qquad \dfrac{(\epsilon, w_3 \triangleright \epsilon); (w_3, w_4 \triangleright w_1); \cdots \vdash_E (w_4 = w_1)}{\cdots ; w_4 : q \vdash w_1 : q; \cdots}\,{\scriptstyle id}}{(w_2, w_1 \triangleright w_0); \cdots ; w_3 : p; w_3 : \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle *R}}{(\epsilon, w_3 \triangleright \epsilon); (w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0); w_3 : p; w_3 : \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle E}}{(w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0); w_3 : p; w_3 : \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle \top^* L}}{(w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0); w_3 : p \wedge \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle \wedge L}}{w_0 : ((p \wedge \top^*) * q) * r \vdash w_0 : r * q}\,{\scriptstyle *L \times 2}}{\vdash w_0 : (((p \wedge \top^*) * q) * r) \to (r * q)}\,{\scriptstyle \to R}
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\cdots \vdash_R (w_2 = w_2)}{(w_2, w_1 \triangleright w_0); \cdots ; || \cdots ; w_2 : r \vdash w_2 : r; \cdots}\,{\scriptstyle id} \qquad \dfrac{(\epsilon, w_3 \triangleright \epsilon); (w_3, w_4 \triangleright w_1); \cdots \vdash_R (w_4 = w_1)}{(w_2, w_1 \triangleright w_0); \cdots || \cdots ; w_4 : q \vdash w_1 : q; \cdots}\,{\scriptstyle id} \qquad C}{(\epsilon, w_3 \triangleright \epsilon); (w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0) || w_3 : p; w_3 : \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle *R}}{(w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0) || w_3 : p; w_3 : \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle \top^* L}}{(w_3, w_4 \triangleright w_1); (w_1, w_2 \triangleright w_0) || w_3 : p \wedge \top^*; w_4 : q; w_2 : r \vdash w_0 : r * q}\,{\scriptstyle \wedge L}}{w_0 : ((p \wedge \top^*) * q) * r \vdash w_0 : r * q}\,{\scriptstyle *L \times 2}}{\vdash w_0 : (((p \wedge \top^*) * q) * r) \to (r * q)}\,{\scriptstyle \to R}
$$

where $C ::= (w_1, w_2 \triangleright w_0); \cdots \vdash_R (w_2, w_1 \triangleright w_0)$

Figure 7: Derivations for $(((p \wedge \top^*) * q) * r) \to (r * q)$ in LS^{e}_{BBI} (top) and LS^{sf}_{BBI} (bottom).

5 Mapping proof search to constraint solving

We now consider a proof search strategy for LS^{sf}_{BBI}. As we have isolated all the structural rules into the entailment relation $\vdash_R$, proof search in LS^{sf}_{BBI} consists of guessing the shape of the derivation tree, and then checking that each entailment constraint $\vdash_R$ can be solved. The latter involves guessing a splitting of labels in the $*R$ and $-\!\!* \; L$ rules which also satisfies the equality constraints in the *id* and $\top^* R$ rules. We formalise this via a symbolic proof system, where constraint solving is handled lazily, via the introduction of *free variables* which are essentially existential variables (or logic variables) that must be instantiated to concrete labels satisfying all the entailment constraints in the proof tree, for a symbolic derivation to be sound. The idea of using free variables can be found in the literature, e.g., [1], in which a benefit is that "the use of free variables generates a smaller search space". We shall see in the following sections that our free variable system, although different from existing ones in certain aspects, can also narrow down the search space when zero-premise rules can give exact equality constraints so that the constraints generated by logical rules can be solved based on those generated by zero-premise rules. This means that the applications of structural rules (hidden in the logical rules) are not only driven by logical rules, but also by zero-premise rules.

Free variables are denoted by $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. We use $\mathbf{u}, \mathbf{v}, \mathbf{w}$ to denote either labels or free variables, and a, b, c are ordinary labels. A *symbolic sequent* is just a sequent but possibly with occurrences of free variables in place of labels. We shall sometimes refer to the normal (non-symbolic) sequent as a *ground sequent* to emphasize the fact that it contains no free variables. The symbolic proof system $FVLS_{BBI}$ is given in Figure 8. The rules are mostly similar to LS^{sf}_{BBI}, but lacking the entailment relations $\vdash_R$. Instead, new free variables are introduced when applying $*R$ and $-\!\!* \; L$ backward. Notice also that in $FVLS_{BBI}$, the $*R$ and $-\!\!* \; L$ rules do not compute the set $\mathcal{S}(\mathcal{G}, \sigma)$. So the relational atoms in $FVLS_{BBI}$ are those that are created by $*L$, $-\!\!*\ R$, $\top^* L$. In the following, given a derivation in $FVLS_{BBI}$, we shall assume that the free variables that are created in different branches of the derivation are pairwise distinct. We shall sometimes refer to a derivation in $FVLS_{BBI}$ simply as a *symbolic derivation.*

Initial Sequent:

$$\frac{}{\mathcal{G}||\Gamma; \mathbf{w}_1 : P \vdash \mathbf{w}_2 : P; \Delta}\ {\scriptstyle id}$$

Logical Rules:

$$\frac{}{\mathcal{G}||\Gamma; \mathbf{w} : \bot \vdash \Delta}\ {\scriptstyle \bot L} \qquad \frac{}{\mathcal{G}||\Gamma \vdash \mathbf{w} : \top; \Delta}\ {\scriptstyle \top R}$$

$$\frac{\mathcal{G}; (\epsilon, \mathbf{w} \triangleright \epsilon)||\Gamma \vdash \Delta}{\mathcal{G}||\Gamma; \mathbf{w} : \top^* \vdash \Delta}\ {\scriptstyle \top^* L} \qquad \frac{}{\mathcal{G}||\Gamma \vdash \mathbf{w} : \top^*; \Delta}\ {\scriptstyle \top^* R}$$

$$\frac{\mathcal{G}||\Gamma; \mathbf{w} : A; \mathbf{w} : B \vdash \Delta}{\mathcal{G}||\Gamma; \mathbf{w} : A \land B \vdash \Delta}\ {\scriptstyle \land L} \qquad \frac{\mathcal{G}||\Gamma \vdash \mathbf{w} : A; \Delta \qquad \mathcal{G}||\Gamma \vdash \mathbf{w} : B; \Delta}{\mathcal{G}||\Gamma \vdash \mathbf{w} : A \land B; \Delta}\ {\scriptstyle \land R}$$

$$\frac{\mathcal{G}||\Gamma \vdash \mathbf{w} : A; \Delta \qquad \mathcal{G}||\Gamma; \mathbf{w} : B \vdash \Delta}{\mathcal{G}||\Gamma; \mathbf{w} : A \to B \vdash \Delta}\ {\scriptstyle \to L} \qquad \frac{\mathcal{G}||\Gamma; \mathbf{w} : A \vdash \mathbf{w} : B; \Delta}{\mathcal{G}||\Gamma \vdash \mathbf{w} : A \to B; \Delta}\ {\scriptstyle \to R}$$

$$\frac{\mathcal{G}; (a, b \triangleright \mathbf{w})||\Gamma; a : A; b : B \vdash \Delta}{\mathcal{G}||\Gamma; \mathbf{w} : A * B \vdash \Delta}\ {\scriptstyle *L^{\dagger}} \qquad \frac{\mathcal{G}; (a, \mathbf{w} \triangleright c)||\Gamma; a : A \vdash c : B; \Delta}{\mathcal{G}||\Gamma \vdash \mathbf{w} : A -\!\!* B; \Delta}\ {\scriptstyle -\!\!*\ R^{\ddagger}}$$

$$\frac{\mathcal{G}||\Gamma \vdash \mathbf{x} : A; \mathbf{w} : A * B; \Delta \qquad \mathcal{G}||\Gamma \vdash \mathbf{y} : B; \mathbf{w} : A * B; \Delta}{\mathcal{G}||\Gamma \vdash \mathbf{w} : A * B; \Delta}\ {\scriptstyle *R^{\sharp}}$$

$$\frac{\mathcal{G}||\Gamma; \mathbf{w} : A -\!\!* B \vdash \mathbf{x} : A; \Delta \qquad \mathcal{G}||\Gamma; \mathbf{w} : A -\!\!* B; \mathbf{z} : B \vdash \Delta}{\mathcal{G}||\Gamma; \mathbf{w} : A -\!\!* B \vdash \Delta}\ {\scriptstyle -\!\!*\ L^{\natural}}$$

$\dagger$: a and b must be fresh in $*L$ $\qquad$ $\ddagger$: a and c must be fresh in $-\!\!*\ R$

$\sharp$: $\mathbf{x}$ and $\mathbf{y}$ are new free variables in $*R$ $\qquad$ $\natural$: $\mathbf{x}$ and $\mathbf{z}$ are new free variables in $-\!\!*\ L$

Figure 8: Labelled Sequent Calculus $FVLS_{BBI}$ for Boolean BI.

An *equality constraint* is an expression of the form $\mathcal{G} \vdash^?_R (\mathbf{u} = \mathbf{v})$, and a *relational constraint* is an expression of the form $\mathcal{G} \vdash^?_R (\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$. In both cases, we refer to $\mathcal{G}$ as the left hand side of the constraints, and $(\mathbf{u} = \mathbf{v})$ and $(\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$ as the right hand side. Constraints are ranged over by $\mathfrak{c}, \mathfrak{c}', \mathfrak{c}_1, \mathfrak{c}_2$, etc. Given a constraint $\mathfrak{c}$, we write $\mathcal{G}(\mathfrak{c})$ for the left hand side of $\mathfrak{c}$. A *constraint system* is just a multiset of constraints. We write $\mathcal{G} \vdash^?_R C$ for either an equality or relational constraint. We write $fv(\mathfrak{c})$ to denote the set of free variables in $\mathfrak{c}$, and $fv(\mathcal{C})$ to denote the set of free variables in a set of constraints $\mathcal{C}$.

Definition 5.1 (Constraint systems)**.** A *constraint system* is a pair $(\mathcal{C}, \preceq)$ of a finite multiset of constraints and a partial order on elements of $\mathcal{C}$ satisfying:

Monotonicity : $\mathfrak{c}_1 \preceq \mathfrak{c}_2$ implies $\mathcal{G}(\mathfrak{c}_1) \subseteq \mathcal{G}(\mathfrak{c}_2)$.

A constraint system is *well-formed* if it also satisfies

Unique variable origin : $\forall \mathbf{x}$ in $\mathcal{C}$, there exists a unique constraint occurrence $\mathfrak{c}(\mathbf{x}) = \mathcal{G}_x \vdash^?_R (\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$ s.t. $\mathbf{x}$ occurs in $(\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$, but not in $\mathcal{G}_x$, and $\mathbf{x}$ does not occur in any $\mathfrak{c}'$ where

$\mathfrak{c}' \neq \mathfrak{c}(\mathbf{x})$ and $\mathfrak{c}' \preceq \mathfrak{c}(\mathbf{x})$. Such a $\mathfrak{c}(\mathbf{x})$ is the origin of $\mathbf{x}$. Furthermore, for any free variable $\mathbf{x}$ and any constraint $\mathfrak{c}'$, if $\mathbf{x}$ occurs in $\mathcal{G}(\mathfrak{c}')$, then $\mathfrak{c}(\mathbf{x}) \preceq \mathfrak{c}'$.

The definition of constraint systems in Definition 5.1 is inspired by a similar definition used in security protocol analysis, see e.g., [8, 23], which involves proof search in certain proof systems representing intruder deduction capabilities. There the constraints are linearly ordered whereas in our case they are partially ordered.

Notice that $\mathcal{C}$ is a multiset, rather than a set. We could have defined $\mathcal{C}$ as a set, but the definition of composition of constraint systems (Definition 5.8) would be more complicated. Thus in $\mathcal{C}$ there can be more than one occurrence of the same constraint $\mathfrak{c}$.[3] We write $\mathcal{C}_1 \uplus \mathcal{C}_2$ to denote the multiset union of $\mathcal{C}_1$ and $\mathcal{C}_2$. We shall use the same symbol for a constraint and its occurrences. We shall often refer to a constraint occurrence as simply a constraint when it is clear from the context of discussions that we are referring to an occurrence rather than a constraint.

In a well-formed constraint system $(\mathcal{C}, \preceq)$, in every minimum constraint $\mathfrak{c}$, with respect to $\preceq$, $\mathcal{G}(\mathfrak{c})$ must be ground (see Lemma 5.2). The existence of such a $\mathfrak{c}$ is important in the definition of solutions for a well-formed constraint system, and in the proof that the symbolic proof system is sound with respect to its concrete counterpart (i.e., the derivation for the same formula in LS^{sf}_{BBI}).

From now on, we shall denote with $\mathfrak{c}(\mathbf{x})$ the constraint occurrence from where $\mathbf{x}$ originates, as defined in the above definition. We use the letter $\mathbb{C}$ to range over constraint systems.

We write $\mathfrak{c}_i \prec \mathfrak{c}_j$ when $\mathfrak{c}_i \preceq \mathfrak{c}_j$ and $\mathfrak{c}_i \neq \mathfrak{c}_j$. Further, we define a direct successor relation $\lessdot$ as follows: $\mathfrak{c}_i \lessdot \mathfrak{c}_j$ iff $\mathfrak{c}_i \prec \mathfrak{c}_j$ and there does not exist any $\mathfrak{c}_k$ such that $\mathfrak{c}_i \prec \mathfrak{c}_k \prec \mathfrak{c}_j$.

During proof search, associated constraints are generated as follows.

Definition 5.2. To a given symbolic derivation Π, we define a multiset of constraints $\mathcal{C}(\Pi)$ by structural induction on Π. We shall assume that variables introduced in instances of $*R$ and $-\!\!* \, L$ in Π are pairwise distinct. In the following, for each instance of the rules, we use the same naming schemes for labels and variables as in Figure 8. We distinguish several (base/inductive) cases based on the lowest rule of Π:

id	$\mathcal{C}(\Pi) = \{\mathcal{G} \vdash^?_R (\mathbf{w}_1 = \mathbf{w}_2)\}$
$\top^* R$	$\mathcal{C}(\Pi) = \{\mathcal{G} \vdash^?_R (\mathbf{w} = \epsilon)\}$
$*R$	$\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \uplus \mathcal{C}(\Pi_2) \uplus \{\mathcal{G} \vdash^?_R (\mathbf{x}, \mathbf{y} \triangleright \mathbf{w})\}$ where the left premise derivation is Π_1 and the right-premise derivation is Π_2
$-\!\!* \, L$	$\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \uplus \mathcal{C}(\Pi_2) \uplus \{\mathcal{G} \vdash^?_R (\mathbf{x}, \mathbf{w} \triangleright \mathbf{y})\}$ where the left premise derivation is Π_1 and the right-premise derivation is Π_2
–	If Π ends with any other rule, with premise derivations $\{\Pi_1, \dots, \Pi_n\}$, then $\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \uplus \cdots \uplus \mathcal{C}(\Pi_n)$.

Proof search in the free variable system is a refutation procedure. The ternary relational atoms generated by $\top^* L$, $*L$, and $-\!\!* \, R$ are the base knowledge about the monoidal semantics; the constraints generated by $*R$ and $-\!\!* \, L$ give hints on what are further needed to falsify the end sequent, these constraints need to be solved by grounding the free variables to labels. Zero-premise rules refute that the ternary relational atoms generated along a branch cannot form a counter-model. The constraint generated by the rule id or $\top^* R$ can close a branch whenever the constraint is satisfied.

Each constraint $\mathfrak{c} \in \mathcal{C}(\Pi)$ corresponds to a rule instance $r(\mathfrak{c})$ in Π where $\mathfrak{c}$ is generated. The ordering of the rules in the derivation tree of Π then naturally induces a partial order on $\mathcal{C}(\Pi)$. That

[3]Another way of looking at this is to consider a set of pairs of the form $(i, \mathfrak{c})$ where i is an identifier (e.g., a natural number) and $\mathfrak{c}$ is a constraint. Multisets offer a more convenient abstraction.

is, let $\preceq^{\Pi}$ be an ordering on $\mathcal{C}(\Pi)$ defined as follows: $\mathfrak{c}_1 \preceq^{\Pi} \mathfrak{c}_2$ iff the conclusion of $r(\mathfrak{c}_1)$ appears in the path from the root sequent to the conclusion of $r(\mathfrak{c}_2)$. Then obviously $\preceq^{\Pi}$ is a partial order.

Lemma 5.1. *Let Π be a symbolic derivation. Then $(\mathcal{C}(\Pi), \preceq^{\Pi})$ is a constraint system. Moreover, if the root sequent is ground, then $(\mathcal{C}(\Pi), \preceq^{\Pi})$ is well-formed.*

Proof. Each constraint in $\mathcal{C}(\Pi)$ is associated with an instance of a rule in Π; this induces a partial order on the constraints as follows: $c_1 \preceq c_2$ iff the rule instance where c_1 originates from appears in the path from the root to the rule instance where c_2 originates from, in the derivation tree of Π. It is easy to see that this gives us a partial order. The unique variable origin property of $\mathcal{C}(\Pi)$ is also satisfied by the fact that new variables can only be created at $*R$ and $-\!\!* \ L$. So the minimum constraint for each variable is the constraint generated by the rule instance where these variables are created. $\square$

Given a symbolic derivation Π, we define $\mathbb{C}(\Pi)$ as the constraint system $(\mathcal{C}(\Pi), \preceq^{\Pi})$ as defined above. A consequence of Lemma 5.1 is that if $\mathcal{C}(\Pi) \neq \{\ \}$, then there exists a minimum constraint $\mathfrak{c}$, w.r.t. the partial order $\preceq^{\Pi}$, such that $\mathcal{G}(\mathfrak{c})$ is ground.

We now define what it means for a constraint system to be solvable. The complication arises when we need to capture that (ternary) relational atoms created by the solution need to be accumulated along each branch in the proof search in order to guarantee the soundness of $FVLS_{BBI}$. A *free-variable substitution* θ is a total mapping from free variables to free-variables or labels, which is an identity map on all variables except for a finite number of them. The *domain* of a substitution θ, denoted by $dom(\theta)$, is defined as $dom(\theta) = \{\mathbf{x} \mid \theta(\mathbf{x}) \neq \mathbf{x}\}$. Given θ, θ' with $dom(\theta') \subseteq dom(\theta)$, and a set V of free variables, $\theta \uparrow V$ is the substitution obtained from θ by intersecting its domain with V; and $\theta \setminus \theta'$ is the result of θ "subtract" θ'; formally:

$$\mathbf{x}(\theta \uparrow V) = \begin{cases} \mathbf{x}\theta & \text{if } \mathbf{x} \in V \\ \mathbf{x} & \text{otherwise.} \end{cases} \qquad \mathbf{x}(\theta \setminus \theta') = \begin{cases} \mathbf{x}\theta & \text{if } \mathbf{x} \notin dom(\theta') \\ \mathbf{x} & \text{otherwise.} \end{cases}$$

Definition 5.3 (Simple constraints and their solutions)**.** A constraint $\mathfrak{c}$ is *simple* if its left hand side $\mathcal{G}(\mathfrak{c})$ contains no free variables. A *solution* (θ, σ) to a simple constraint $\mathfrak{c}$ is a substitution θ and a sequence σ of abstract instances of structural rules such that:

- If $\mathfrak{c}$ is $\mathcal{G} \vdash_R^? (\mathbf{u} = \mathbf{v})$ then σ is a derivation of $\mathcal{G} \vdash_R (\mathbf{u}\theta = \mathbf{v}\theta)$.
- If $\mathfrak{c}$ is $\mathcal{G} \vdash_R^? (\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$ then σ is a derivation of $\mathcal{G} \vdash_R (\mathbf{u}\theta, \mathbf{v}\theta \triangleright \mathbf{w}\theta)$.

Lemma 5.2. *If $(\mathcal{C}, \preceq)$ is well-formed, the minimum constraints in $\mathcal{C}$ are simple constraints.*

Proof. Suppose otherwise, i.e., there exists a minimum $\mathfrak{c} \in \mathcal{C}$ such that $\mathcal{G}(\mathfrak{c})$ contains a free variable $\mathbf{x}$. By the unique variable origin property (Definition 5.1), there exists $\mathfrak{c}(\mathbf{x})$ such that $\mathbf{x}$ is not free in $\mathcal{G}(\mathfrak{c}(\mathbf{x}))$ and $\mathfrak{c}(\mathbf{x}) \preceq \mathfrak{c}$. Also by Definition 5.1, $\mathbf{x}$ cannot occur in $\mathfrak{c}(\mathbf{x})$, so $\mathfrak{c}(\mathbf{x})$ is not $\mathfrak{c}$, and we have $\mathfrak{c}(\mathbf{x}) \prec \mathfrak{c}$, which contradicts the minimality of $\mathfrak{c}$. Therefore $\mathcal{G}(\mathfrak{c})$ must be ground. $\square$

From the above definition, a simple constraint $\mathcal{G} \vdash_R^? (\mathbf{u} = \mathbf{v})$ is solvable if there is a series of structural rule applications on $\mathcal{G}$ and a series of free variable substitutions θ, such that $\mathbf{u}\theta = \mathbf{v}\theta$. Since $\mathcal{G}$ only contains labels, the structural rule applications in this case are often not needed. For example, the simple constraint

$$(w_1, w_2 \triangleright w_3) \vdash_R^? (w_1 = \mathbf{x})$$

is solvable by applying no structural rules and only assigning $\mathbf{x}$ to w_1 in the free variable substitution θ. The case for a ternary relation on the r.h.s. is defined similarly, but this case often involves structural rule applications. For example, the simple constraint

$$(w_1, w_2 \triangleright w_0); (w_3, w_4 \triangleright w_1) \vdash^?_R (w_4, w_2 \triangleright \mathbf{x})$$

can be solved by letting σ be $[A(\cdots); E(\cdots)]$ as in the following structural rule applications:

$$\dfrac{\dfrac{(w_4, w_2 \triangleright w_5); \cdots}{(w_3, w_5 \triangleright w_0); (w_2, w_4 \triangleright w_5); \cdots}\,E}{(w_1, w_2 \triangleright w_0); (w_3, w_4 \triangleright w_1); \cdots}\,A$$

And the free variable substitution θ would be $[w_5/\mathbf{x}]$.

Definition 5.4 (Restricting a constraint system)**.** Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system, and $\mathfrak{c}$ be a minimum (simple) constraint occurrence in $\mathbb{C}$. Let (θ, σ) be a solution to $\mathfrak{c}$ and $\mathcal{G}' = \mathcal{S}(\mathcal{G}(\mathfrak{c}), \sigma)$, and let $\mathcal{C}_\mathfrak{c}$ be $\mathcal{C}$ with the constraint occurrence $\mathfrak{c}$ removed. Define a function f on constraints:

$$f(\mathfrak{c}') = \begin{cases} (\mathcal{G}' \cup \mathcal{G}\theta \vdash^?_R C\theta) & \text{if } \mathfrak{c}' = (\mathcal{G} \vdash^?_R C) \in \mathcal{C}_\mathfrak{c} \textit{ and } \mathfrak{c} \preceq \mathfrak{c}', \\ \mathfrak{c}' & \text{otherwise.} \end{cases}$$

The *restriction of* $\mathbb{C}$ by $(\mathfrak{c}, \theta, \sigma)$, written $\mathbb{C} \uparrow (\mathfrak{c}, \theta, \sigma)$, is the pair $(\mathcal{C}', \preceq')$, where (1) $\mathcal{C}' = \{f(\mathfrak{c}') \mid \mathfrak{c}' \in \mathcal{C}_\mathfrak{c}\}$ and (2) $f(\mathfrak{c}_1) \preceq' f(\mathfrak{c}_2)$ iff $\mathfrak{c}_1 \preceq \mathfrak{c}_2$.

The proof of the following lemma is straightforward from Definition 5.4.

Lemma 5.3. *The pair $\mathbb{C} \uparrow (\mathfrak{c}, \theta, \sigma)$ as defined in Definition 5.4 is a well-formed constraint system.*

Definition 5.5 (Solution to a well-formed constraint system)**.** Let $\mathbb{C} = (\{\mathfrak{c}_1, \ldots, \mathfrak{c}_n\}, \preceq)$ be a well-formed constraint system. A *solution* $(\theta, \{\sigma_1, \ldots, \sigma_n\})$ to $\mathbb{C}$ is a substitution and a set of sequences of structural rules, such that:

If $n = 0$ then $(\theta, \{\sigma_1, \ldots, \sigma_n\})$ is trivially a solution.

If $n \geq 1$ then there must exist some minimum (simple) constraint in $\mathbb{C}$. For any minimum constraint $\mathfrak{c}_i$, let $\theta_i = \theta \uparrow fv(\mathfrak{c}_i)$, then (θ_i, σ_i) is a solution to $\mathfrak{c}_i$, and $(\theta \setminus \theta_i, \{\sigma_1, \ldots, \sigma_n\} \setminus \sigma_i)$ is a solution to $\mathbb{C} \uparrow (\mathfrak{c}_i, \theta_i, \sigma_i)$.

If a constraint system $\mathbb{C} = (\{\mathfrak{c}_1, \cdots, \mathfrak{c}_n\}, \preceq)$ has a solution $(\theta, \{\sigma_1, \cdots, \sigma_n\})$, then there is a solution to each $\mathfrak{c}_i$, which is computed recursively from the minimum constraints in $\mathbb{C}$, as defined in Definition 5.5. It is easy to see that every $\mathfrak{c} \in \{\mathfrak{c}_1, \cdots, \mathfrak{c}_n\}$ has a solution of the form $(\theta_\mathfrak{c}, \sigma)$, for some $\theta_\mathfrak{c}$, where $\sigma \in \{\sigma_1, \ldots, \sigma_n\}$. Moreover, σ is uniquely related to $\mathfrak{c}$. In the following, given such a constraint $\mathfrak{c}$, we shall write $dev(\mathfrak{c})$ to refer to that sequence of rules σ in its solution.

Example 5.1. We now give an example of how to prove a formula using the free variable system. Suppose we want to prove $((P * Q) * R) \to (P * (Q * R))$, where P, Q and R are propositional variables. Using $FVLS_{BBI}$, we build a symbolic derivation as in Figure 9. Note that this derivation is generated automatically from our theorem prover which uses $a_0, a_1, \cdots$ for label variables, hence the end-sequent is of the form $a_0 : F$ where a_0 stands for the label variable w and some of the indices of variables may not be contiguous. We subscript $\vdash$ with a number i to indicate that the rule applied on this sequent generates the constraint $\mathfrak{c}_i$. The set of constraints $\mathcal{C} = \{\mathfrak{c}_1, \cdots, \mathfrak{c}_5\}$ are

Let $\mathcal{G} = \{(a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1)\}$ and $\Gamma_1 := \{a_2 : R\ ;\ a_3 : P\}$ and $\Gamma_2 := \{a_3 : P\ ;\ a_4 : Q\}$ in

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{G}||a_2 : R; a_3 : P; a_4 : Q \vdash_5 \mathbf{x}_5 : P}\,id \quad \dfrac{\dfrac{}{\mathcal{G}||\Gamma_1; a_4 : Q \vdash_4 \mathbf{x}_7 : Q}\,id \quad \dfrac{}{\mathcal{G}||a_2 : R; \Gamma_2 \vdash_3 \mathbf{x}_8 : R}\,id}{\mathcal{G}||a_2 : R; a_3 : P; a_4 : Q \vdash_2 \mathbf{x}_6 : Q * R}\,*R}{\mathcal{G}||a_2 : R; a_3 : P; a_4 : Q \vdash_1 a_0 : P * (Q * R)}\,*R}{(a_1, a_2 \triangleright a_0)||a_1 : P * Q; a_2 : R \vdash a_0 : P * (Q * R)}\,*L}{a_0 : (P * Q) * R \vdash a_0 : P * (Q * R)}\,*L}{\vdash a_0 : ((P * Q) * R) \to (P * (Q * R))}\,\to R$$

Figure 9: A symbolic derivation for $((P * Q) * R) \to (P * (Q * R))$.

generated from this derivation as below:

$$\begin{array}{lll}
\mathfrak{c}_5 : & (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_3 = \mathbf{x}_5) \\
\mathfrak{c}_4 : & (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_4 = \mathbf{x}_7) \\
\mathfrak{c}_3 : & (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_2 = \mathbf{x}_8) \\
\mathfrak{c}_2 : & (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (\mathbf{x}_7, \mathbf{x}_8 \triangleright \mathbf{x}_6) \\
\mathfrak{c}_1 : & (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (\mathbf{x}_5, \mathbf{x}_6 \triangleright a_0).
\end{array}$$

The partial order $\preceq$ on these constraints is based on the order of rule applications in the symbolic derivation in Figure 9. Thus we obtain that $\mathfrak{c}_1 \preceq \mathfrak{c}_2$, $\mathfrak{c}_1 \preceq \mathfrak{c}_5$, $\mathfrak{c}_2 \preceq \mathfrak{c}_3$, and $\mathfrak{c}_2 \preceq \mathfrak{c}_4$. By Lemma 5.1, the constraint system $(\mathcal{C}, \preceq)$ is well-formed.

The naive way to solve a constraint system would start by solving the minimum constraint (i.e., $\mathfrak{c}_1$), then solve the other constraints in the partial order $\preceq$. Finally, if the constraints generated by zero-premise rules can be solved based on the existing free variable substitutions, then we obtain a solution to the constraint system; otherwise we have to backtrack and try to solve previous constraints using different structural rule applications and/or free variable substitutions.

We have not introduced a method to solve the constraints yet, the next section will give a heuristic method. For now let us just non-deterministically guess the solutions, starting from the minimum constraint $\mathfrak{c}_1$. If we were to solve $\mathfrak{c}_1$ with the solution $(\{\mathbf{x}_5 \mapsto a_0, \mathbf{x}_6 \mapsto a_1 1\}, \{\})$, then we would have trouble when solving the constraint $\mathfrak{c}_5$, because $(a_3 = a_0)$ cannot be derived by any structural rule applications. Backtracking to $\mathfrak{c}_1$, suppose the oracle says we should apply structural rules on $\mathcal{G}(\mathfrak{c}_1)$ as below:

$$\dfrac{(a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) \vdash_R (\mathbf{x}_5, \mathbf{x}_6 \triangleright a_0)}{(a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) \vdash_R (\mathbf{x}_5, \mathbf{x}_6 \triangleright a_0)}\,{\scriptstyle A}$$

where w is a fresh label created by the A application. Then $\mathfrak{c}_1$ can be solved by the solution $(\{\mathbf{x}_5 \mapsto a_3, \mathbf{x}_6 \mapsto w\}, \{A(\{(a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1)\}, [], \{w\}, \{(a_3, w \triangleright a_0); (a_2, a_4 \triangleright w)\})\})$. By Def. 5.4, this solution restricts the constraints system as follows:

$$\begin{array}{lll}
\mathfrak{c}_5 : & (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_3 = a_3) \\
\mathfrak{c}_4 : & (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_4 = \mathbf{x}_7) \\
\mathfrak{c}_3 : & (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_2 = \mathbf{x}_8) \\
\mathfrak{c}_2 : & (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (\mathbf{x}_7, \mathbf{x}_8 \triangleright w).
\end{array}$$

The next step is to use the rule E to create $(a_4, a_2 \triangleright w)$ from $(a_2, a_4 \triangleright w)$. The constraint $\mathfrak{c}_2$ can then be solved by the solution $(\{\mathbf{x}_7 \mapsto a_4, \mathbf{x}_8 \mapsto a_2\}, \{E(\{(a_2, a_4 \triangleright w)\}, [], \{\}, \{(a_4, a_2 \triangleright w)\})\})$. The constraint system is now:

$$\begin{array}{lll}
\mathfrak{c}_5 : & (a_4, a_2 \triangleright w); (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_3 = a_3) \\
\mathfrak{c}_4 : & (a_4, a_2 \triangleright w); (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_4 = a_4) \\
\mathfrak{c}_3 : & (a_4, a_2 \triangleright w); (a_3, w \triangleright a_0); (a_2, a_4 \triangleright w); (a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1) & \vdash_R \quad (a_2 = a_2).
\end{array}$$

The remaining constraints can be trivially solved by the solution $(\{\}, \{\})$. Since the constraint system is solvable, the symbolic derivation in Figure 9 corresponds to a valid proof of the formula by the coming soundness and completeness theorems. The reader can check that the above solutions satisfy the conditions in Def. 5.5.

Theorem 5.4 (Soundness)**.** *Let Π be a symbolic derivation of a ground sequent $\mathcal{G}||\Gamma \vdash \Delta$. If $\mathcal{C}(\Pi)$ is solvable, then $\mathcal{G}||\Gamma \vdash \Delta$ is derivable in LS^{sf}_{BBI}.*

Proof. By induction on the height n of derivation Π. The basic idea of the proof is that one progressively "grounds" a symbolic derivation, starting from the root of the derivation. At each inductive step we show that grounding the premises corresponds to restricting the constraint system induced by the symbolic derivation.
Base case: $n = 1$. In this case, we can only use a zero-premise rule to prove the sequent. Since the sequent is ground, there are no free variables. Thus the constraint generated by the rule application is a simple constraint, of the form $\mathcal{G} \vdash^?_R (a = b)$ or $\mathcal{G} \vdash^?_R (a = \epsilon)$. A solution of this constraint is simply a derivation σ of $\mathcal{G} \vdash_R (a = b)$ (resp. $\mathcal{G} \vdash (a = \epsilon)$. In either case, this translates straightforwardly into a derivation in LS^{sf}_{BBI} with the same rule.
Inductive case: $n > 1$. This can be done by a case analysis of the last rule application in Π. We demonstrate the case for $*R$, where a constraint is generated. The case for $-\!\!* \, L$ is analogous, and the other cases are easy since we can use the induction hypothesis directly. Suppose Π runs as follows.

$$\dfrac{\begin{array}{c}\Pi_1\\ \mathcal{G}||\Gamma \vdash \mathbf{x} : A; w : A * B; \Delta\end{array} \qquad \begin{array}{c}\Pi_2\\ \mathcal{G}||\Gamma \vdash \mathbf{y} : B; w : A * B; \Delta\end{array}}{\mathcal{G}||\Gamma \vdash w : A * B; \Delta} {\scriptstyle *R}$$

Suppose $\mathbb{C}(\Pi) = (\{\mathfrak{c}_1, \ldots, \mathfrak{c}_k\}, \preceq)$, for some $k \geq 1$. Suppose that the constraint generated by this rule application is $\mathcal{G} \vdash^?_R (\mathbf{x}, \mathbf{y} \triangleright w)$ and it corresponds to $\mathfrak{c}_i$ for some $i \in \{1, \ldots, k\}$. By the assumption, there is a solution $(\theta, \{\sigma_1, \cdots, \sigma_k\})$ for the constraint system $\mathbb{C} = (\mathcal{C}(\Pi), \preceq^{\Pi})$. Now $\mathfrak{c}_i$ must be a simple constraint in $\mathbb{C}$, as the end sequent is ground. Let (θ_i, σ_i) be the solution to $\mathfrak{c}_i$, where θ_i is a restriction to θ containing $\mathbf{x}$ and $\mathbf{y}$, and $\sigma_i \in \{\sigma_1, \cdots, \sigma_k\}$. By definition of the solution to a simple constraint, σ_i is a derivation of $\mathcal{G} \vdash_R (\mathbf{x}\theta_i, \mathbf{y}\theta_i \triangleright w)$. Therefore in LS^{sf}_{BBI}, to derive the end sequent, we apply $*R$ backwards:

$$\dfrac{\mathcal{S}(\mathcal{G}, \sigma_i)||\Gamma \vdash \mathbf{x}\theta_i : A; w : A * B; \Delta \qquad \mathcal{S}(\mathcal{G}, \sigma_i)||\Gamma \vdash \mathbf{y}\theta_i : B; w : A * B; \Delta}{\mathcal{G}||\Gamma \vdash w : A * B; \Delta} {\scriptstyle *R}$$

The condition on this rule is $\mathcal{G} \vdash_R (\mathbf{x}\theta_i, \mathbf{y}\theta_i \triangleright w)$. Now we construct the derivation for both branches in the following way. Firstly we substitute $\mathbf{x}$ and $\mathbf{y}$ with $\mathbf{x}\theta_i$ and $\mathbf{y}\theta_i$ respectively in Π_1 and Π_2, making the end sequents in the two derivations ground. Let us refer to the modified derivations as Π'_1 and Π'_2 respectively. Then we add $\mathcal{S}(\mathcal{G}, \sigma_i)$ to the left hand side of each sequent in Π'_1 and Π'_2 and each constraint in $\mathcal{C}(\Pi'_1) \uplus \mathcal{C}(\Pi'_2)$. Let the resultant derivations be Π''_1 and Π''_2 respectively. Now the end sequents of Π''_1 and Π''_2 are respectively just the same as the two branches we created in the LS^{sf}_{BBI} derivation. Moreover, each constraint in $\mathcal{C}(\Pi''_1) \uplus \mathcal{C}(\Pi''_2)$ is in the restricted constraint system $\mathbb{C}' = (\mathcal{C}', \preceq') = (\mathcal{C}(\Pi), \preceq^{\Pi}) \uparrow (\mathfrak{c}_i, \theta_i, \sigma_i)$, which has a solution $(\theta \setminus \theta_i, \{\sigma_1, \cdots, \sigma_k\} \setminus \sigma_i)$, and obeys the partial order $\preceq'$. Further, as Π''_1 (resp. Π''_2) uses the same rule applications as in Π_1 (resp. Π_2), the order of constraints is preserved. That is, in the constraints system $\mathbb{C}_1 = (\mathcal{C}(\Pi''_1), \preceq^{\Pi''_1})$ (resp.

$\mathbb{C}_2 = (\mathcal{C}(\Pi_2''), \preceq^{\Pi_2''})$), if $\mathfrak{c} \preceq^{\Pi_1''} \mathfrak{c}'$ (resp. $\mathfrak{c} \preceq^{\Pi_2''} \mathfrak{c}'$) then $\mathfrak{c} \preceq' \mathfrak{c}'$ in $\mathbb{C}'$. Therefore we can construct the solution (θ_1'', Σ_1) to $\mathbb{C}_1$ (and analogously to $\mathbb{C}_2$) as follows.

$$\theta_1'' = (\theta \setminus \theta_i) \uparrow fv(\mathcal{C}(\Pi_1''))$$
$$\Sigma_1 = \{\sigma \mid \mathfrak{c} \in \mathcal{C}(\Pi_1''), \sigma \in \{\sigma_1, \cdots, \sigma_k\} \setminus \sigma_i, \ \textit{and} \ \sigma = dev(\mathfrak{c})\}.$$

By the induction hypothesis, we can obtain a LS_{BBI}^{sf} derivation for each branch. □

To prove the completeness of $FVLS_{BBI}$, we show that for every cut-free derivation Π of a (ground) sequent in LS_{BBI}^{sf}, there is a symbolic derivation Π' of the same sequent such that $\mathcal{C}(\Pi')$ is solvable. It is quite obvious that Π' should have exactly the same rule applications as Π; the only difference is that some relational atoms are omitted in the derivation, but instead are accumulated in the constraint system. Additionally, some (new) labels are replaced with free variables. This is formalised in the following definition.

Definition 5.6. Given a sequent in a LS_{BBI}^{sf} derivation, let $\mathcal{G}$ be the set of its relational atoms, we define $\mathcal{G}_E$ as the subset of $\mathcal{G}$ that contains those ternary relational atoms created by $*L$, $-\!\!* R$, and $\top^* L$. We define $\mathcal{G}_S = \mathcal{G} \setminus \mathcal{G}_E$. We refer to $\mathcal{G}_E$ as the **essential** subset of $\mathcal{G}$, and $\mathcal{G}_S$ as the **supplementary** subset of $\mathcal{G}$.

For a list L, we denote by $head(L)$ the first element in the list L and $tail(L)$ the list of L without the first element, and $end(L)$ the last element in L. We denote by $L_1@L_2$ the concatenation of two lists L_1 and L_2, and $pre(x)$ the predecessor of x in a list L, and $suc(x)$ the successor of x in L.

Given a well-formed constraint system $(\mathcal{C}, \preceq)$, we can define a partial order $\preceq^v$ on free variables of $\mathcal{C}$ as follows: $\mathbf{x} \preceq^v \mathbf{y}$ iff $\mathfrak{c}(\mathbf{x}) \preceq \mathfrak{c}(\mathbf{y})$. That is, free variables are ordered according to their origins. The relations $\prec^v$ and $\lessdot^v$ are defined analogously to $\prec$ and $\lessdot$, i.e., as the non-reflexive subset of $\preceq^v$ and the successor relation.

Definition 5.7 (A thread of variables)**.** Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system, and let X be a list of free variables $\mathbf{x}_1, \dots, \mathbf{x}_n$, where $n \geq 0$. Let $\preceq^v$ be the partial order on variables, derived from $\preceq$. We say X is a *thread of free variables of* $\mathbb{C}$ (or simply a *thread* of $\mathbb{C}$) iff it satisfies the following conditions:

1. $\forall \mathbf{x} \in X$, $\mathbf{x} \in fv(\mathcal{C})$
2. For every $i \in \{1, \dots, n-1\}$, $\mathbf{x}_i \lessdot^v \mathbf{x}_{i+1}$.
3. If $n \geq 1$, then $\mathbf{x}_1$ is a minimum element and $\mathbf{x}_n$ is a maximum element of $\preceq^v$.
4. If $n \geq 1$, then $\mathfrak{c}(\mathbf{x}_1)$ is a minimum constraint in $\mathcal{C}$.

A thread is effectively those variables that are generated along a certain branch in a $FVLS_{BBI}$ symbolic derivation. It is not hard to verify that in a valid symbolic derivation in $FVLS_{BBI}$ of a ground sequent, the set of free variables in any symbolic sequent in the derivation can be linearly ordered as a thread.

Definition 5.8. Let $\mathbb{C} = (\mathcal{C}_1, \preceq_1)$ be a well-formed constraint system, let X be a thread of $\mathbb{C}_1$ and $\mathbb{C}_2 = (\mathcal{C}_2, \preceq_2)$ be a constraint system (may not be well-formed) such that X consists of free variables in $fv(\mathcal{C}_1) \cap fv(\mathcal{C}_2)$. Also, assume that every variable $\mathbf{x}$ in $\mathcal{C}_2$, except for those in X, satisfies the unique variable origin property, i.e., $\mathbf{x}$ originates from a constraint in $\mathcal{C}_2$. The *composition* of $\mathbb{C}_1$ and $\mathbb{C}_2$ along the thread X, written $\mathbb{C}_1 \circ^X \mathbb{C}_2$, is the constraint system $(\mathcal{C}, \preceq)$ such that:

- $\mathcal{C} = \mathcal{C}_1 \uplus \mathcal{C}_2$; and

- Define a relation $\mathcal{R}$ as follows: for $\mathfrak{c}_1, \mathfrak{c}_2 \in \mathcal{C}$, $\mathfrak{c}_1 \mathcal{R} \mathfrak{c}_2$ iff either one of the following holds:
  - $\mathfrak{c}_1 \preceq_1 \mathfrak{c}_2$,
  - $\mathfrak{c}_1 \preceq_2 \mathfrak{c}_2$, or
  - X is non-empty, $\mathbf{y} = \mathit{end}(X)$, $\mathfrak{c}_1 = \mathfrak{c}(\mathbf{y})$ and $\mathfrak{c}_2 \in \mathcal{C}_2$.

  Then define $\preceq$ to be the transitive closure of $\mathcal{R}$.

This definition basically says that the composition of $\mathbb{C}_1$ and $\mathbb{C}_2$ along X is obtained by simply ordering the constraints so that all constraints $\mathcal{C}_2$ are greater than $\mathfrak{c}(\mathbf{y})$, where $\mathbf{y}$ is the last variable in X. If X is empty, then $\mathcal{C}_1$ and $\mathcal{C}_2$ are independent, and $\preceq$ is simply the union of $\preceq_1$ and $\preceq_2$.

The following two lemmas follow straightforwardly from the definitions.

Lemma 5.5. *Let $(\mathcal{C}, \preceq)$ be as defined in Definition 5.8. Then $(\mathcal{C}, \preceq)$ is well-formed.*

Lemma 5.6. *Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system and let X be a thread of $\mathbb{C}$. Let Π be a symbolic derivation such that the free variables in its end sequent are exactly those in X. Then $\mathbb{C} \circ^X \mathbb{C}(\Pi)$ is well-formed.*

Definition 5.9. Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system and let $S = (\theta, \{\vec{\sigma}\})$ be its solution. Let X be a thread of $\mathbb{C}$. Define a set of relational atoms $\mathcal{S}^*(\mathbb{C}, S, X)$ inductively by the length n of X as follows:

- If $n = 0$ then $\mathcal{S}^*(\mathbb{C}, S, []) = \emptyset$
- Suppose $n > 0$. Let $\mathit{head}(X) = \mathbf{x}$. Then $\mathfrak{c}(\mathbf{x}) \in \mathcal{C}$ is a minimum constraint of $\mathbb{C}$, and there exists $\sigma_{\mathbf{x}} \in \{\vec{\sigma}\}$ such that $(\theta_{\mathbf{x}}, \sigma_{\mathbf{x}})$ is a solution to $\mathfrak{c}(\mathbf{x})$, where $\theta_{\mathbf{x}} = \theta \uparrow fv(\mathfrak{c}(\mathbf{x}))$. In this case, $\mathcal{S}^*(\mathbb{C}, S, X)$ is defined as follows.

$$\mathcal{S}^*(\mathbb{C}, S, X) = \mathcal{S}(\mathcal{G}(\mathfrak{c}(\mathbf{x})), \sigma_{\mathbf{x}}) \cup \mathcal{S}^*(\mathbb{C} \uparrow (\mathfrak{c}(\mathbf{x}), \theta_{\mathbf{x}}, \sigma_{\mathbf{x}}), S', \mathit{tail}(X))$$

  where $S' = (\theta \setminus \theta_{\mathbf{x}}, \{\vec{\sigma}\} \setminus \{\sigma_{\mathbf{x}}\})$.

Notice that by the definition of restriction to a constraint system, every time a minimum constraint $\mathfrak{c}_{\mathbf{x}}$ is eliminated in the second clause in the above definition, $\mathcal{S}(\mathcal{G}(\mathfrak{c}_{\mathbf{x}}), \sigma_{\mathbf{x}})$ is also added to the left hand side of every successor constraints of $\mathfrak{c}_{\mathbf{x}}$ in $\mathcal{C}$. Therefore it is straightforward that the following proposition holds.

Proposition 5.7. *Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system. Let $\mathcal{G} = \mathcal{S}^*(\mathbb{C}, S, X)$, for some thread X of $\mathbb{C}$, let $\mathbf{x}_e = \mathit{end}(X)$ and let $S = (\theta, \{\vec{\sigma}\})$ be a solution to $\mathbb{C}$. Let $\mathfrak{c} = \mathcal{G}_{\mathfrak{c}} \vdash^?_R C_{\mathfrak{c}}$ be a constraint not in $\mathcal{C}$, such that $\mathcal{G}_{\mathfrak{c}}$ only contains free variables that occur in $\mathbb{C}$. Let $\mathbf{x}$ be a new variable occurring only on the right hand side of $\mathfrak{c}$. Let $\mathbb{C}' = (\mathcal{C}', \preceq')$ be the following constraint system:*

- *$\mathcal{C}' = \mathcal{C} \uplus \{\mathfrak{c}\}$;*
- *$\preceq'$ is the smallest extension of $\preceq$ such that $\mathfrak{c}(\mathbf{x}_e) \lessdot \mathfrak{c}$.*

Let (θ_x, σ_x) be the solution to $\mathfrak{c}' = \mathcal{G} \cup \mathcal{G}_{\mathfrak{c}}\theta \vdash^?_R C_{\mathfrak{c}}\theta$, $S' = (\theta \cup \theta_x, \{\vec{\sigma}, \sigma_x\})$, and $X' = X@[\mathbf{x}]$. Then $\mathcal{S}^*(\mathbb{C}', S', X') = \mathcal{S}(\mathcal{G} \cup \mathcal{G}_{\mathfrak{c}}\theta, \sigma_x)$.

Theorem 5.8. *Let Π be a derivation of a sequent in LS^{sf}_{BBI}. Then there exists a symbolic derivation Π' of the same sequent such that $\mathcal{C}(\Pi')$ is solvable.*

Proof. We describe the construction from a LS^{sf}_{BBI} derivation Π to a $FVLS_{BBI}$ derivation Π'. We need to prove a stronger invariant: for each sequent $\mathcal{G}_E; \mathcal{G}_S || \Gamma \vdash \Delta$ in Π, if there exists a triple consisting of:

(1) a symbolic sequent $\mathcal{G}'_E || \Gamma' \vdash \Delta'$,
(2) a well-formed constraint system $\mathbb{C} = (\mathcal{C}, \preceq)$,
(3) and a solution $S = (\theta, \{\overrightarrow{\sigma}\})$ to $\mathbb{C}$

such that

(I) there exists a thread X of $\mathbb{C}$ consisting of $fv(\mathcal{G}'_E || \Gamma' \vdash \Delta')$,
(II) $\mathcal{G}'_E\theta = \mathcal{G}_E$, $\Gamma'\theta = \Gamma$, $\Delta'\theta = \Delta$ and
(III) $\mathcal{G}_E \cup \mathcal{G}_S = \mathcal{S}^*(\mathbb{C}, S, X)$,

then there is a symbolic derivation Ψ of $\mathcal{G}'_E || \Gamma' \vdash \Delta'$ such that $\mathbb{C} \circ^X \mathbb{C}(\Psi)$ is well-formed and solvable.

First of all, by Lemma 5.6, since the end sequent in Ψ only contains the free variables occurring in X, the composition $\mathbb{C} \circ^X \mathbb{C}(\Psi)$ must be well-formed. Thus we only need to show that there is a solution to this constraint system. We prove this by case analysis on the last rule in Π, and show that in each case, for each premise of the rule, one can find a triple satisfying the above property, such that the symbolic sequent(s) in the premise(s), together with the one in the conclusion form a valid inference in $FVLS_{BBI}$. We illustrate it here with a case when Π ends with $*R$:

Suppose Π ends with $*R$, where the derivation runs as:

$$\dfrac{\begin{array}{cc} \Pi_1 & \Pi_2 \\ \mathcal{S}((\mathcal{G}_E;\mathcal{G}_S),\sigma)||\Gamma \vdash w_1 : A; w : A * B; \Delta & \mathcal{S}((\mathcal{G}_E;\mathcal{G}_S),\sigma)||\Gamma \vdash w_2 : B; w : A * B; \Delta \end{array}}{\mathcal{G}_E;\mathcal{G}_S||\Gamma \vdash w : A * B; \Delta}\ {\scriptstyle *R}$$

and the relational entailment is $\mathcal{G}_E; \mathcal{G}_S \vdash_R (w_1, w_2 \triangleright w)$. Suppose that the relation in the last item is derived via σ. Suppose further that we can find a triple consisting of (1) a symbolic sequent $\mathcal{G}'_E || \Gamma' \vdash \mathbf{w} : A * B; \Delta'$, (2) a well-formed constraint system $\mathbb{C} = (\mathcal{C}, \preceq)$, and (3) a solution $S = (\theta, \{\sigma_1, \ldots, \sigma_n\})$ to $\mathbb{C}$, satisfying the following: (I) X is a thread of $\mathbb{C}$ consisted of $fv(\mathcal{G}'_E || \Gamma' \vdash \mathbf{w} : A * B; \Delta')$, (II) $\mathcal{G}'_E\theta = \mathcal{G}_E$, $\Gamma'\theta = \Gamma$, $\Delta'\theta = \Delta$, $w = \mathbf{w}\theta$, and (III) $\mathcal{G}_E \cup \mathcal{G}_S = \mathcal{S}^*(\mathbb{C}, S, X)$. We need to show that we can find such triples for the premises, and more importantly, the symbolic sequents in the premises are related to the symbolic sequent in the conclusion via $*R$. In this case, the symbolic sequents are simply the following:

1. $\mathcal{G}'_E || \Gamma' \vdash \mathbf{x} : A; \mathbf{w} : A * B; \Delta'$, for the left premise,
2. $\mathcal{G}'_E || \Gamma' \vdash \mathbf{y} : A; \mathbf{w} : A * B; \Delta'$, for the right premise.

The constraint systems are: $\mathbb{C}' = (\mathcal{C} \uplus \{\mathfrak{c}_j\}, \preceq')$ for both premises, where $\mathfrak{c}_j = \mathcal{G}'_E \vdash^?_R (\mathbf{x}, \mathbf{y} \triangleright \mathbf{w})$ and $\preceq'$ is $\preceq$ extended with $\mathfrak{c}(end(X)) \preceq' \mathfrak{c}_j$. The solutions, for both premises, are the tuple $S' = (\theta', \Sigma)$ where $\theta' = \theta \cup \{\mathbf{x} \mapsto w_1,\ \mathbf{y} \mapsto w_2\}$ and $\Sigma = \{\sigma_1, \ldots, \sigma_n, \sigma\}$. It is guaranteed that θ' is enough to make both premises grounded, as $\mathbf{x}$ and $\mathbf{y}$ are the only two new free variables. The threads of free variables X_1 and X_2 for the two premises are naturally $X@[\mathbf{x}]$ and $X@[\mathbf{y}]$ respectively. By Proposition 5.7, in each premise, the following holds:

$$\mathcal{G}_E \cup \mathcal{G}'_S = \mathcal{S}(\mathcal{G}_E \cup \mathcal{G}_S, \sigma) = \mathcal{S}(\mathcal{G}_E \cup \mathcal{G}_S \cup \mathcal{G}'_E\theta, \sigma) = \mathcal{S}^*(\mathbb{C}', S', X_1) = \mathcal{S}^*(\mathbb{C}', S', X_2).$$

So by the induction hypothesis we have a symbolic derivation Π'_1 for sequent (1) and a symbolic derivation Π'_2 for sequent (2), such that $\mathbb{C}_{\beta 1} = \mathbb{C}' \circ^{X_1} \mathbb{C}(\Pi'_1)$ and $\mathbb{C}_{\beta 2} = \mathbb{C}' \circ^{X_2} \mathbb{C}(\Pi'_2)$ are both solvable. Suppose the solutions are respectively $(\theta' \cup \theta_1, \Sigma \cup \Sigma_1)$ and $(\theta' \cup \theta_2, \Sigma \cup \Sigma_2)$. Then construct Π' by applying the $*R$ rule to Π'_1 and Π'_2. Note that the variables created in Π'_1 are Π'_2 are distinct so their constraints are independent of each other. So we can construct $\mathbb{C}_p = \mathbb{C}(\Pi'_1) \circ^{\emptyset} \mathbb{C}(\Pi'_2) = (\mathcal{C}_p, \preceq_p)$, along an empty thread $\emptyset$. Now $\mathbb{C}(\Pi')$ is obtained as $(\mathcal{C}_p \uplus \{\mathfrak{c}_j\}, \preceq^{\Pi'})$, where $\preceq^{\Pi'}$ is derived as follows.

- If $\mathfrak{c} \preceq_p \mathfrak{c}'$ in $\mathbb{C}_p$, then $\mathfrak{c} \preceq^{\Pi'} \mathfrak{c}'$ in $\mathbb{C}(\Pi')$
- For any minimum constraint $\mathfrak{c}_m$ in $\mathbb{C}_p$, $\mathfrak{c}_j \preceq^{\Pi'} \mathfrak{c}_m$ in $\mathbb{C}(\Pi')$.

The solution to $\mathbb{C}_\alpha = \mathbb{C} \circ^X \mathbb{C}(\Pi')$ is constructed as the combination of the solutions to $\mathbb{C}_{\beta 1}$ and $\mathbb{C}_{\beta 2}$: $(\theta' \cup \theta_1 \cup \theta_2, \Sigma \cup \Sigma_1 \cup \Sigma_2)$. This construction of the solution is indeed valid, because the symbolic derivation that gives $\mathbb{C}_\alpha$ also yields exactly $\mathbb{C}_{\beta 1}$ and $\mathbb{C}_{\beta 2}$ (respectively on its two branches created by the $*R$ rule). □

6 A heuristic method for proof search

In this section, we first give an example of deriving a formula and solving the generated constraints in $FVLS_{BBI}$ based on a heuristic method, in which constraints generated by zero-premise rules are solved first, then the constraints from logical rules are solved based on what we have gained in the solved constraints. We then extend this idea and formalise it in the remainder of this section.

Consider again the constraint system in Example 5.1, which is generated from the symbolic derivation in Figure 9. Our heuristic constraint solving method differs from the naive method shown in Section 5 in two aspects. Firstly, we start by solving the constraints generated by zero-premise rules. Since the constraints $\mathfrak{c}_3, \mathfrak{c}_4, \mathfrak{c}_5$ are required by the *id* rule, we must accept them by assigning $\mathbf{x}_5, \mathbf{x}_7, \mathbf{x}_8$ to a_3, a_4, a_2 respectively. Then we are only left with the constraints $\mathfrak{c}_1$ and $\mathfrak{c}_2$. In the following, we shall write $(a_1, a_2 \triangleright a_0); (a_3, a_4 \triangleright a_1)$ as $\mathcal{G}_1$, and $(a_3, \mathbf{x}_6 \triangleright a_0); (a_2, a_4 \triangleright \mathbf{x}_6)$ as $\mathcal{G}_2$. Now $\mathbf{x}_6$ is the only remaining free variable. We can apply the rule A (upward) on $\mathcal{G}_1$ to obtain $(a_3, w \triangleright a_0); (a_2, a_4 \triangleright w)$, where w is a new label. Then apply the rule E (upward) to obtain $(a_4, a_2 \triangleright w)$. The two constraints can be solved by the above derivation and assigning w to $\mathbf{x}_6$.

The second novelty of our heuristic method is that we view a set of relational atoms as trees, and in certain cases, we can solve the constraints by only looking at the root and the leaves, ignoring the structure of the trees. For the running example, $\mathcal{G}_1$ is a tree tr_1 with root a_0 and leaves $\{a_2, a_3, a_4\}$, which are exactly the same as those in the tree tr_2 of $\mathcal{G}_2$, although the internal structures of tr_1 and tr_2 are different. We will show in Lemma 6.1 that, since every internal node in tr_2 is a unique free variable, and there are no ϵ labels in tr_2, it is guaranteed that there exists a sequence of structural rule applications on $\mathcal{G}_1$ to obtain a tree tr'_1, which has the same structure as tr_2, but only differing in the labels of internal nodes. Hence we can assign the labels of internal nodes in tr'_1 to the corresponding free variables in tr_2. The labels of nodes in tr'_1 may be existing labels occurring in tr_1, or fresh labels created by A, A_C applications. We will first try to match the free variables in tr_2 with existing labels in tr_1, if this is not possible, we will assign the free variable to a fresh label. In the example, $\mathbf{x}_6$ cannot be matched to any existing label, so we can globally replace $\mathbf{x}_6$ with a fresh label w, and add $\mathcal{G}_2$ to the left hand side of the successor constraints of $\mathfrak{c}_1$ in the partial order $\preceq$. The advantage of this process is that we do not care about the structural rule applications to obtain tr'_1 at all, but we know that the fresh label w as in the previous paragraph must exist and can substitute $\mathbf{x}_6$.

We can extend this method to a chain of multiple relational atoms which form a labelled binary tree. We define a labelled binary tree as a binary tree where each node is associated with a label. Each node in a labelled binary tree has a left child and a right child. The minimum labelled binary tree has a root and two leaves, which corresponds to a single relational atom. We define the following function inductively from a labelled binary tree to a set of relational atoms.

Definition 6.1. Let tr be a labelled binary tree, the set $Rel(tr)$ of relational atoms w.r.t. tr is defined as follows.

- (Base case): tr only contains a root node labelled with r and two leaves labelled with a, b respectively. Then $Rel(tr) = \{(a, b \triangleright r)\}$
- (Inductive case): tr contains a root node labelled with r, whose left and right children are labelled with a and b respectively. Then $Rel(tr) = Rel(tr_a) \cup Rel(tr_b) \cup \{(a, b \triangleright r)\}$, where tr_a and tr_b are the subtrees rooted at, respectively, the node labelled with a and b. .

The *width* of a labelled binary tree is defined as the number of leaves in the tree. A labelled binary tree is a *variant* of another labelled binary tree if either they are exactly the same, or they differ only in the labels of the internal nodes.

We say that a set $\mathcal{R}$ of relational atoms *forms a labelled binary tree tr* when $\mathcal{R} = Rel(tr)$. In this case, the leaves in tr are actually a "splitting" of the root node. Commutativity and associativity guarantee that we can split a node arbitrarily, as long as the leaves in the tree are the same. Moreover, since all internal nodes are free variables, we can assign them to either existing labels or fresh labels (created by A, A_C) without clashing with existing relational atoms. This idea is formalised in the following lemma, and is proved in Appendix A.13.

Lemma 6.1. *Given constraints $\mathfrak{c}_1 \lessdot \cdots \lessdot \mathfrak{c}_n$ with $\mathcal{G} = \mathcal{G}(\mathfrak{c}_1) = \cdots = \mathcal{G}(\mathfrak{c}_n)$ where the r.h.s. of these constraints gives the set $\mathcal{R}$ of relational atoms, $\mathfrak{c}_1, \cdots, \mathfrak{c}_n$ are solvable if the following hold:*

1. *$\mathcal{R} = Rel(tr)$, for some labelled binary tree tr where every internal node label is a free variable $\mathbf{x}$ which only occurs once in tr, and $\mathfrak{c}_1 \preceq \mathfrak{c}(\mathbf{x})$.*
2. *The other node labels in tr are non-ϵ labels.*
3. *There exist $\mathcal{G}' \subseteq \mathcal{G}$ and tr' such that $\mathcal{G}' = Rel(tr')$ and tr' has the same root and leaves as tr.*

Our heuristic method is not complete, so it gives only sufficient conditions. Nevertheless, our heuristic method seems effective in many cases, as will be demonstrated in the next section.

7 Implementation and experiments

We used a Dell Optiplex 790 desktop with Intel CORE i7 2600 @ 3.4 GHz CPU and 8GB memory as the platform, and tested the following provers on the formulae from Park et al. [20]. (1) BBeye: the OCaml prover from Park et al. based upon nested sequents [20]; (2) Naive (Vamp): translates a BBI formula into a first-order formula using the standard translation, then uses Vampire 2.6 [11] to solve it; (3) $FVLS_{BBI}$ Heuristic: backward proof search in $FVLS_{BBI}$, using the heuristic-based method to solve the set of constraints, implemented in *OCaml*.

The results are shown in Table 1. The BBeye (opt) column shows the results from Park et al's prover where the d() indicates the depth of proof search. The other two columns are for the two methods stated above. We see that naive translation is comparable with BBeye in most cases, but

Formula	BBeye	Naive	$FVLS_{BBI}$
	(opt)	(Vamp)	Heuristic
$(P \mathrel{-\!\!*} Q) \wedge (\top * (\top^* \wedge P)) \to Q$	d(2) 0	0.003	0.001
$(\top^* \mathrel{-\!\!*} \neg(\neg P * \top^*)) \to P$	d(2) 0	0.003	0.000
$\neg((P \mathrel{-\!\!*} \neg(P * Q)) \wedge ((\neg P \mathrel{-\!\!*} \neg Q) \wedge Q))$	d(2) 0	0.004	0.001
$\top^* \to ((P \mathrel{-\!\!*} (Q \mathrel{-\!\!*} R)) \mathrel{-\!\!*} ((P * Q) \mathrel{-\!\!*} R))$	d(2) 0.015	0.017	0.001
$\top^* \to ((P * (Q * R)) \mathrel{-\!\!*} ((P * Q) * R))$	d(2) 0.036	0.006	0.000
$\top^* \to ((P * ((Q \mathrel{-\!\!*} V) * R)) \mathrel{-\!\!*} ((P * (Q \mathrel{-\!\!*} V)) * R))$	d(2) 0.07	0.019	0.001
$\neg((P \mathrel{-\!\!*} \neg(\neg(U \mathrel{-\!\!*} \neg(P * (R * Q))) * P)) \wedge R * (U \wedge (P * Q)))$	d(2) 0.036	0.037	0.001
$\neg((R * (U * V)) \wedge B)$ where	d(2) 0.016	0.075	0.039
$B := ((P \mathrel{-\!\!*} \neg(\neg(Q \mathrel{-\!\!*} \neg(U * (V * R))) * P)) * (Q \wedge (P * \top)))$			
$\neg(C * (U \wedge (P * (Q * V))))$ where	d(3) 96.639	0.089	0.038
C $:= ((P \mathrel{-\!\!*} \neg(\neg(U \mathrel{-\!\!*} \neg((R * V) * (Q * P))) * P)) \wedge R)$			
$(P * (Q * (R * U))) \to (U * (R * (Q * P)))$	d(2) 0.009	0.048	0.001
$(P * (Q * (R * U))) \to (U * (Q * (R * P)))$	d(3) 0.03	0.07	0.001
$(P * (Q * (R * (U * V)))) \to (V * (U * (P * (Q * R))))$	d(3) 1.625	1.912	0.001
$(P * (Q * (R * (U * V)))) \to (V * (Q * (P * (R * U))))$	d(4) 20.829	0.333	0.001
$\top^* \to (P * ((Q \mathrel{-\!\!*} V) * (R * U)) \mathrel{-\!\!*} ((P * U) * (R * (Q \mathrel{-\!\!*} V))))$	d(3) 6.258	0.152	0.007

Table 1: Initial experimental results.

the latter is not stable. When the tested formulae involves more interaction between multiplicative connectives, BBeye runs significantly slower. The heuristic method outperforms all other methods in the tested cases.

Nonetheless, our prover is slower than BBeye for formulae which contain many occurrences of the same atomic formulae, giving (id) instances such as:

$$\Gamma; w_1 : P; w_2 : P; \cdots ; w_n : P \vdash \mathbf{x} : P; \Delta$$

We have to choose some w_i to match with $\mathbf{x}$ without knowing which choice satisfies other constraints. In the worst case, we have to try each using backtracking. Multiple branches of this form lead to a combinatorial explosion. Determinising the concrete labels (worlds) for formulae in proof search in LS_{BBI} or BBeye [20] avoids this problem. Further work is needed to solve this in $FVLS_{BBI}$.

Even though we do not claim the completeness of our heuristics method, it appears to be a fast way to solve certain problems. Completeness can be restored by fully implementing LS_{BBI} or $FVLS_{BBI}$. The derivations in LS_{BBI} are generally shorter than those in the Display Calculus or Nested Sequent Calculus for BBI. The reader can verify that most of formulae in Table 1 can even be proved by hand in a reasonable time using our labelled system. The optimisations of the implementation, however, is out of the scope of this paper.

Our second experiment features randomly generated BBI theorems. We have further implemented a prover based on LS_{BBI}, which is sound and complete for BBI. This prover will be used in the comparison with the prover for $FVLS_{BBI}$ and Park et al.'s BBeye.

A common way to generate random theorems is to simply globally replace a sub-formula in a theorem by a longer random formula. We avoid this method on purpose, since we can easily "cheat" by forcing our prover not to expand those sub-formulae until necessary. For example, we can globally replace A by $p \wedge q$, and replace B by $r \vee t$ in the BBI theorem $(A * B) \to (B * A)$, obtaining a longer theorem $((p \wedge q) * (r \vee t)) \to ((r \vee t) * (p \wedge q))$. But we can build in a mechanism in our prover that given a formula F, we search for any sub-formula F' that occurs multiple times in F, and replace F' by a "pseudo-proposition", which is only allowed to be decomposed when the prover cannot find a derivation. In this way, no matter how large the generated theorem is, the prover takes the same time to prove it.

Test	n	i	BBeye proved	BBeye avg. time	LS_{BBI} proved	LS_{BBI} avg. time	$FVLS_{BBI}$ proved	$FVLS_{BBI}$ avg. time
1	10	20	82%	0.93s	81%	0.25s	77%	0.01s
2	20	20	63%	1.43s	60%	0.73s	63%	0.01s
3	30	20	51%	4.33s	35%	0.76s	37%	0.01s
4	20	30	60%	2.24s	65%	0.77s	66%	0.01s
5	20	40	62%	2.51s	57%	1.35s	57%	0.01s
6	20	50	53%	1.20s	52%	1.15s	51%	0.01s
7	30	50	40%	2.94s	40%	1.18s	35%	0.01s

Table 2: Experiment 2 results.

We create random BBI theorems by first generating some random formulae (not necessarily theorems) of length n, and perform global substitution of these formulae to certain places in a BBI axiom schema; then we use the deduction rules $-\!\!*\ 1$ and $-\!\!*\ 2$ in Figure 3 to mutate the resultant formula in random places, repeat this step by i iterations, yielding the final theorem.

When generating random theorems, our procedure randomly chooses axioms (and deduction rules), but is biased to use the axioms in Figure 3 more often rather than the classical axioms. Given the parameter n (resp. i) as in the previous paragraph, our procedure chooses a random number ranging from 1 to n (resp. i) and proceeds as above. The mutation step is vital to generate theorems with $-\!\!*$, since BBI axioms do not involve $-\!\!*$ at all. Moreover, the "cheat" mechanism would more often fail when we use the deduction rules to mutate the formula, because the internal structure of the formula is changed. Our random theorem generation does not create theorems of a fixed length, but the length grows as n increases.

We compare the performance of our provers with Park et al.'s BBI prover BBeye against our randomly generated theorem suites in Table 2. The column with LS_{BBI} is our complete prover based on the original LS_{BBI}; the column with $FVLS_{BBI}$ is our incomplete prover based on the free-variable system using the heuristic constraint solving method; and BBeye is Park et al.'s prover. As said previously, the parameter n is the maximum length of random formulae to be substituted into a BBI axiom, i is the maximum iteration of mutation. Each test suite contains 100 BBI theorems, the "proved" column for each prover gives the percentage of successfully proved formulae within the time out, and the "avg. time" column is the average time used when a formula is proved. We set the time out as 50 seconds, if the prover cannot prove a formula within 50 seconds, the time is not counted in the average time. The size of the tested formulae only depends on n. When $n = 10$, an average formula has about 20 logical connectives, this number is increased to around 50 when $n = 30$. The parameter i decides how different the formulae are from the original BBI axioms.

Table 2 shows that, BBeye has a slightly higher successful rate in general when timeout is set to 50 seconds. However, BBeye spends more time than others on successful attempts. $FVLS_{BBI}$ is the fastest prover in comparison, with an average of 0.01 second on successful attempts no matter what the parameters are. Comparing test suites 1,2,3, we see that the successful rate of LS_{BBI} and $FVLS_{BBI}$ drop faster than BBeye when the size of the formula increases. Test suite 4 is an outlier in which LS_{BBI} and $FVLS_{BBI}$ proved more formulae than BBeye within the timeout. Comparing test suites 2,5,6, we see the advantage of BBeye diminishes when the iteration of mutation increases. The same phenomenon happens when comparing test suites 3 and 7. So when dealing with more "complex" formulae, BBeye has similar successful rates as our prover based on LS_{BBI}.

$$\frac{(a,b\triangleright c);\Gamma[c/d]\vdash\Delta[c/d]}{(a,b\triangleright c);(a,b\triangleright d);\Gamma\vdash\Delta}\ {\scriptstyle P}\qquad\frac{(a,b\triangleright c);\Gamma\vdash\Delta}{\Gamma\vdash\Delta}\ {\scriptstyle T}$$

$$\frac{(\epsilon,\epsilon\triangleright\epsilon);\Gamma[\epsilon/a][\epsilon/b]\vdash\Delta[\epsilon/a][\epsilon/b]}{(a,b\triangleright\epsilon);\Gamma\vdash\Delta}\ {\scriptstyle IU}\qquad\frac{(a,b\triangleright c);\Gamma[b/d]\vdash\Delta[b/d]}{(a,b\triangleright c);(a,d\triangleright c);\Gamma\vdash\Delta}\ {\scriptstyle C}$$

In T, a, b do occur in the conclusion but c does not
In all substitutions $[y/x]$, $x \neq \epsilon$

Figure 10: Some auxiliary structural rules.

8 Conclusion and Future Work

Our main contribution is a labelled sequent calculus for BBI_{ND} that is sound, complete, and enjoys cut-elimination. There are no explicit contraction rules in LS_{BBI} and all structural rules can be restricted so that proof search is entirely driven by logical rules. We further propose a free variable system to restrict the proof search space so that some applications of $*R$, $-\!\!* \, L$ rules can be guided by zero-premise rules. Although we can structure proof search to be more manageable compared to the unrestricted (labelled or display) calculus, the undecidability of BBI implies that there is no terminating proof search strategy for a sound and complete system. The essence of proof search resides in guessing which relational atoms to use in the $*R$ and $-\!\!* \, L$ rules and whether they need to be applied more than once to a formula. Nevertheless, our initial experimental results raise the hope that a more efficient proof search strategy can be developed based on our calculus.

An immediate task is to find a complete and terminating (if possible) constraint solving strategy. We also plan to investigate counter-model construction for BBI_{ND}. Such a countermodel construction has been studied for BBI_{PD} in [13, 12].

Another interesting topic is to extend our calculus to handle some semantics other than the non-deterministic monoidal ones. Our design of the structural rules in LS_{BBI} can be generalised as follows. If there is a semantic condition of the form $(w_{11}, w_{12} \triangleright w_{13}) \wedge \cdots \wedge (w_{i1}, w_{i2} \triangleright w_{i3}) \Rightarrow (w'_{11}, w'_{12} \triangleright w'_{13}) \wedge \cdots \wedge (w'_{j1}, w'_{j2} \triangleright w'_{j3}) \wedge (x_{11} = x_{12}) \wedge \cdots \wedge (x_{k1} = x_{k2})$, we create a rule:

$$\frac{(w'_{11}, w'_{12} \triangleright w'_{13}); \cdots ; (w'_{j1}, w'_{j2} \triangleright w'_{j3}); (w_{11}, w_{12} \triangleright w_{13}); \cdots ; (w_{i1}, w_{i2} \triangleright w_{i3}); \Gamma \vdash \Delta}{(w_{11}, w_{12} \triangleright w_{13}); \cdots ; (w_{i1}, w_{i2} \triangleright w_{i3}); \Gamma \vdash \Delta}\ {\scriptstyle r}$$

And apply substitutions $[x_{12}/x_{11}] \cdots [x_{k2}/x_{k1}]$ globally on the premise, where ϵ is not substituted. Many additional features can be added in this way. We summarise the following desirable ones: (1) PD-semantics: the composition of two elements is either the empty set or a singleton, i.e., $(a, b \triangleright c) \wedge (a, b \triangleright d) \Rightarrow (c = d)$; (2) TD-semantics: the composition of any two elements is always defined as a singleton, i.e., $\forall a, b, \exists c$ s.t. $(a, b \triangleright c)$; (3) indivisible unit: (cf. Section 1) $(a, b \triangleright \epsilon) \Rightarrow (a = \epsilon) \wedge (b = \epsilon)$; and (4) cancellativity: if $w \circ w'$ is defined and $w \circ w' = w \circ w''$, then $w' = w''$, i.e., $(a, b \triangleright c) \wedge (a, d \triangleright c) \Rightarrow (b = d)$. Note that (2) and (4) are in addition to (1). The above are formalised in the rules P, T, IU, C respectively in Figure 10.

The formula $(F * F) \rightarrow F$, where $F = \neg(\top -\!\!* \neg \top^*)$, differentiates BBI_{ND} and BBI_{PD} [15] and is provable using $LS_{BBI} + P$. Using $LS_{BBI} + T$, we can prove $(\neg \top^* -\!\!* \bot) \rightarrow \top^*$, which is valid in BBI_{TD} but not in BBI_{PD} [15], and also $(\top^* \wedge ((p * q) -\!\!* \bot)) \rightarrow ((p -\!\!* \bot) \vee (q -\!\!* \bot))$, which is valid in separation models iff the composition is total [5]. These additional rules do not break cut-elimination.

Except for the TD-semantics for BBI, the above properties, along with others in *separation* *theory* [9, 6], such as disjointness, cross-split, splittability, can all be captured by using our labelled method [12]. Totality is not considered in that work, as it is less frequently used in applications, and causes inefficiency in proof search.

The above treatment is not the first one to consider other semantics for BBI. Larchey-Wendling and Galmiche's labelled tableau calculus [16] can be converted into a labelled sequent calculus that has the same set of logical rules as that of LS_{BBI}, the difference lies in the rules for capturing the semantics. Our structural rules directly encode the properties of the non-deterministic monoidal semantics of BBI (i.e., identity, commutativity, and associativity) by explicitly using ternary relational atoms. In comparison, Larchey-Wendling and Galmiche's tableau calculus indirectly captures the partial-deterministic semantics via a set of rules for *Partial Monoidal Equivalences* (PMEs). Their rules do not employ ternary relations, but treat a combination of worlds as a string, building in the partial-determinism reading. This could be part of the reason why their tableau calculus can be further specialised to capture other properties in separation theory, but it is hard to be generalised to capture the non-deterministic semantics. The tableau method also differs from ours in that its completeness is proved via a counter-model construction, whereas our completeness is proved by simulating the Hilbert system for BBI. In fact, there does not exist a Hilbert system for BBI_{PD} [6], thus their counter-model construction is necessary.

Oddly, the formula $\neg(\top^* \land A \land (B * \neg(C {-\!\!*}\, (\top^* \to A))))$, which is valid in BBI_{ND}, is very hard to prove in the display calculus and Park et al.'s method. We ran this formula using Park et al.'s prover for a week on a CORE i7 2600 processor, without success. Very short proofs of this formula exist in LS_{BBI} or Larchey-Wendling and Galmiche's labelled tableau (this formula must also be valid in BBI_{PD}). We are currently investigating this phenomenon. The proofs for the formulae in this section can be found in Appendix A.14.

References

[1] Bernhard Beckert and Rajeev Goré. Free-variable tableaux for propositional modal logics. *Studia Logica*, 69(1):59–96, 2001.

[2] James Brotherston. A unified display proof theory for bunched logic. *ENTCS*, 265:197–211, September 2010.

[3] James Brotherston and Cristiano Calcagno. Classical BI: Its semantics and proof theory. *LMCS*, 6(3), 2010.

[4] James Brotherston and Max Kanovich. Undecidability of propositional separation logic and its neighbours. In *LICS*, pages 130–139, 2010.

[5] James Brotherston and Max I. Kanovich. Undecidability of propositional separation logic and its neighbours. *J. ACM*, 61(2):14, 2014.

[6] James Brotherston and Jules Villard. Parametric completeness for separation theories. In *POPL*, pages 453–464. ACM, 2014.

[7] Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang. Local action and abstract separation logic. In *LICS*, pages 366–378. IEEE, 2007.

[8] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zalinescu. Deciding security properties for cryptographic protocols: application to key cycles. *ACM Trans. Comput. Log.*, 11(2), 2010.

[9] Robert Dockins, Aquinas Hobor, and Andrew W. Appel. A fresh look at separation algebras and share accounting. In *APLAS*, volume 5904 of *LNCS*, pages 161–177, 2009.

[10] Didier Galmiche and Dominique Larchey-Wendling. Expressivity properties of Boolean BI through relational models. In *FSTTCS*, pages 358–369, 2006.

[11] Krystof Hoder and Andrei Voronkov. Comparing unification algorithms in first-order theorem proving. KI'09, pages 435–443. Springer-Verlag, 2009.

[12] Zhe Hou, Ranald Clouston, Rajeev Goré, and Alwen Tiu. Proof search for propositional abstract separation logics via labelled sequents. In *POPL*, pages 465–476. ACM, 2014.

[13] Dominique Larchey-Wendling. The formal strong completeness of partial monoidal Boolean BI. *Journal of Logic and Computation*, 2014.

[14] Dominique Larchey-Wendling and Didier Galmiche. Exploring the relation between intuitionistic BI and Boolean BI: An unexpected embedding. *MSCS*, 19(3):435–500, 2009.

[15] Dominique Larchey-Wendling and Didier Galmiche. The undecidability of Boolean BI through phase semantics. *LICS*, 0:140–149, 2010.

[16] Dominique Larchey-Wendling and Didier Galmiche. Non-deterministic phase semantics and the undecidability of Boolean BI. *ACM TOCL*, 14(1), 2013.

[17] Sara Negri. Proof analysis in modal logic. *JPL*, 34(5-6):507–544, 2005.

[18] Sara Negri and Jan von Plato. *Structural Proof Theory.* CUP, 2001.

[19] Peter W. O'Hearn and David J. Pym. The logic of bunched implications. *BSL*, 5(2):215–244, 1999.

[20] Jonghyun Park, Jeongbong Seo, and Sungwoo Park. A theorem prover for Boolean BI. POPL '13, pages 219–232, New York, NY, USA, 2013. ACM.

[21] David J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications.* Applied Logic Series. Kluwer Academic Publishers, 2002.

[22] John C. Reynolds. Separation logic: A logic for shared mutable data structures. LICS '02, pages 55–74. IEEE Computer Society, 2002.

[23] Alwen Tiu, Rajeev Goré, and Jeremy E. Dawson. A proof theoretic analysis of intruder theories. *Logical Methods in Computer Science*, 6(3), 2010.

[24] Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory.* CUP, 1996.

A Appendix

This section provides the details of the proofs in this paper.

A.1 Soundness of LS_{BBI}

Proof for Theorem 2.1.

Proof. To prove the soundness of LS_{BBI}, we show that each rule preserves falsifiability upwards, as this is a more natural direction in terms of backward proof search. Therefore to prove that a rule is sound, we need to show that if the conclusion is falsifiable, then at least one of the premises is falsifiable (usually in the same choice of v, ρ, and $\mathcal{M}$). Most of the cases are easy, we show some samples here.

id Since there is no premise in this rule, we simply need to show that the conclusion is not falsifiable.

Suppose the sequent $\Gamma; w : P \vdash w : P; \Delta$ is falsifiable, then Γ must be true and $\rho(w) \Vdash A$ and $\rho(w) \not\Vdash A$ and Δ must be false. However, $\rho(w) \Vdash A$ and $\rho(w) \not\Vdash A$ cannot hold at the same time for any $(\mathcal{M}, \triangleright, \epsilon)$, v and ρ, so we have a contradiction, thus this sequent is not falsifiable.

$\top^* L$ Assume $\Gamma; w : \top^* \vdash \Delta$ is falsifiable, then Γ is true and $\rho(w) \Vdash \top^*$ and Δ is false.

From the semantics of $\top^*$ we know that $\rho(w) \Vdash \top^*$ iff $\rho(w) = \epsilon$. Therefore by choosing the same ρ, v, and $\mathcal{M}$ for the premise, replacing every w by ϵ in Γ and Δ preserves their valuations, as we know that $\rho(\epsilon) = \epsilon$. That is, $\Gamma[\epsilon/w]$ must be true and $\Delta[\epsilon/w]$ must be false. So the premise is falsifiable.

$*L$ Assume the conclusion is falsifiable, so under some v, ρ, $\mathcal{M}$, we have that Γ is true and $\rho(z) \Vdash A * B$ and Δ is false.

From the semantics of $A * B$, we know that $\exists a, b$ s.t. $a, b \triangleright \rho(z)$ and $a \Vdash A$ and $b \Vdash B$. So we can choose a mapping ρ' with $\rho' = (x \mapsto a) \cup (y \mapsto b) \cup \rho$. Since x and y are fresh, they should not affect anything in ρ. Then, under ρ', the following hold: $(x, y \triangleright z)$ is true and Γ is true and $\rho'(x) \Vdash A$ and $\rho'(y) \Vdash B$ and Δ is false. Thus the premise is falsifiable in v, ρ', and $\mathcal{M}$.

$*R$ Assume under some v, ρ, and $\mathcal{M}$, $(x, y \triangleright z)$ is true and Γ is true and $\rho(z) \not\Vdash A * B$ and Δ is false.

The semantics of $A * B$ yields the following:

$$\begin{aligned} \rho(z) \not\Vdash A * B &\Leftrightarrow \neg(\exists a, b.\ (a, b \triangleright \rho(z)\ \mathit{and}\ a \Vdash A\ \mathit{and}\ b \Vdash B)) \\ &\Leftrightarrow \forall a, b.\ (a, b \triangleright \rho(z)\ \mathit{doesn't\ hold\ or}\ a \not\Vdash A\ \mathit{or}\ b \not\Vdash B) \end{aligned}$$

If we pick the same set of v, ρ, $\mathcal{M}$ for the premises, however, in both premises the relational atom $(x, y \triangleright z)$ already exists, which means $\rho(x), \rho(y) \triangleright \rho(z)$ holds. So the possibility is only that either $\rho(x) \not\Vdash A$ or $\rho(y) \not\Vdash B$. Assume the former one holds, then the left premise is falsifiable, otherwise the right premise is falsifiable.

Rules for additive connectives are straightforward, the cases for $-\!*$ can be proved similarly as for $*$ above. Structural rules E, A (and A_C), Eq_1 (and Eq_2 and U) can be proved by using the commutativity, associativity, and identity properties of the monoid structure respectively. □

A.2 Substitution for labels

The proof for Lemma 3.1.

Proof. By induction on $ht(\Pi)$.
(Base case) If $ht(\Pi) = 0$, then the only applicable rules are id, $\bot L$, $\top R$ and $\top^* R$. If the label $x \neq \epsilon$ being substituted is not on the principal formula, then the substitution does not affect the original derivation. Note that since we do not allow to substitute for the label ϵ, the proof for $\top^* R$ can only be this case. Otherwise we obtain the new derivation by simply replacing the label of the principal formula.
(Inductive case) If $ht(\Pi) > 0$, then consider the last rule applied in the derivation. We consider three main cases.

1. Neither x nor y is the label of the principal formula.

   (a) Suppose the last rule applied is $\top^* L$, and $x \neq w$ and $y \neq w$, and Π is the following derivation:

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma'[\epsilon/w] \vdash \Delta[\epsilon/w]\end{array}}{\Gamma'; w : \top^* \vdash \Delta}\ \top^* L$$

   By the induction hypothesis, there is a derivation Π_1' of $\Gamma'[\epsilon/w][y/x] \vdash \Delta[\epsilon/w][y/x]$ with $ht(\Pi_1') \leq ht(\Pi_1)$. Since x and y are different from w, this sequent is equal to $\Gamma'[y/x][\epsilon/w] \vdash \Delta[y/x][\epsilon/w]$. Therefore Π' is constructed as follows.

$$\dfrac{\begin{array}{c}\Pi_1'\\ \Gamma'[y/x][\epsilon/w] \vdash \Delta[y/x][\epsilon/w]\end{array}}{\Gamma'[y/x]; w : \top^* \vdash \Delta[y/x]}\ \top^* L$$

   Obviously $ht(\Pi') \leq ht(\Pi)$.

   (b) If the last rule applied is Eq_1, we distinguish the following cases: x is not w or w'; $x = w$; $x = w'$.

      i. $x \neq w$ and $x \neq w'$. The original derivation is as follows.

$$\dfrac{\begin{array}{c}\Pi_1\\ (\epsilon, w \triangleright w); \Gamma'[w/w'] \vdash \Delta[w/w']\end{array}}{(\epsilon, w' \triangleright w); \Gamma' \vdash \Delta}\ Eq_1$$

      A. If $y \neq w$ and $y \neq w'$, by the induction hypothesis, there is a derivation Π_1' of $(\epsilon, w \triangleright w); \Gamma'[w/w'][y/x] \vdash \Delta[w/w'][y/x]$ with $ht(\Pi_1') \leq ht(\Pi_1)$. Since x, y, w, w' are different labels, this sequent is equal to $(\epsilon, w \triangleright w); \Gamma'[y/x][w/w'] \vdash \Delta[y/x][w/w']$. Thus the derivation Π' is constructed as follows.

$$\dfrac{\begin{array}{c}\Pi_1'\\ (\epsilon, w \triangleright w); \Gamma'[y/x][w/w'] \vdash \Delta[y/x][w/w']\end{array}}{(\epsilon, w' \triangleright w); \Gamma'[y/x] \vdash \Delta[y/x]}\ Eq_1$$

      B. If $y = w$, this case is similar to Case 1.(b).i.A.

      C. Suppose $y = w'$. Then we need to derive $(\epsilon, y \triangleright w); \Gamma'[y/x] \vdash \Delta[y/x]$. If $y \neq \epsilon$, we construct Π' by first applying Eq_1 bottom-up:

$$\frac{(\epsilon, w \triangleright w); \Gamma'[y/x][w/y] \vdash \Delta[y/x][w/y]}{(\epsilon, y \triangleright w); \Gamma'[y/x] \vdash \Delta[y/x]}\ Eq_1$$

Now the premise is equal to $(\epsilon, w \triangleright w); \Gamma'[w/y][w/x] \vdash \Delta[w/y][w/x]$, and by the induction hypothesis, there is a derivation Π_1' of this sequent, with $ht(\Pi_1') \leq ht(\Pi_1)$.
If $y = \epsilon$, then we need to apply Eq_2, instead of Eq_1:

$$\frac{(\epsilon, \epsilon \triangleright \epsilon); \Gamma'[\epsilon/x][\epsilon/w] \vdash \Delta[\epsilon/x][\epsilon/w]}{(\epsilon, \epsilon \triangleright w); \Gamma'[\epsilon/x] \vdash \Delta[\epsilon/x]}\ Eq_2$$

Note that the sequent $(\epsilon, \epsilon \triangleright \epsilon); \Gamma'[\epsilon/x][\epsilon/w] \vdash \Delta[\epsilon/x][\epsilon/w]$ is the same as

$$(\epsilon, \epsilon \triangleright \epsilon); \Gamma'[w/w'][\epsilon/w][\epsilon/x] \vdash \Delta[w/w'][\epsilon/w][\epsilon/x].$$

So the premise can be proved by two successive applications of the induction hypothesis to Π_1, one using substitution $[\epsilon/w]$ and the other using substitution $[\epsilon/x]$. Here we can apply the induction hypothesis twice to Π_1 because substitution does not increase the height of derivations.

ii. $x = w$ (so w cannot be ϵ).

A. If $y \neq w'$, then Π has the form:

$$\frac{\begin{array}{c}\Pi_1\\(\epsilon, x \triangleright x); \Gamma'[x/w'] \vdash \Delta[x/w']\end{array}}{(\epsilon, w' \triangleright x); \Gamma' \vdash \Delta}\ Eq_1$$

By the induction hypothesis we have the folowing derivation:

$$\begin{array}{c}\Pi_1'\\(\epsilon, y \triangleright y); \Gamma'[x/w'][y/x] \vdash \Delta[x/w'][y/x]\end{array}$$

The end sequent is equal to the following:

$$(\epsilon, y \triangleright y); \Gamma'[y/x][y/w'] \vdash \Delta[y/x][y/w'].$$

Then by using Eq_1, we construct Π' as follows.

$$\frac{\begin{array}{c}\Pi_1'\\(\epsilon, y \triangleright y); \Gamma'[y/x][y/w'] \vdash \Delta[y/x][y/w']\end{array}}{(\epsilon, w' \triangleright y); \Gamma'[y/x] \vdash \Delta[y/x]}\ Eq_1$$

B. If $y = w'$, then Π has the form:

$$\frac{\begin{array}{c}\Pi_1\\(\epsilon, x \triangleright x); \Gamma'[x/y] \vdash \Delta[x/y]\end{array}}{(\epsilon, y \triangleright x); \Gamma' \vdash \Delta}\ Eq_1$$

By the induction hypothesis, we have the following derivation:

$$\begin{array}{c}\Pi_1'\\(\epsilon, y \triangleright y); \Gamma'[x/y][y/x] \vdash \Delta[x/y][y/x]\end{array}$$

Since in the end sequent, we replace every y by x, and then change every x back to y, the effect is the same as just keeping every y unchanged and only replace every x by y. Thus the end sequent is equal to:

$$(\epsilon, y \triangleright y); \Gamma'[y/x] \vdash \Delta[y/x]$$

which is exactly what we need to derive. Therefore we let $\Pi' = \Pi_1'$. Notice that in this case $ht(\Pi') < ht(\Pi)$.

iii. $x = w'$.

A. If $y \neq w$ and $y \neq \epsilon$, the original derivation is as follows.

$$\dfrac{\begin{matrix}\Pi_1\\(\epsilon, w \triangleright w); \Gamma'[w/x] \vdash \Delta[w/x]\end{matrix}}{(\epsilon, x \triangleright w); \Gamma' \vdash \Delta}\, Eq_1$$

By the induction hypothesis (instead of replacing every x by y, we now replace every y by w), we have the following derivation:

$$\begin{matrix}\Pi_1'\\(\epsilon, w \triangleright w); \Gamma'[w/x][w/y] \vdash \Delta[w/x][w/y]\end{matrix}$$

The end sequent is equal to:

$$(\epsilon, w \triangleright w); \Gamma'[y/x][w/y] \vdash \Delta[y/x][w/y]$$

Thus Π' is constructed as follows.

$$\dfrac{\begin{matrix}\Pi_1'\\(\epsilon, w \triangleright w); \Gamma'[y/x][w/y] \vdash \Delta[y/x][w/y]\end{matrix}}{(\epsilon, y \triangleright w); \Gamma'[y/x] \vdash \Delta[y/x]}\, Eq_1$$

B. If $y = \epsilon$ and $w \neq \epsilon$, we need to derive the following sequent:

$$(\epsilon, \epsilon \triangleright w); \Gamma'[\epsilon/x] \vdash \Delta[\epsilon/x]$$

By induction hypothesis, replacing every w by ϵ in Π_1, then using the rule Eq_2, we get the new derivation:

$$\dfrac{\begin{matrix}\Pi_1'\\(\epsilon, \epsilon \triangleright \epsilon); \Gamma'[\epsilon/x][\epsilon/w] \vdash \Delta[\epsilon/x][\epsilon/w]\end{matrix}}{(\epsilon, \epsilon \triangleright w); \Gamma'[\epsilon/x] \vdash \Delta[\epsilon/x]}\, Eq_2$$

C. If $y = w$, then the premise of the last rule is exactly what we need to derive.

(c) If the last rule applied is Eq_2, we consider three cases: $x \neq w$ and $y \neq w$; $x = w$; and $y = w$. These are symmetric to the case where the last rule is Eq_1, already discussed above.

2. y is the label of the principal formula. Most of the cases follow similarly as above, except for $\top^* L$. In this case the original derivation is as follows.

$$\dfrac{\begin{matrix}\Pi_1\\\Gamma'[\epsilon/y] \vdash \Delta[\epsilon/y]\end{matrix}}{\Gamma'; y : \top^* \vdash \Delta}\, \top^* L$$

Our goal is to derive $\Gamma'[y/x]; y : \top^* \vdash \Delta[y/x]$. Applying $\top^* L$ as in backward proof search, we get

$$\Gamma'[y/x][\epsilon/y] \vdash \Delta[y/x][\epsilon/y]$$

Note that this sequent is equal to $\Gamma'[\epsilon/y][\epsilon/x] \vdash \Delta[\epsilon/y][\epsilon/x]$, and from induction hypothesis we know that there is a derivation of this sequent of height less than or equal to $ht(\Pi)$.

3. x is the label of the principal formula.

(a) For the additive rules, since the labels stay the same in the premises and conclusions of the rules, even if the label of the principal formula is replaced by some other label, we can still apply the induction hypothesis on the premise, then use the rule to derive the conclusion.

For $\wedge L$,

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma'; x : A; x : B \vdash \Delta\end{array}}{\Gamma'; x : A \wedge B \vdash \Delta}\wedge L \quad\rightsquigarrow\quad \dfrac{\begin{array}{c}\Pi'_1\\ \Gamma'[y/x]; y : A; y : B \vdash \Delta[y/x]\end{array}}{\Gamma'[y/x]; y : A \wedge B \vdash \Delta[y/x]}\wedge L$$

For $\wedge R$,

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma' \vdash x : A; \Delta\end{array} \qquad \begin{array}{c}\Pi_2\\ \Gamma' \vdash x : B; \Delta\end{array}}{\Gamma' \vdash x : A \wedge B; \Delta}\wedge R \quad\rightsquigarrow$$

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Gamma'[y/x] \vdash y : A; \Delta[y/x]\end{array} \qquad \begin{array}{c}\Pi'_2\\ \Gamma'[y/x] \vdash y : B; \Delta[y/x]\end{array}}{\Gamma'[y/x] \vdash y : A \wedge B; \Delta[y/x]}\wedge R$$

For $\to L$,

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma' \vdash x : A; \Delta\end{array} \qquad \begin{array}{c}\Pi_2\\ \Gamma'; x : B \vdash \Delta\end{array}}{\Gamma'; x : A \to B \vdash \Delta}\to L \quad\rightsquigarrow$$

$$\dfrac{\begin{array}{c}\Pi'_1\\ \Gamma'[y/x] \vdash y : A; \Delta[y/x]\end{array} \qquad \begin{array}{c}\Pi'_2\\ \Gamma'[y/x]; y : B \vdash \Delta[y/x]\end{array}}{\Gamma'[y/x]; y : A \to B \vdash \Delta[y/x]}\to L$$

For $\to R$,

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma'; x : A \vdash x : B; \Delta\end{array}}{\Gamma' \vdash x : A \to B; \Delta}\to R \quad\rightsquigarrow\quad \dfrac{\begin{array}{c}\Pi'_1\\ \Gamma'[y/x]; y : A \vdash y : B; \Delta[y/x]\end{array}}{\Gamma'[y/x] \vdash y : A \to B; \Delta[y/x]}\to R$$

(b) For multiplicative rules that do not produce eigenvariables ($*R, -\!\!* L, \top^* L$), we can proceed similarly as in the additive cases, except for the $\top^* L$ rule. For the $\top^* L$ rule, if the label x of the principal formula is replaced by some (other) label y, i.e., Π is

$$\dfrac{\begin{array}{c}\Pi_1\\ \Gamma'[\epsilon/x] \vdash \Delta[\epsilon/x]\end{array}}{\Gamma'; x : \top^* \vdash \Delta}\top^* L$$

then we then need a derivation of the sequent $\Gamma'[y/x]; y : \top^* \vdash \Delta[y/x]$. Using $\top^* L$ rule we have:

$$\dfrac{\Gamma[y/x][\epsilon/y] \vdash \Delta[y/x][\epsilon/y]}{\Gamma[y/x]; y : \top^* \vdash \Delta[y/x]}\top^* L$$

Note that the premise now is equal to $\Gamma[\epsilon/x][\epsilon/y] \vdash \Delta[\epsilon/x][\epsilon/y]$, and can be proved using the induction hypothesis on Π_1.

If $y = \epsilon$, then Π' is obtained by applying Lemma A.3.1 to Π_1.

(c) For the multiplicative rules that have eigenvariables ($*L$ and $-\!\!* \, R$), if the label of the principal formula is replaced by a label other than the newly created labels in the rules, then we proceed similarly as in additive cases. If the label of the principal formula is replaced by one of the newly created labels, then we just need to create a different new label in the new relation.

For $*L$, we have the derivation:

$$\dfrac{\begin{matrix}\Pi_1\\ (y, z \triangleright x); \Gamma'; y : A; z : B \vdash \Delta\end{matrix}}{\Gamma'; x : A * B \vdash \Delta} *L$$

If x is substituted by y (the case for substituting to z is symmetric), then we need a derivation of $\Gamma'[y/x]; y : A * B \vdash \Delta[y/x]$. Note that since the $*L$ rule requires the relation $(y, z \triangleright x)$ to be fresh, so in the original derivation y and z cannot be in Γ or Δ. Therefore by induction hypothesis we must have a derivation Π_1' for

$$(y', z' \triangleright x); \Gamma'; y' : A; z' : B \vdash \Delta,$$

where y' and z' are new labels, such that $ht(\Pi_1') \leq ht(\Pi_1)$. Applying the induction hypothesis again to Π_1', we have a derivation Π_1'' $(y', z' \triangleright y); \Gamma'[y/x]; y' : A; z' : B \vdash \Delta[y/x]$, with $ht(\Pi_1'') \leq ht(\Pi_1)$. Thus the derivation Π' is constructed as follows.

$$\dfrac{\begin{matrix}\Pi_1''\\ (y', z' \triangleright y); \Gamma'[y/x]; y' : A; z' : B \vdash \Delta[y/x]\end{matrix}}{\Gamma'[y/x]; y : A * B \vdash \Delta[y/x]} *L$$

The case for $-\!\!* \, R$ is similar. suppose Π is:

$$\dfrac{\begin{matrix}\Pi_1\\ (y, x \triangleright z); \Gamma; y : A \vdash z : B; \Delta'\end{matrix}}{\Gamma \vdash x : A -\!\!* \, B; \Delta'} -\!\!* \, R$$

If x is replaced by y, then we have the following derivation.

$$\dfrac{\begin{matrix}\Pi_1'\\ (y', y \triangleright z'); \Gamma[y/x]; y' : A \vdash z' : B; \Delta'[y/x]\end{matrix}}{\Gamma[y/x] \vdash y : A -\!\!* \, B; \Delta'[y/x]} -\!\!* \, R$$

If x is replaced by z, then we have the following derivation.

$$\dfrac{\begin{matrix}\Pi_1'\\ (y', z \triangleright z'); \Gamma[z/x]; y' : A \vdash z' : B; \Delta'[z/x]\end{matrix}}{\Gamma[z/x] \vdash z : A -\!\!* \, B; \Delta'[z/x]} -\!\!* \, R$$

□

A.3 Weakening admissibility of LS_{BBI}

Lemma A.3.1. *For all structures Γ, Δ, labelled formula $w : A$, and ternary relation $(x, y \triangleright z)$, if $\Gamma \vdash \Delta$ is derivable, then there exists a derivation of the same height for each of the following sequents:*

$$\Gamma; w : A \vdash \Delta \qquad \Gamma \vdash w : A; \Delta \qquad (x, y \triangleright z); \Gamma \vdash \Delta.$$

Proof. By induction on $ht(\Pi)$. Since id, $\bot L$, $\top R$, and $\top^* R$ all have weakening built in, the base case trivially holds. For the inductive cases, the only nontrivial case is for $*L$ and $-\!\!* \, R$, where new labels have to be introduced. These labels can be systematically renamed to make sure that they do not clash with the labels in the weakened formula/relational atom. □

This yields the proof for Lemma 3.2 in the paper. Furthermore, we can prove more useful lemmas based on the weakening property.

We next prove a 'strengthening' property, which is the converse of weakening property of derivation and which will be useful in proving cut-elimination. That is, if a formula is never principal in a derivation, it can obviously be omitted.

Lemma A.3.2. *If $w : A$ is not the principal formula of any rule application in the derivation of $\Gamma; w : A \vdash \Delta$ ($\Gamma \vdash w : A; \Delta$ resp.), then there is a derivation of $\Gamma \vdash \Delta$ with the same series of rule applications.*

A consequence of this lemma is that certain formula occurrences to which no introduction rules can be applied (thus they can never be principal in any derivation) can be removed from a derivation without affecting the validity of the derivation. Examples of this are formulas $\epsilon : \top^*$ or $x : \top$ occuring on the left hand side of a sequent.

Lemma A.3.3. *If $\Gamma; \epsilon : \top^* \vdash \Delta$ is derivable, then $\Gamma \vdash \Delta$ is derivable with the same series of rule applications.*

Lemma A.3.4. *If $\Gamma; w : \top \vdash \Delta$ is derivable, then $\Gamma \vdash \Delta$ is derivable with the same series of rule applications.*

A.4 Invertibility of rules in LS_{BBI}

Proof for Lemma 3.3.

Proof. As the additive rules in LS_{BBI} are exactly the same as those in Negri's labelled system for Modal logic or $G3c$ (cf. [18]), the proof for them is similar. The main difference is that the rest of our rules are of different forms. However, as most of our rules do not modify the side structures, simply by applying the induction hypothesis and then using the corresponding rule, we get the new derivation. The cases where the last rule applied is $\top^* L$, Eq_1, or Eq_2 follow essentially the same, except a global substitution needs to be considered, but that is of no harm.

Rules E, A, U, A_C, $*R$ and $-\!\!* \, L$ are trivially invertible as the conclusion is a subset of the premise, and weakening is height-preserving admissible.

To prove the cases for $*L$ and $-\!\!* \, R$, we do inductions on the height n of the derivation. In each case below, it is obvious that each premise is always cut-free derivable with less or same height as the conclusion.

The case for $*L$ is as follows.

(Base case) If $n = 0$, then the conclusion of $*L$ is one of the conlucsions of id, $\bot L$, $\top R$, $\top^* R$, notice that the identity rule is restricted to propositions, therefore the premise of $*L$ is also the conclusions of the corresponding axiom rule.
(Inductive case) If $n > 0$, and the last rule applied is not $*L$ or $-\!\!* R$, then no fresh labels are involved, so we can safely apply the induction hypothesis on the premise of the last rule and then use the rule to get the derivation. If the last rule is $*L$ or $-\!\!* R$, but the principal formula is in Γ or Δ, we proceed similarly, and use the Substitution Lemma to ensure that the eigenvariables are new. If the principal formula is $z : A * B$, then the premise of the last rule yields the desired conclusion.

The case for $-\!\!* R$ follows similarly.

For $\top^* L$, again, we do an induction on the height n of the derivation.
(Base case) If $n = 0$, then $\Gamma; x : \top^* \vdash \Delta$ is the conclusion of one of id, $\bot L$, $\top R$, $\top^* R$, and $x : \top^*$ cannot be the principal formula. Note that in the first three cases the principal formulae can be labelled with anything. Since, in the sequent $\Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$, the label x is uniformly replaced by ϵ, this sequent can be the conclusion of the corresponding rule as well. For $\top^* R$, since $\top^*$ on the right hand side can only be labelled with ϵ, so replacing x to ϵ does not change its label. Thus this case is not broken either.
(Inductive case) If $n > 0$, consider the last rule applied in the derivation.

1. If the principal formula or relation does not involve the label x, then we can apply the induction hypothesis directly on the premise of the last rule, then use the last rule to get the derivation.

2. Otherwise, if the principal formula or relation has label x, and the last rule is not $\top^* L$, we proceed similarly, except replacing the label in the principal relation or formula. The detail is exemplified using $*L$.

   For $*L$, we have the following derivation:

$$\frac{\begin{array}{c}\Pi \\ (y, z \triangleright x); \Gamma; x : \top^*; y : A; z : B \vdash \Delta\end{array}}{\Gamma; x : \top^*; x : A * B \vdash \Delta} *L$$

   The condition of the rule $*L$ guarantees that y and z cannot be in Γ and Δ, so we do not have to worry if they are identical to x. By applying the induction hypothesis and then using the rule, we get the following derivation:

$$\frac{\begin{array}{c}\Pi' \\ (y, z \triangleright \epsilon); \Gamma[\epsilon/x]; y : A; z : B \vdash \Delta[\epsilon/x]\end{array}}{\Gamma[\epsilon/x]; \epsilon : A * B \vdash \Delta[\epsilon/x]} *L$$

   Another way to do this is by using the Substitution Lemma, replacing x by ϵ, we get a derivation to the premise that has a redundant $\epsilon : \top^*$, since we know that this labelled formula on the left hand side does not contribute to the derivation, we can safely derive the sequent without it using the same inference, cf. Lemma A.3.2.

   The case where the last rule is $-\!\!* R$ is similar.

   If the last rule is Eq_1, we consider the following cases:

(a) The label of $\top^*$ is not in the principal relation (i.e., $x \neq w$ and $x \neq w'$). The original derivation is as follows.

$$\dfrac{\begin{array}{c}\Pi \\ (\epsilon, w \triangleright w); \Gamma[w/w']; x : \top^* \vdash \Delta[w/w']\end{array}}{(\epsilon, w' \triangleright w); \Gamma; x : \top^* \vdash \Delta}\ Eq_1$$

By the induction hypothesis, we have the following derivation:

$$\begin{array}{c}\Pi' \\ (\epsilon, w \triangleright w); \Gamma[w/w'][\epsilon/x] \vdash \Delta[w/w'][\epsilon/x]\end{array}$$

Note that since x, w, w' are all different, the end sequent is equal to the following:

$$(\epsilon, w \triangleright w); \Gamma[\epsilon/x][w/w'] \vdash \Delta[\epsilon/x][w/w']$$

from which we can use the rule Eq_1 and derive $(\epsilon, w' \triangleright w); \Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$.

(b) $x = w$. The original derivation is as follows.

$$\dfrac{\begin{array}{c}\Pi \\ (\epsilon, x \triangleright x); \Gamma[x/w']; x : \top^* \vdash \Delta[x/w']\end{array}}{(\epsilon, w' \triangleright x); \Gamma; x : \top^* \vdash \Delta}\ Eq_1$$

By the substitution lemma, replacing every x by ϵ in the premise of the last rule, we get the following derivation:

$$\begin{array}{c}\Pi' \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[x/w'][\epsilon/x]; \epsilon : \top^* \vdash \Delta[x/w'][\epsilon/x]\end{array}$$

The end sequent is equal to:

$$(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/x][\epsilon/w']; \epsilon : \top^* \vdash \Delta[\epsilon/x][\epsilon/w']$$

By Lemma A.3.3, $\epsilon : \top^*$ in the antecedent can be omitted. Apply the Eq_1 rule on this sequent without $\epsilon : \top^*$, we finally get $(\epsilon, w' \triangleright \epsilon); \Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$.

(c) $x = w'$. The original derivation is as follows.

$$\dfrac{\begin{array}{c}\Pi \\ (\epsilon, w \triangleright w); \Gamma[w/x]; w : \top^* \vdash \Delta[w/x]\end{array}}{(\epsilon, x \triangleright w); \Gamma; x : \top^* \vdash \Delta}\ Eq_1$$

By the induction hypothesis, we have the following derivation:

$$\begin{array}{c}\Pi' \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[w/x][\epsilon/w] \vdash \Delta[w/x][\epsilon/w]\end{array}$$

Now the end sequent is equal to:

$$(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/x][\epsilon/w] \vdash \Delta[\epsilon/x][\epsilon/w]$$

By using the rule Eq_2 on this sequent, we derive $(\epsilon, \epsilon \triangleright w); \Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$.

The case where the last rule is Eq_2 is similar to the case for Eq_1.

If the last rule is $\top^* L$, then the derivation to the premise of the last rule yields the new derivation.

The invertibility of Eq_1 and Eq_2 follows from the Substitution Lemma, as the reverse versions of these two rules are only about replacing labels. $\square$

A.5 Contraction admissibility of LS_{BBI}

Lemma A.5.1. *For all structures Γ, Δ, and labelled formula $w : A$, the following holds in LS_{BBI}:*

1. *If there is a cut-free derivation Π of $\Gamma; w : A; w : A \vdash \Delta$, then there is a cut-free derivation Π' of $\Gamma; w : A \vdash \Delta$ with $ht(\Pi') \leq ht(\Pi)$.*

2. *If there is a cut-free derivation Π of $\Gamma \vdash w : A; w : A; \Delta$, then there is a cut-free derivation Π' of $\Gamma \vdash w : A; \Delta$ with $ht(\Pi') \leq ht(\Pi)$.*

Proof. By simultaneous induction on the height of derivations for the left and right contraction. Let $n = ht(\Pi)$.
(Base case) If $n = 0$, the premise is one of the conclusions of id, $\bot L$, $\top R$ and $\top^* R$, then the contracted sequent is also the conclusion of the corresponding rules.
(Inductive case) If $n > 0$, consider the last rule applied to the premise of the contraction.

(i) If the contracted formula is not principal in the last rule, then we can apply the induction hypothesis on the premise(s) of the last rule, then use the rule to get the derivation.

(ii) If the contracted formula is the principal formula of the last rule, we have several cases. For the additive rules the cases are reduced to contraction on smaller formulae, cf. [18].

For $\top^* L$, we have the following derivation:

$$\cfrac{\overset{\Pi}{\Gamma[\epsilon/x]; \epsilon : \top^* \vdash \Delta[\epsilon/x]}}{\Gamma; x : \top^*; x : \top^* \vdash \Delta}\ \top^* L$$

Note that the only case where $\top^*$ is useful on the left hand side is when it is labelled with a world other than ϵ. Since the substitution $[\epsilon/\epsilon]$ does not do anything to the sequent, Π can also be the derivation for $\Gamma[\epsilon/x] \vdash \Delta[\epsilon/x]$, cf. Lemma A.3.3, which leads to $\Gamma; x : \top^* \vdash \Delta$.

For $*R$ and $-\!\!* L$, we can apply the induction hypothesis directly on the premise of the corresponding rule since the rules carry the principal formula into the premise(s).

For $*L$, we have a derivation as follows.

$$\cfrac{\overset{\Pi}{(x, y \triangleright z); \Gamma; z : A * B; x : A; y : B \vdash \Delta}}{\Gamma; z : A * B; z : A * B \vdash \Delta}\ *L$$

Apply the Invertibility Lemma on the premise of $*L$, we have:

$$\overset{\Pi'}{(x, y \triangleright z); (x', y' \triangleright z); \Gamma; x' : A; y' : B; x : A; y : B \vdash \Delta}$$

The Substitution Lemma yields a derivation for $(x, y \triangleright z); (x, y \triangleright z); \Gamma; x : A; y : B; x : A; y : B \vdash \Delta$. Apply the induction hypothesis twice and admissibility of contraction on relational atoms on this sequent, to get a derivation for $(x, y \triangleright z); \Gamma; x : A; y : B \vdash \Delta$. Apply $*L$ on this sequent to get $\Gamma; z : A * B \vdash \Delta$.

The case for $-\!\!* R$ follows similarly. We have a derivation as follows.

$$\cfrac{\overset{\Pi}{(x, y \triangleright z); \Gamma; x : A \vdash z : B; y : A -\!\!* B; \Delta}}{\Gamma \vdash y : A -\!\!* B; y : A -\!\!* B; \Delta}\ -\!\!* R$$

The Invertibility of $-\!\!* R$ in the premise yields:

$$\begin{array}{c}\Pi\\ (x,y\triangleright z);(x',y\triangleright z');\Gamma;x:A;x':A\vdash z:B;z':B;\Delta\end{array}$$

We obtain $(x,y\triangleright z);(x,y\triangleright z);\Gamma;x:A;x:A\vdash z:B;z:B;\Delta$ by the Substitution Lemma. Apply induction hypothesis twice, and the admissibility of contraction on relations on this sequent, to get $(x,y\triangleright z);\Gamma;x:A\vdash z:B\Delta$. Finally, apply $-\!\!* R$, to derive $\Gamma\vdash y:A-\!\!* B;\Delta$ in the nth step. $\square$

A.6 Cut elimination

The proof for Theorem 3.6.

Proof. By induction on the cut ranks of the proof in LS_{BBI}. We show that each application of *cut* can either be eliminated, or be replaced by one or more *cut* rules of smaller cut ranks. The argument for termination is similar to the cut-elimination proof for *G3ip* [18]. We start to eliminate the topmost *cut* first, and repeat this procedure until there is no *cut* in the derivation. We first show that *cut* can be eliminated when the *cut height* is the lowest, i.e., at least one premise is of height 1. Then we show that the *cut height* is reduced in all cases in which the cut formula is not principal in both premises of cut. If the cut formula is principal in both premises, then the *cut* is reduced to one or more *cut*s on smaller formulae or shorter derivations. Since atoms cannot be principal in logical rules, finally we can either reduce all *cut*s to the case where the cut formula is not principal in both premises, or reduce those *cut*s on compound formulae until their *cut height*s are minimal and then eliminate those *cut*s.
(Base case) If at least one premise of the *cut* rule is *id*, $\bot L$, $\top R$, or $\top^* R$, we consider the following cases:

1. The left premise of *cut* is an application of *id*, and the cut formula is not principal, then the derivation is transformed as follows.

$$\dfrac{\dfrac{}{\Gamma;y:B\vdash y:B;x:A;\Delta}\,id \qquad \begin{array}{c}\Pi\\ \Gamma';x:A\vdash\Delta'\end{array}}{\Gamma;\Gamma';y:B\vdash y:B;\Delta;\Delta'}\,cut \rightsquigarrow$$

$$\dfrac{}{\Gamma;\Gamma';y:B\vdash y:B;\Delta;\Delta'}\,id$$

   The same transformation works for $\bot L$, $\top R$, $\top^* R$ in this case.

2. The left premise of *cut* is an application of *id*, and the cut formula is principal, then the derivation is transformed as follows.

$$\dfrac{\dfrac{}{\Gamma;x:A\vdash x:A;\Delta}\,id \qquad \begin{array}{c}\Pi\\ \Gamma';x:A\vdash\Delta'\end{array}}{\Gamma;\Gamma';x:A\vdash\Delta;\Delta'}\,cut \rightsquigarrow$$

$$\dfrac{\begin{array}{c}\Pi\\ \Gamma';x:A\vdash\Delta'\end{array}}{\Gamma;\Gamma';x:A\vdash\Delta;\Delta'}\text{ Lemma 3.2}$$

3. The left premise of *cut* is an application of $\top R$, and the cut formula is principal, then the derivation is transformed as follows.

$$\dfrac{\dfrac{}{\Gamma \vdash x : \top; \Delta}\,\top R \qquad \begin{array}{c}\Pi \\ \Gamma'; x : \top \vdash \Delta'\end{array}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,\mathit{cut} \rightsquigarrow$$

$$\dfrac{\begin{array}{c}\Pi' \\ \Gamma' \vdash \Delta'\end{array}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\ \text{Lemma 3.2}$$

As $x : \top$ cannot be a principal formula in the antecedent, by Lemma A.3.2 there is a derivation Π' of $\Gamma' \vdash \Delta'$.

The same holds for $\top^* R$.

4. The right premise of *cut* is an application of *id*, $\bot L$, $\top R$ or $\top^* R$, and the cut formula is not principal. This case is similar to case 1.

5. The right premise of *cut* is an application of *id*, and the cut formula is principal. This case is similar to case 2.

6. The right premise of *cut* is an application of $\bot L$, and the cut formula is principal. This case is similar to case 3.

(Inductive case) If both premises are not in one of the base cases, we distinguish three cases here: the cut formula is not principal in the left premises; the cut formula is only principal in the left premise; and the cut formula is principal in both premises.

1. The cut formula is not principal in the left premise. Suppose the left premise ends with a rule r.

   (a) If r is $\top^* L$, w.l.o.g. we assume the label of the principal formula is y (which might be equal to x). The original derivation is as follows.

   $$\dfrac{\dfrac{\begin{array}{c}\Pi_1 \\ \Gamma[\epsilon/y] \vdash x : A; \Delta[\epsilon/y]\end{array}}{\Gamma; y : \top^* \vdash x : A; \Delta}\,\top^* L \qquad \begin{array}{c}\Pi_2 \\ \Gamma'; x : A \vdash \Delta'\end{array}}{\Gamma; \Gamma'; y : \top^* \vdash \Delta; \Delta'}\,\mathit{cut}$$

   By the Substitution lemma, there is a derivation Π_2' of $\Gamma'[\epsilon/y]; x : A \vdash \Delta[\epsilon/y]$. Thus we can transform the derivation into the following:

   $$\dfrac{\dfrac{\begin{array}{c}\Pi_1 \\ \Gamma[\epsilon/y] \vdash x : A; \Delta[\epsilon/y]\end{array} \qquad \begin{array}{c}\Pi_2' \\ \Gamma'[\epsilon/y]; x : A \vdash \Delta'[\epsilon/y]\end{array}}{\Gamma[\epsilon/y]; \Gamma'[\epsilon/y] \vdash \Delta[\epsilon/y]; \Delta'[\epsilon/y]}\,\mathit{cut}}{\Gamma; \Gamma'; y : \top^* \vdash \Delta; \Delta'}\,\top^* L$$

   If $x = y$ in the original derivation, then the new derivation cuts on $\epsilon : A$ instead. As substitution is height preserving, the cut height in this case is reduced as well.

   (b) If r is Eq_1, and the label x of the principal formula is not equal to w', the original derivation is as follows.

$$\dfrac{\dfrac{\overset{\Pi_1}{(\epsilon, w \rhd w); \Gamma[w/w'] \vdash x : A; \Delta[w/w']}}{(\epsilon, w' \rhd w); \Gamma \vdash x : A; \Delta}\,Eq_1 \qquad \overset{\Pi_2}{\Gamma'; x : A \vdash \Delta'}}{(\epsilon, w' \rhd w); \Gamma; \Gamma' \vdash \Delta; \Delta'}\,cut$$

This *cut* is reduced in the same way as the $\top^* L$ case, where we get Π_2' from the Substitution Lemma:

$$\dfrac{\dfrac{\overset{\Pi_1}{(\epsilon, w \rhd w); \Gamma[w/w'] \vdash x : A; \Delta[w/w']} \qquad \overset{\Pi_2'}{\Gamma'[w/w']; x : A \vdash \Delta'[w/w']}}{(\epsilon, w \rhd w); \Gamma[w/w']; \Gamma'[w/w'] \vdash \Delta[w/w']; \Delta'[w/w']}\,cut}{(\epsilon, w' \rhd w); \Gamma; \Gamma' \vdash \Delta; \Delta'}\,Eq_1$$

If $x = w'$, then we cut on $w : A$ instead in the reduced version.

(c) If r is Eq_2, the procedure follows similarly as the case for Eq_1 above.

(d) If r is a unary inference except for $\top^* L$, Eq_1, and Eq_2, then the original derivation is as follows.

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma_1 \vdash x : A; \Delta_1}}{\Gamma \vdash x : A; \Delta}\,r \qquad \overset{\Pi_2}{\Gamma'; x : A \vdash \Delta'}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,cut$$

Then we can delay the application of *cut* as follows.

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma_1 \vdash x : A; \Delta_1} \qquad \overset{\Pi_2}{\Gamma'; x : A \vdash \Delta'}}{\Gamma_1; \Gamma' \vdash \Delta_1; \Delta'}\,cut}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,r$$

Note that as all our rules except $\top^* L$, Eq_1, and Eq_2 do not modify side structures, Γ' and Δ' in the premise of r are not changed. The cut rank of the original *cut* is $(|x : A|, |\Pi_1| + 1 + |\Pi_2|)$, whereas the cut rank of the new *cut* is $(|x : A|, |\Pi_1| + |\Pi_2|)$, so the *cut height* reduces.

(e) If r is a binary inference, we can transform the derivation similarly.

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma_1 \vdash x : A; \Delta_1} \qquad \overset{\Pi_2}{\Gamma_2 \vdash x : A; \Delta_2}}{\Gamma \vdash x : A; \Delta}\,r \qquad \overset{\Pi_3}{\Gamma'; x : A \vdash \Delta'}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,cut \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\overset{\Pi_1}{\Gamma_1 \vdash x : A; \Delta_1} \qquad \overset{\Pi_3}{\Gamma'; x : A \vdash \Delta'}}{\Gamma_1; \Gamma' \vdash \Delta_1; \Delta'}\,cut \qquad \dfrac{\overset{\Pi_2}{\Gamma_2 \vdash x : A; \Delta_2} \qquad \overset{\Pi_3}{\Gamma'; x : A \vdash \Delta'}}{\Gamma_2; \Gamma' \vdash \Delta_2; \Delta'}\,cut}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\,r$$

The cut rank of the original *cut* is $(|x : A|, max(|\Pi_1|, |\Pi_2|) + 1 + |\Pi_3|)$, and that of the new two *cut*s are $(|x : A|, |\Pi_1| + |\Pi_3|)$ and $(|x : A|, |\Pi_2| + |\Pi_3|)$ respectively. Thus the cut heights are reduced.

2. The cut formula is only principal in the left premise. We only consider the last rule in the right branch. The proof of this case is symmetric to those in Case 1.

3. The cut formula is principal in both premises. We do a case analysis on the main connective of the cut formula. If the main connective is additive, then there is no need to substitute any labels.

   For $\wedge$,

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \Gamma \vdash x : A; \Delta\end{array} \qquad \begin{array}{c}\Pi_2\\ \Gamma \vdash x : B; \Delta\end{array}}{\Gamma \vdash x : A \wedge B; \Delta}\wedge R \qquad \dfrac{\begin{array}{c}\Pi_3\\ \Gamma'; x : A; x : B \vdash \Delta'\end{array}}{\Gamma'; x : A \wedge B \vdash \Delta'}\wedge L}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\mathit{cut} \rightsquigarrow$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \Gamma \vdash x : A; \Delta\end{array} \qquad \dfrac{\begin{array}{c}\Pi_2\\ \Gamma \vdash x : B; \Delta\end{array} \qquad \begin{array}{c}\Pi_3\\ \Gamma'; x : A; x : B \vdash \Delta'\end{array}}{\Gamma; \Gamma'; x : A \vdash \Delta; \Delta'}\mathit{cut}}{\Gamma; \Gamma; \Gamma' \vdash \Delta; \Delta; \Delta'}\mathit{cut}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\text{Lemma 3.5}$$

   For $\to$,

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ \Gamma'; x : A \vdash x : B; \Delta'\end{array}}{\Gamma' \vdash x : A \to B; \Delta'}\to R \qquad \dfrac{\begin{array}{c}\Pi_2\\ \Gamma \vdash x : A; \Delta\end{array} \qquad \begin{array}{c}\Pi_3\\ \Gamma; x : B \vdash \Delta\end{array}}{\Gamma; x : A \to B \vdash \Delta}\to L}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\mathit{cut} \rightsquigarrow$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_2\\ \Gamma \vdash x : A; \Delta\end{array} \qquad \dfrac{\begin{array}{c}\Pi_1\\ \Gamma'; x : A \vdash x : B; \Delta'\end{array} \qquad \begin{array}{c}\Pi_3\\ \Gamma; x : B \vdash \Delta\end{array}}{\Gamma; \Gamma'; x : A \vdash \Delta; \Delta'}\mathit{cut}}{\Gamma; \Gamma; \Gamma' \vdash \Delta; \Delta; \Delta'}\mathit{cut}}{\Gamma; \Gamma' \vdash \Delta; \Delta'}\text{Lemma 3.5}$$

   For both $\wedge$ and $\to$, *cut* is reduced to applications on smaller formulae, therefore the cut rank of the *cut* reduces.

   There is an asymmetry in the rules for $\top^*$. That is, the left rule for $\top^*$ requires that the label w of $\top^*$ cannot be ϵ, whereas the right rule for $\top^*$ restricts the label of $\top^*$ to be ϵ only. As a consequence, when the cut formula is $\top^*$, it cannot be the principal formula of both premises at the same time. Therefore the cases for $\top^*$ are handled in the proof above.

   When the main connective of the cut formula is $*$ or $-\!\!*$, the case is more complicated. For $*$, we have the following two derivations as the premises of the *cut* rule:

$$\dfrac{\begin{array}{c}\Pi_1\\ (x, y \rhd z); \Gamma \vdash x : A; z : A * B; \Delta\end{array} \qquad \begin{array}{c}\Pi_2\\ (x, y \rhd z); \Gamma \vdash y : B; z : A * B; \Delta\end{array}}{(x, y \rhd z); \Gamma \vdash z : A * B; \Delta}{\scriptstyle *R}$$

   and

$$\dfrac{\overset{\Pi_3}{(x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'}}{\Gamma'; z : A * B \vdash \Delta'}\; {*L}$$

And the *cut* rule gives the end sequent $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$. The cut rank of this *cut* is $(|A * B|, max(|\Pi_1|, |\Pi_2|) + 1 + |\Pi_3| + 1)$.

We use several *cut*s with smaller ranks to derive $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$ as follows.

Firstly,

$$\dfrac{\overset{\Pi_1}{(x, y \triangleright z); \Gamma \vdash x : A; z : A * B; \Delta} \qquad \dfrac{\overset{\Pi_3}{(x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'}}{\Gamma'; z : A * B \vdash \Delta'}\; {*L}}{(x, y \triangleright z); \Gamma; \Gamma' \vdash x : A; \Delta; \Delta'}\; cut$$

The cut rank of this *cut* is $(|A * B|, |\Pi_1| + |\Pi_3| + 1))$, thus is less than the original *cut*.

The second *cut* works similarly.

$$\dfrac{\overset{\Pi_2}{(x, y \triangleright z); \Gamma \vdash y : B; z : A * B; \Delta} \qquad \dfrac{\overset{\Pi_3}{(x', y' \triangleright z); \Gamma'; x' : A; y' : B \vdash \Delta'}}{\Gamma'; z : A * B \vdash \Delta'}\; {*L}}{(x, y \triangleright z); \Gamma; \Gamma' \vdash y : B; \Delta; \Delta'}\; cut$$

The third *cut* works on a smaller formula.

$$\dfrac{(x, y \triangleright z); \Gamma; \Gamma' \vdash x : A; \Delta; \Delta' \qquad \overset{\Pi_3'}{(x, y \triangleright z); \Gamma'; x : A; y : B \vdash \Delta'}}{(x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma'; \Gamma'; y : B \vdash \Delta; \Delta'; \Delta'}\; cut$$

The cut formula is $x : A$, thus the cut rank of this *cut* is less regardless of the height of the derivations.

Note that in the Π_3 branch, the $*L$ rule requires that the relation $(x', y' \triangleright z)$ is newly created, so x' and y' cannot be ϵ and they cannot be in Γ' or Δ'. Therefore we are allowed to use the substitution lemma to get a derivation Π_3' of $(x, y \triangleright z); \Gamma'; x : A; y : B \vdash \Delta'$ by just substituting x' for x and y' for y.

Finally we cut on another smaller formula $y : B$.

$$\dfrac{(x, y \triangleright z); \Gamma; \Gamma' \vdash y : B; \Delta; \Delta' \qquad (x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma'; \Gamma'; y : B \vdash \Delta; \Delta'; \Delta'}{(x, y \triangleright z); (x, y \triangleright z); (x, y \triangleright z); \Gamma; \Gamma; \Gamma'; \Gamma'; \Gamma' \vdash \Delta; \Delta; \Delta'; \Delta'; \Delta'}\; cut$$

The cut rank of this *cut* is less than the original *cut*. We then apply the admissibility of contraction to derive $(x, y \triangleright z); \Gamma; \Gamma' \vdash \Delta; \Delta'$.

The case for $-\!\!*$ is similar. The two premises in the original *cut* are as follows.

$$\dfrac{\overset{\Pi_1}{(x', y \triangleright z'); \Gamma'; x' : A \vdash z' : B; \Delta'}}{\Gamma' \vdash y : A -\!\!* B; \Delta'}\; {-\!\!*\, R}$$

and

$$\dfrac{\begin{array}{c}\Pi_2\\ (x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B\vdash x:A;\Delta\end{array}\qquad\begin{array}{c}\Pi_3\\ (x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B;z:B\vdash\Delta\end{array}}{(x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B\vdash\Delta}\ {\mathrel{-\!\!*}}L$$

And the *cut* rule yields the end sequent $(x,y\triangleright z);\Gamma;\Gamma'\vdash\Delta;\Delta'$. We use two cuts on the same formula, but with smaller derivation height.

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ (x',y\triangleright z');\Gamma';x':A\vdash z':B;\Delta'\end{array}}{\Gamma'\vdash y:A\mathrel{-\!\!*}B;\Delta'}\ {\mathrel{-\!\!*}}R\qquad\begin{array}{c}\Pi_2\\ (x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B\vdash x:A;\Delta\end{array}}{(x,y\triangleright z);\Gamma;\Gamma'\vdash x:A;\Delta;\Delta'}\ cut$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_1\\ (x',y\triangleright z');\Gamma';x':A\vdash z':B;\Delta'\end{array}}{\Gamma'\vdash y:A\mathrel{-\!\!*}B;\Delta'}\ {\mathrel{-\!\!*}}R\qquad\begin{array}{c}\Pi_3\\ (x,y\triangleright z);\Gamma;y:A\mathrel{-\!\!*}B;z:B\vdash\Delta\end{array}}{(x,y\triangleright z);\Gamma;\Gamma';z:B\vdash\Delta;\Delta'}\ cut$$

Then we cut on a smaller formula $x:A$.

$$\dfrac{(x,y\triangleright z);\Gamma;\Gamma'\vdash x:A;\Delta;\Delta'\qquad\begin{array}{c}\Pi_1'\\ (x,y\triangleright z);\Gamma';x:A\vdash z:B;\Delta'\end{array}}{(x,y\triangleright z);(x,y\triangleright z);\Gamma;\Gamma';\Gamma'\vdash z:B;\Delta;\Delta';\Delta'}\ cut$$

Again, in the original derivation, x' and z' are fresh in the premise of $\mathrel{-\!\!*}R$ rule, thus by the Substitution Lemma we can have a derivation Π_1' of the sequent $(x,y\triangleright z);\Gamma';x:A\vdash z:B;\Delta'$, with x' substituted to x and z' substituted to z.

Then we cut on $z:B$.

$$\dfrac{(x,y\triangleright z);(x,y\triangleright z);\Gamma;\Gamma';\Gamma'\vdash z:B;\Delta;\Delta';\Delta'\qquad(x,y\triangleright z);\Gamma;\Gamma';z:B\vdash\Delta;\Delta'}{(x,y\triangleright z);(x,y\triangleright z);(x,y\triangleright z);\Gamma;\Gamma;\Gamma';\Gamma';\Gamma'\vdash\Delta;\Delta;\Delta';\Delta';\Delta'}\ cut$$

In the end we use the theorem of admissibility of contraction to obtain the required sequent $(x,y\triangleright z);\Gamma;\Gamma'\vdash\Delta;\Delta'$.

□

A.7 Permutation of structural rules in LS_{BBI}

Proof for Lemma 4.1.

Proof. To prove this lemma, we need to show that if a derivation involves the structural rules, we can always apply them exactly before $*R$ and $\mathrel{-\!\!*}L$, or before zero-premise rules. We show this by an induction on the height of the derivation. Since we do not permute structural rules through zero-premise rules, the proof in the base case and the inductive step are essentially the same. Here we give some examples of the permutations. Assuming the lemma holds up to any derivation of height $n-1$, consider a derivation of height n.

1. Permute the application of Eq_1 or Eq_2 through non-zero-premise logical rules except for $*R$ and $-\!\!* \, L$. Here we give some examples, the rest are similar.

   (a) Permute Eq_2 through additive logical rules is trivial, this is exemplified by $\wedge L$, assuming the label of the principal formula is modified by the Eq_2 application. The original derivation is as follows.

$$\dfrac{\dfrac{\begin{matrix}\Pi \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w]; \epsilon : A; \epsilon : B \vdash \Delta[\epsilon/w]\end{matrix}}{(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w]; \epsilon : A \wedge B \vdash \Delta[\epsilon/w]} \wedge L}{(\epsilon, \epsilon \triangleright w); \Gamma; w : A \wedge B \vdash \Delta} Eq_2$$

   The derivation is changed to the following:

$$\dfrac{\dfrac{\begin{matrix}\Pi \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w]; \epsilon : A; \epsilon : B \vdash \Delta[\epsilon/w]\end{matrix}}{(\epsilon, \epsilon \triangleright w); \Gamma; w : A; w : B \vdash \Delta} Eq_2}{(\epsilon, \epsilon \triangleright w); \Gamma; w : A \wedge B \vdash \Delta} \wedge L$$

   (b) Permute Eq_1 through $\top^* L$, assuming the label of principal formula is w. The derivation is as follows.

$$\dfrac{\dfrac{\begin{matrix}\Pi \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[w/w'][\epsilon/w] \vdash \Delta[w/w'][\epsilon/w]\end{matrix}}{(\epsilon, w \triangleright w); \Gamma[w/w']; w : \top^* \vdash \Delta[w/w']} \top^* L}{(\epsilon, w' \triangleright w); \Gamma; w' : \top^* \vdash \Delta} Eq_1$$

   We modify the derivation as follows.

$$\dfrac{\dfrac{\begin{matrix}\Pi \\ (\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w'][\epsilon/w] \vdash \Delta[\epsilon/w'][\epsilon/w]\end{matrix}}{(\epsilon, \epsilon \triangleright w); \Gamma[\epsilon/w'] \vdash \Delta[\epsilon/w']} Eq_2}{(\epsilon, w' \triangleright w); \Gamma; w' : \top^* \vdash \Delta} \top^* L$$

   Notice that the premises of the two derivations below Π are exactly the same. The application of Eq_1 in the original derivation is changed to an application of Eq_2 in the modified derivation. However, this does not break the proof, as the induction hypothesis ensures that either of them can be permuted upwards.

   Also, the label of principal formula in the rule $\top^* L$ cannot be the one that is replaced in the rule Eq_2 below it, this is the reason we do not exemplify this situation using Eq_2.

   (c) Permute Eq_2 through $*L$, assuming the label of principal formula is z, and it is modified by the Eq_2 application.

$$\dfrac{\dfrac{\begin{matrix}\Pi \\ (x, y \triangleright \epsilon); (\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/z]; x : A; y : B \vdash \Delta[\epsilon/z]\end{matrix}}{(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/z]; \epsilon : A * B \vdash \Delta[\epsilon/z]} *L}{(\epsilon, \epsilon \triangleright z); \Gamma; z : A * B \vdash \Delta} Eq_2$$

   Since x and y are fresh labels, they will not be affected by Eq_2. Thus the derivation can be changed to the following:

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\(x,y\triangleright\epsilon);(\epsilon,\epsilon\triangleright\epsilon);\Gamma[\epsilon/z];x:A;y:B\vdash\Delta[\epsilon/z]\end{array}}{(x,y\triangleright z);(\epsilon,\epsilon\triangleright z);\Gamma;x:A;y:B\vdash\Delta}\ Eq_2}{(\epsilon,\epsilon\triangleright y);\Gamma;z:A*B\vdash\Delta}\ *L$$

Since Eq_1 and Eq_2 only globally replaces labels, their action can be safely delayed through all the rules other than $*R$ and $-\!\!* \ L$. The applications of these two rules after the last $*R$ or $*L$ will be delayed until the zero-premise rule is necessary.

2. Permute the applications of E, U, A, and A_C through non-zero premise logical rules other than $*R$ and $-\!\!* \ L$. Again, we give some examples, the rest are similar.

   (a) Permute E through $\top^* L$, assuming the label of the principal formula is y. The original derivation runs as follows.

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\(\epsilon,x\triangleright z);(x,\epsilon\triangleright z);\Gamma[\epsilon/y]\vdash\Delta[\epsilon/y]\end{array}}{(y,x\triangleright z);(x,y\triangleright z);\Gamma;y:\top^*\vdash\Delta}\ \top^*L}{(x,y\triangleright z);\Gamma;y:\top^*\vdash\Delta}\ E$$

   The new derivation is as follows.

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\(\epsilon,x\triangleright z);(x,\epsilon\triangleright z);\Gamma[\epsilon/y]\vdash\Delta[\epsilon/y]\end{array}}{(x,\epsilon\triangleright z);\Gamma[\epsilon/y]\vdash\Delta[\epsilon/y]}\ E}{(x,y\triangleright z);\Gamma;y:\top^*\vdash\Delta}\ \top^*L$$

   This shows that if the logical rule only does substitution, delaying the application of structural rules makes no difference.

   (b) Permute U through $*L$, assuming the label of the principal formula is z. The original derivation is as follows.

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\(x,y\triangleright z);(z,\epsilon\triangleright z);\Gamma;x:A;y:B\vdash\Delta\end{array}}{(z,\epsilon\triangleright z);\Gamma;z:A*B\vdash\Delta}\ *L}{\Gamma;z:A*B\vdash\Delta}\ U$$

   The new derivation is as follows.

$$\cfrac{\cfrac{\begin{array}{c}\Pi\\(z,\epsilon\triangleright z);(x,y\triangleright z);\Gamma;x:A;y:B\vdash\Delta\end{array}}{(x,y\triangleright z);\Gamma;z:A*B\vdash\Delta}\ U}{\Gamma;z:A*B\vdash\Delta}\ *L$$

   Since the labels x and y are all fresh labels, it is safe to change the order to rule applications as above.

   Additive logical rules are totally independent on the relational atoms, so those cases are similar as the one shown above, except that those rules do not add relational atoms to the sequent.

□

A.8 Soundness of LS^e_{BBI}

Theorem A.8.1. *If there is a derivation* Π *for a sequent* $\Gamma \vdash \Delta$ *in* LS^e_{BBI}*, then there is a derivation* Π' *for the same sequent in* LS_{BBI}*.*

Proof. By induction on the height n of Π.

1. Base case: $n = 1$. In this case the only rule must be a zero-premise rule. If the rule is $\bot L$ or $\top R$, then we can use the same rule in LS_{BBI}, since they are the same. Otherwise, suppose the rule is *id*, then Π reads as follows.

$$\frac{\mathcal{G} \vdash_E (w_1 = w_2)}{\Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle id}$$

Since $\mathcal{G} \vdash_E (w_1 = w_2)$ is true, there is a sequence σ of Eq_1, Eq_2 applications such that $\mathcal{S}(\mathcal{G}, \sigma)$ is defined and $w_1\theta = w_2\theta$, where $\theta = subst(\sigma)$. Therefore we can construct Π' are follows.

$$\begin{array}{c}\dfrac{}{\Gamma\theta; w_1\theta : P \vdash w_2\theta : P; \Delta\theta}\ {\scriptstyle id}\\ \vdots\sigma \\ \Gamma; w_1 : P \vdash w_2 : P; \Delta\end{array}$$

If the rule is $\top^* R$, Π is:

$$\frac{\mathcal{G} \vdash_E (w = \epsilon)}{\Gamma \vdash w : \top^*; \Delta}\ {\scriptstyle \top^* R}$$

We construct Π' similarly, as $w\theta = \epsilon$ after the application of σ.

$$\begin{array}{c}\dfrac{}{\Gamma\theta \vdash w\theta : \top^*; \Delta\theta}\ {\scriptstyle \top^* R}\\ \vdots\sigma \\ \Gamma \vdash w : \top^*; \Delta\end{array}$$

2. Inductive cases: suppose every sequent that is derivable in LS^e_{BBI} with height less than n is also derivable in LS_{BBI}, consider a LS^e_{BBI} derivation of height n. We do a case analysis on the bottom rule in the derivation.

   (a) If the rule is $\wedge L$, $\wedge R$, $\rightarrow L$, $\rightarrow R$, $*L$, $*R$, E or U, we can use the same rule in LS_{BBI}, since nothing is changed.

   (b) If the rule is $\top^* L$, then Π must be the following:

$$\frac{\begin{array}{c}\Pi_1\\ (\epsilon, w \triangleright \epsilon); \Gamma \vdash \Delta\end{array}}{\Gamma; w : \top^* \vdash \Delta}\ {\scriptstyle \top^* L}$$

   By the induction hypothesis, $(\epsilon, w \triangleright \epsilon); \Gamma \vdash \Delta$ is derivable in LS_{BBI}. Applying Lemma 3.1 (substitution for labels in LS_{BBI}) with $[\epsilon/w]$, we obtain $(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]$. Thus we construct Π' as follows.

$$\cfrac{\cfrac{\begin{array}{c}\Pi'_1\\(\epsilon,\epsilon\triangleright\epsilon);\Gamma[\epsilon/w]\vdash\Delta[\epsilon/w]\end{array}}{(w,\epsilon\triangleright w);\Gamma;w:\top^*\vdash\Delta}\;{\scriptstyle \top^*L}}{\Gamma;w:\top^*\vdash\Delta}\;{\scriptstyle U}$$

(c) If the rule is $*R$, Π runs as follows.

$$\cfrac{\begin{array}{c}\Pi_1\\(x,y\triangleright z');\Gamma\vdash x:A;z:A*B;\Delta\end{array}\qquad\begin{array}{c}\Pi_2\\(x,y\triangleright z');\Gamma\vdash y:B;z:A*B;\Delta\end{array}}{(x,y\triangleright z');\Gamma\vdash z:A*B;\Delta}\;{\scriptstyle *R}$$

The condition on the $*R$ rule is $\mathcal{G}\vdash_E (z=z')$. Let σ be the sequence of Eq_1, Eq_2 applications such that $\mathcal{S}(\mathcal{G},\sigma)$ is defined and, $z\theta = z'\theta$ holds, where $\theta = subst(\sigma)$. Also, applying the induction hypothesis on Π_1 and Π_2, we obtain the LS_{BBI} derivations for each branch respectively. Then with the help of the Substitution lemma, we get two derivations as follows. Note that we use dashed lines when applying the Substitution lemmas.

$$\cfrac{\begin{array}{c}\Pi'_1\\(x,y\triangleright z');\Gamma\vdash x:A;z:A*B;\Delta\end{array}}{(x\theta,y\theta\triangleright z'\theta);\Gamma\theta\vdash x\theta:A;z\theta:A*B;\Delta\theta}\;{\scriptstyle \text{Lemma 3.1}}$$

and

$$\cfrac{\begin{array}{c}\Pi'_2\\(x,y\triangleright z');\Gamma\vdash y:B;z:A*B;\Delta\end{array}}{(x\theta,y\theta\triangleright z'\theta);\Gamma\theta\vdash y\theta:B;z\theta:A*B;\Delta\theta}\;{\scriptstyle \text{Lemma 3.1}}$$

Then we can apply $*R$ and obtain $(x\theta,y\theta\triangleright z'\theta);\Gamma\vdash z\theta:A*B;\Delta\theta$. Then by applying σ we obtain the end sequent as follows.

$$\begin{array}{c}(x\theta,y\theta\triangleright z'\theta);\Gamma\vdash z\theta:A*B;\Delta\theta\\ \vdots\sigma\\ (x,y\triangleright z');\Gamma\vdash z:A*B;\Delta\end{array}$$

The case for $-\!\!* \; L$ is treated similarly.

(d) If the rule is A, the treatment for the equality entailment is the same. Π is in the following form:

$$\cfrac{\begin{array}{c}\Pi_1\\(u,w\triangleright z);(y,v\triangleright w);(x,y\triangleright z);(u,v\triangleright x');\Gamma\vdash\Delta\end{array}\qquad\mathcal{G}\vdash_E (x=x')}{(x,y\triangleright z);(u,v\triangleright x');\Gamma\vdash\Delta}\;{\scriptstyle A}$$

Let $\mathcal{S}(\mathcal{G},\sigma)$ yield $x\theta = x'\theta$, where $\theta = subst(\sigma)$, we obtain Π' as follows.

$$\begin{array}{c}\cfrac{\cfrac{\begin{array}{c}\Pi'_1\\(u,w\triangleright z);(y,v\triangleright w);(x,y\triangleright z);(u,v\triangleright x');\Gamma\vdash\Delta\end{array}}{(u\theta,w\theta\triangleright z\theta);(y\theta,v\theta\triangleright w\theta);(x\theta,y\theta\triangleright z\theta);(u\theta,v\theta\triangleright x'\theta);\Gamma\theta\vdash\Delta\theta}\;{\scriptstyle \text{Lemma 3.1}}}{(x\theta,y\theta\triangleright z\theta);(u\theta,v\theta\triangleright x'\theta);\Gamma\theta\vdash\Delta\theta}\;{\scriptstyle A}\\ \vdots\sigma\\ (x,y\triangleright z);(u,v\triangleright x');\Gamma\vdash\Delta\end{array}$$

The case for A_C is similar.

$\square$

A.9 Completeness of LS^e_{BBI}

To prove the completeness of LS^e_{BBI}, firstly we add Eq_1 and Eq_2 in LS^e_{BBI} and show that the resultant system has the same power as LS_{BBI}. Then we prove the admissibility of Eq_1 and Eq_2 in LS^e_{BBI}.

Lemma A.9.1. *If a sequent* $\Gamma \vdash \Delta$ *is derivable in* LS_{BBI}*, then it is derivable in* $LS^e_{BBI}+Eq_1+Eq_2$.

Proof. By induction on the height of the LS_{BBI} derivation. Since with Eq_1 and Eq_2, most of other rules become identical, the only non-trivial case is $\top^* L$.

In LS_{BBI}, the derivation runs as follows.

$$\cfrac{\begin{array}{c}\Pi \\ \Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]\end{array}}{\Gamma; w : \top^* \vdash \Delta}\ {\scriptstyle \top^* L}$$

By the induction hypothesis, there is a derivation for $\Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]$ in $LS^e_{BBI} + Eq_1 + Eq_2$. Therefore we construct the derivation as follows.

$$\cfrac{\cfrac{\cfrac{\begin{array}{c}\Pi' \\ \Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]\end{array}}{(\epsilon, \epsilon \triangleright \epsilon); \Gamma[\epsilon/w] \vdash \Delta[\epsilon/w]}\ {\scriptstyle \text{Lemma 3.2}}}{(\epsilon, w \triangleright \epsilon); \Gamma \vdash \Delta}\ {\scriptstyle Eq_1}}{\Gamma; w : \top^* \vdash \Delta}\ {\scriptstyle \top^* L}$$

$\square$

Lemma A.9.2. *If* $\mathcal{G}[x/y]; (\epsilon, x \triangleright x) \vdash_E (w_1[x/y] = w_2[x/y])$ *then* $\mathcal{G}; (\epsilon, y \triangleright x) \vdash_E (w_1 = w_2)$.

Proof. Let $\mathcal{G}' = \mathcal{G}; (\epsilon, y \triangleright x)$ and $\mathcal{S}(\mathcal{G}'[x/y], \sigma)$ yield $(w_1[x/y]\theta = w_2[x/y]\theta)$, we show that $\mathcal{G}' \vdash_E (x = y)$ by following:

$$\cfrac{\begin{array}{c}\mathcal{G}'[x/y]\theta \vdash_E (w_1[x/y]\theta = w_2[x/y]\theta) \\ \vdots \sigma \\ \mathcal{G}'[x/y] \vdash_E (w_1[x/y] = w_2[x/y])\end{array}}{\mathcal{G}; (\epsilon, y \triangleright x) \vdash_E (x = y)}\ {\scriptstyle Eq_1}$$

$\square$

Now we show that Eq_1 is admissible in LS^e_{BBI}.

Lemma A.9.3. *If* $(\epsilon, x \triangleright x); \Gamma[x/y] \vdash \Delta[x/y]$ *is derivable in* LS^e_{BBI}*, then* $(\epsilon, y \triangleright x); \Gamma \vdash \Delta$ *is derivable in* LS^e_{BBI}.

Proof. We show that Eq_1 can always permute up through all other rules, and eventually disappear when it hits the zero-premise rule. Since Lemma 4.1 is sufficient to show the permutations through nagative rules, here we particularly show the cases for positive rules.

1. First let us show the cases for the zero-premise rules. $\bot L$ and $\top R$ are trivial, as they are applicable for an arbitrary label. The permutation for id runs as follows, where $\mathcal{G}$ is the set of relational atoms in $(\epsilon, y \triangleright x); \Gamma$.

$$\cfrac{\cfrac{\mathcal{G}[x/y] \vdash_E (w_1[x/y] = w_2[x/y])}{(\epsilon, x \triangleright x); \Gamma[x/y]; w_1[x/y] : P \vdash w_2[x/y] : P; \Delta}\ {\scriptstyle id}}{(\epsilon, y \triangleright x); \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle Eq_1}$$

By Lemma A.9.2, if $\mathcal{G}[x/y] \vdash_E (w_1[x/y] = w_2[x/y])$ then $\mathcal{G} \vdash_E (w_1 = w_2)$ (note that this is because $(\epsilon, y \triangleright x) \in \mathcal{G}$). Therefore we can apply id directly on the bottom sequent, and eliminate the Eq_1 application.

The case for $\top^* R$ is treated similarly. As we have shown, structural rules can permute through $\top^* L$, $\wedge L$, $\wedge R$, $\to L$, $\to R$, $* L$ and $-\!\!* R$, so these cases are left out here.

2. Permute Eq_1 through E, assuming the label being replaced is y. The original derivation is as follows.

$$\cfrac{\cfrac{\cfrac{\Pi}{(w, x, \triangleright z); (x, w \triangleright z); (\epsilon, w \triangleright w); \Gamma[w/y] \vdash \Delta[w/y]}}{(x, w \triangleright z); (\epsilon, w \triangleright w); \Gamma[w/y] \vdash \Delta[w/y]}\ {\scriptstyle E}}{(x, y \triangleright z); (\epsilon, y \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle Eq_1}$$

The permuted derivation is as follows.

$$\cfrac{\cfrac{\cfrac{\Pi}{(w, x, \triangleright z); (x, w \triangleright z); (\epsilon, w \triangleright w); \Gamma[w/y] \vdash \Delta[w/y]}}{(y, x \triangleright z); (x, y \triangleright z); (\epsilon, y \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle Eq_1}}{(x, y \triangleright z); (\epsilon, y \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle E}$$

3. Premute Eq_1 through U, assuming the replaced label is x. Then the derivation runs as follows.

$$\cfrac{\cfrac{\cfrac{\Pi}{(w, \epsilon \triangleright w); (\epsilon, w \triangleright w); \Gamma[w/x] \vdash \Delta[w/x]}}{(\epsilon, w \triangleright w); \Gamma[w/x] \vdash \Delta[w/x]}\ {\scriptstyle U}}{(\epsilon, x \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle Eq_1}$$

We modify the derivation as follows.

$$\cfrac{\cfrac{\cfrac{\Pi}{(w, \epsilon \triangleright w); (\epsilon, w \triangleright w); \Gamma[w/x] \vdash \Delta[w/x]}}{(x, \epsilon \triangleright x); (\epsilon, x \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle Eq_1}}{(\epsilon, x \triangleright w); \Gamma \vdash \Delta}\ {\scriptstyle U}$$

Note that we can also generate $(w, \epsilon \triangleright w)$ directly using the U rule, but the effect is the same.

4. Permute Eq_1 through $*R$. Suppose the principal relational atom of Eq_1 is not the same as the one used in $*R$, let $\mathcal{G}$ be the set of relational atoms in $(\epsilon, w \triangleright w')(x, y \triangleright z'); \Gamma$, the derivation runs as follows. Here we write $(\Gamma \vdash \Delta)[x/y]$ to mean that replace every y by x in the entire sequent. The equality entailment is $\mathcal{G}[w'/w] \vdash_E (z[w'/w] = z'[w'/w])$ (to save space, we do not write the constraint in the derivation).

$$\dfrac{\dfrac{}{((\epsilon, w' \triangleright w')(x, y \triangleright z'); \Gamma \vdash z : A * B; \Delta)[w'/w]}\,{\scriptstyle *R}}{(\epsilon, w \triangleright w')(x, y \triangleright z'); \Gamma \vdash z : A * B; \Delta}\,{\scriptstyle Eq_1}$$

The two premises of the $*R$ rule application are listed below.

$$((\epsilon, w' \triangleright w'); (; x, y \triangleright z'); \Gamma \vdash x : A; z : A * B; \Delta)[w'/w]$$
$$((\epsilon, w' \triangleright w'); (x, y \triangleright z'); \Gamma \vdash y : B; z : A * B; \Delta)[w'/w]$$

By Lemma A.9.2, since $\mathcal{G}[w'/w] \vdash_E (z[w'/w] = z'[w'/w])$, and $(\epsilon, w \triangleright w') \in \mathcal{G}$, $\mathcal{G} \vdash_E (z = z')$ holds. Therefore we have the following two derivations:

$$\dfrac{((\epsilon, w' \triangleright w'); (; x, y \triangleright z'); \Gamma \vdash x : A; z : A * B; \Delta)[w'/w]}{(\epsilon, w \triangleright w'); (; x, y \triangleright z'); \Gamma \vdash x : A; z : A * B; \Delta}\,{\scriptstyle Eq_1}$$

and

$$\dfrac{((\epsilon, w' \triangleright w'); (x, y \triangleright z'); \Gamma \vdash y : B; z : A * B; \Delta)[w'/w]}{(\epsilon, w \triangleright w'); (x, y \triangleright z'); \Gamma \vdash y : B; z : A * B; \Delta}\,{\scriptstyle Eq_1}$$

then we use the $*R$ rule, where the equality entailment is $\mathcal{G} \vdash_E (z = z')$, to obtain the end sequent $(\epsilon, w \triangleright w')(x, y \triangleright z'); \Gamma \vdash z : A * B; \Delta$.

If the principal relational atom is used in the $*R$ rule, the permutation is analogous. The permutation through $-\!\!* \ L$ is similar.

5. Permutation through A. We show the case where the principal relational atom in Eq_1 is not in A, the other cases are similar. The original derivation is as follows.

$$\dfrac{\dfrac{((\epsilon, w \triangleright w); (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta)[w/w']}{((\epsilon, w \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta)[w/w']}\,{\scriptstyle A}}{(\epsilon, w' \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta}\,{\scriptstyle Eq_1}$$

The condition on the A rule is $\mathcal{G}[w/w'] \vdash_E (x[w/w'] = x'[w/w'])$. By Lemma A.9.2, $\mathcal{G} \vdash_E (x = x')$ holds. Therefore the derivation is transformed into the following:

$$\dfrac{\dfrac{((\epsilon, w' \triangleright w); (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta)[w/w']}{(\epsilon, w' \triangleright w); (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta}\,{\scriptstyle Eq_1}}{(\epsilon, w' \triangleright w); (x, y \triangleright z); (u, v \triangleright x'); \Gamma \vdash \Delta}\,{\scriptstyle A}$$

The condition on the A rule is $\mathcal{G} \vdash_E (x = x')$. A_C is treated similarly.

□

Lemma A.9.4. *If* $(\epsilon, y \triangleright y); \Gamma[y/x] \vdash \Delta[y/x]$ *is derivable in* LS^e_{BBI}*, then* $(\epsilon, y \triangleright x); \Gamma \vdash \Delta$ *is derivable in* LS^e_{BBI}*.*

Proof. Symmetric to the proof in Lemma A.9.3. □

Theorem A.9.5. *If a sequent is derivable in* LS_{BBI}*, then it is also derivable in* LS^e_{BBI}*.*

Proof. Immediate by Lemma A.9.1, A.9.3, A.9.4. □

A.10 Substitution lemma for LS^e_{BBI}

This section proves the substitution lemma for the intermediate system LS^e_{BBI}, as this will be used in some proofs.

Lemma A.10.1. *If* $\mathcal{G} \vdash_E (x = y)$ *then for any substitution* $[s/t]$*, where* $t \neq \epsilon$*,* $\mathcal{G}[s/t] \vdash_E (x[s/t] = y[s/t])$*.*

Proof. Let $(\mathcal{G}, \sigma, \phi)$ be the solution to $\mathcal{G} \vdash_E (x = y)$, we prove this lemma by induction on the length of σ.

1. Base case, σ is an empty sequence. In this case, the sequence of substitutions ϕ is also empty, therefore $x = y$. As a result, it must be the case that $x[s/t] = y[s/t]$, so $\mathcal{G}[s/t] \vdash_E (x[s/t] = y[s/t])$ trivially holds.

2. Inductive case, assume $|\sigma| = n$. Let us look at the first rule application in σ. Assume this rule is Eq_1 (the case for Eq_2 is symmetric), and the principal relational atom is $(\epsilon, u \triangleright v)$, then σ is as follows.

$$\dfrac{\begin{array}{c}\mathcal{G}\phi \vdash_E (x\phi = y\phi) \\ \vdots \sigma' \\ \mathcal{G}'[v/u]; (\epsilon, v \triangleright v) \vdash_E (x[v/u] = y[v/u])\end{array}}{\mathcal{G}'; (\epsilon, u \triangleright v) \vdash_E (x = y)} \; {\scriptstyle Eq_1}$$

   (a) If $u = t$ and $v = s$, then the premise of the last rule application is already what we need.

   (b) If $u = t$ and $v \neq s$, we obtain the desired entailment as follows ($IH[x/y]$ stands for applying the induction hypothesis with the substitution $[x/y]$, we use double line to mean that the premise and the conclusion are equivalent).

$$\dfrac{\dfrac{\begin{array}{c} IH[v/s] \\ \mathcal{G}'[v/u][v/s]; (\epsilon, v \triangleright v) \vdash_E (x[v/u][v/s] = y[v/u][v/s])\end{array}}{\mathcal{G}'[s/u][v/s]; (\epsilon, v \triangleright v) \vdash_E (x[s/u][v/s] = y[s/u][v/s])}}{\mathcal{G}'[s/u]; (\epsilon, s \triangleright v) \vdash_E (x[s/u] = y[s/u])} \; {\scriptstyle Eq_1}$$

   (c) If $u = s$, we prove the substituted entailment as follows.

$$\dfrac{\dfrac{\begin{array}{c} IH[v/t] \\ \mathcal{G}'[v/u][v/t]; (\epsilon, v \triangleright v) \vdash_E (x[v/u][v/t] = y[v/u][v/t])\end{array}}{\mathcal{G}'[u/t][v/u]; (\epsilon, v \triangleright v) \vdash_E (x[u/t][v/u] = y[u/t][v/u])}}{\mathcal{G}'[u/t]; (\epsilon, u \triangleright v) \vdash_E (x[u/t] = y[u/t])} \; {\scriptstyle Eq_1}$$

Note that under this case if $v = t$, the proof is just a special case of the one above.

(d) If $v = t$, the case is shown below.

$$\dfrac{\dfrac{\dfrac{IH[s/v]}{\mathcal{G}'[v/u][s/v];(\epsilon, s \triangleright s) \vdash_E (x[v/u][s/v] = y[v/u][s/v])}}{\mathcal{G}'[s/v][s/u];(\epsilon, s \triangleright s) \vdash_E (x[s/v][s/u] = y[s/v][s/u])}}{\mathcal{G}'[s/v];(\epsilon, u \triangleright s) \vdash_E (x[s/v] = y[s/v])}\ {\scriptstyle Eq_1}$$

(e) If $v = s$, the proof is as follows.

$$\dfrac{\dfrac{\dfrac{IH[v/t]}{\mathcal{G}'[v/u][v/t];(\epsilon, v \triangleright v) \vdash_E (x[v/u][v/t] = y[v/u][v/t])}}{\mathcal{G}'[v/t][v/u];(\epsilon, v \triangleright v) \vdash_E (x[v/t][v/u] = y[v/t][v/u])}}{\mathcal{G}'[v/t];(\epsilon, u \triangleright v) \vdash_E (x[v/t] = y[v/t])}\ {\scriptstyle Eq_1}$$

(f) If $[s/t]$ and $[u/v]$ are independent, then we can switch the order of substitution, and derive the entailment as follows.

$$\dfrac{\dfrac{\dfrac{IH[s/t]}{\mathcal{G}'[v/u][s/t];(\epsilon, v \triangleright v) \vdash_E (x[v/u][s/t] = y[v/u][s/t])}}{\mathcal{G}'[s/t][v/u];(\epsilon, v \triangleright v) \vdash_E (x[s/t][v/u] = y[s/t][v/u])}}{\mathcal{G}'[s/t];(\epsilon, u \triangleright v) \vdash_E (x[s/t] = y[s/t])}\ {\scriptstyle Eq_1}$$

□

Since substitution does not break the equality entailment, we can show a substitution lemma for the system LS^e_{BBI}.

Lemma A.10.2 (Substitution in LS^e_{BBI})**.** *If there is a derivation for the sequent $\Gamma \vdash \Delta$ in LS^e_{BBI} then there is a derivation of the same height for the sequent $\Gamma[y/x] \vdash \Delta[y/x]$ in LS^e_{BBI}, where every occurrence of label x ($x \neq \epsilon$) is replaced by label y.*

Proof. The proof is basically the same as the one for LS_{BBI}, since there are a lot of common rules. For the rules that are changed, the case for $\top^* L$ is similar to those cases for additive rules. The proof for the rest of changed rules are straightforward with the help of Lemma A.10.1. □

A.11 Soundness of LS^{sf}_{BBI}

Theorem A.11.1. *If there is a derivation Π for a sequent $\mathcal{G}||\Gamma \vdash \Delta$ in LS^{sf}_{BBI}, then there is a derivation Π' for the sequent $\mathcal{G};\Gamma \vdash \Delta$ in LS^e_{BBI}.*

Proof. The soundness proof for this system is rather straightforward. To prove this, we show that each rule in LS^{sf}_{BBI} can be simulated in LS^e_{BBI}. To do this, one just need to unfold the structural rule applications into the derivation. For instance, we can simulate the *id* rule in LS^{sf}_{BBI} by using the following rules in LS^e_{BBI}:

$$\dfrac{\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w_1 = w_2)}{\mathcal{S}(\mathcal{G},\sigma);\Gamma;w_1 : P \vdash w_2 : P;\Delta}\ {\scriptstyle id}$$

$$\vdots\sigma$$

$$\mathcal{G};\Gamma;w_1 : P \vdash w_2 : P;\Delta$$

The above works because the *id* rule in LS^{sf}_{BBI} requires $\mathcal{G} \vdash_R (w_1 = w_2)$, which by definition ensures that $\mathcal{S}(\mathcal{G}, \sigma) \vdash_E (w_1 = w_2)$ holds. The case for $\top^* R$ works similarly. One thing to notice is that structural rules only add relational atoms into the current set, so except for $\mathcal{G}$ is becoming a bigger set, all the other structures in the sequent remain the same after the sequence σ of applications. Let us examine the simulation of $*R$ in LS^{e}_{BBI}.

$$\dfrac{\dfrac{\mathcal{S}(\mathcal{G},\sigma); \Gamma \vdash x' : A; w : A * B; \Delta \qquad \mathcal{S}(\mathcal{G},\sigma); \Gamma \vdash y' : B; w : A * B; \Delta}{\mathcal{S}(\mathcal{G},\sigma); \Gamma \vdash w : A * B; \Delta}\ {\scriptstyle *R}}{\mathcal{G}; \Gamma \vdash w : A * B; \Delta}\ \vdots\sigma$$

The condition of the $*R$ rule is $\mathcal{S}(\mathcal{G}, \sigma) \vdash_E (w = w')$. Since the LS^{sf}_{BBI} rule requires $\mathcal{G} \vdash_R (x, y \triangleright w)$, which by definition ensures that there is a solution $(\mathcal{G}, \sigma)$ such that $(x', y' \triangleright w') \in \mathcal{S}(\mathcal{G}, \sigma)$, and the following holds:

$$\mathcal{S}(\mathcal{G},\sigma) \vdash_E (x = x')$$
$$\mathcal{S}(\mathcal{G},\sigma) \vdash_E (y = y')$$
$$\mathcal{S}(\mathcal{G},\sigma) \vdash_E (w = w')$$

The last relation entailment is enough to guarantee that the $*R$ rule is applicable. To restore each branch, we need the Lemma A.10.2 (Substitution lemma for LS^{e}_{BBI}). Here we use double line to indicate the premise and the conclusion are equivalent. Let us look at the left branch. By the first relation entailment, there is a sequence σ' of Eq_1, Eq_2 applications so that $x\theta = x'\theta$. Therefore we can construct a proof for the left branch as follows.

$$\dfrac{\dfrac{\dfrac{\mathcal{S}(\mathcal{G},\sigma); \Gamma \vdash x : A; w : A * B; \Delta}{\mathcal{S}(\mathcal{G},\sigma)\theta; \Gamma\theta \vdash x\theta : A; w\theta : A * B; \Delta\theta}\ {\scriptstyle \text{Lemma A.10.2}}}{\mathcal{S}(\mathcal{G},\sigma)\theta; \Gamma\theta \vdash x'\theta : A; w\theta : A * B; \Delta\theta}}{\mathcal{S}(\mathcal{G},\sigma); \Gamma \vdash x' : A; w : A * B; \Delta}\ \vdots\sigma'$$

The case for $-\!\!* \ L$ is analogous. The rest rules are the same as in LS^{e}_{BBI}, thus we conclude that the rules in LS^{sf}_{BBI} are sound. □

A.12 Completeness of LS^{sf}_{BBI}

The completeness proof runs the same as in LS^{e}_{BBI}: if we add the structural rules E, U, A, A_C in LS^{sf}_{BBI}, then it becomes a superset of LS^{e}_{BBI}. Then we prove that these rules are admissible in LS^{sf}_{BBI} by showing they can permute through $*R$, $-\!\!* \ L$, *id*, and $\top^* R$.

First of all, let us show that when we add E, U, A, A_C (from LS^{e}_{BBI}) to LS^{sf}_{BBI}, its rules can simulate those ones in LS^{e}_{BBI}. As most of the rules are identical, the key part is the show the relation entailment is as powerful as the equality entailment. This is "built-in" the definition, so there is no surprise.

Lemma A.12.1. *If* $\mathcal{G} \vdash_E (w_1 = w_2)$*, then* $\mathcal{G} \vdash_R (w_1 = w_2)$*.*

Proof. Let σ be an empty list of rule applications, then $\mathcal{S}(\mathcal{G}, \emptyset) = \mathcal{G}$. Therefore by definition $\mathcal{G} \vdash_R (w_1 = w_2)$. □

If we change $\vdash_R$ to $\vdash_E$ in LS^{sf}_{BBI}, every rule is the same as the one in LS^{e}_{BBI}. Therefore $LS^{sf}_{BBI} + E + U + A + A_C$ is at least as powerful as LS^{e}_{BBI}.

Lemma A.12.2. *The rules E, U, A, and A_C are admissible in LS^{sf}_{BBI}.*

Proof. We show that the said rules can permute upwards through id, $\top^* R$, $*R$ and $-\!\!* \ L$, the other cases are cover by Lemma 4.1. We only give some examples here, the others are similar. The heart of the argument is that the application of structural rules are hidden inside the relation entailment, so we do not have to apply them explicitly.

Permute E through id, the suppose the original derivation runs as follows.

$$\cfrac{\cfrac{\mathcal{G}; (y, x \triangleright z); (x, y \triangleright z) \vdash_R (w_1 = w_2)}{\mathcal{G}; (y, x \triangleright z); (x, y \triangleright z) || \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle id}}{\mathcal{G}; (x, y \triangleright z) || \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle E}$$

The permuted derivation is:

$$\cfrac{\mathcal{G}; (x, y \triangleright z) \vdash_R (w_1 = w_2)}{\mathcal{G}; (x, y \triangleright z) || \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle id}$$

Assume $\mathcal{G}; (y, x \triangleright z); (x, y \triangleright z) \vdash_R (w_1 = w_2)$ is derived by applying a sequence σ of structural rules. Then $\mathcal{S}((\mathcal{G}; (x, y \triangleright z)), \sigma')$ can prove $\mathcal{G}; (x, y \triangleright z) \vdash_R (w_1 = w_2)$, where σ' is $E(\{(x, y \triangleright z)\}, \emptyset)$ followed by σ. That is, the application of E is absorbed in $\vdash_R$.

Permute A through id, the argument is similar. The original derivation is:

$$\cfrac{\cfrac{\mathcal{G}; (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x') \vdash_R (w_1 = w_2)}{\mathcal{G}; (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x') || \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle id}}{\mathcal{G}; (x, y \triangleright z); (u, v \triangleright x') || \Gamma; w_1 : P \vdash w_2 : P; \Delta}\ {\scriptstyle A}$$

The condition on the rule A is $\mathcal{G}; (x, y \triangleright z); (u, v \triangleright x') \vdash_E (x = x')$. Then we can omit the application of A, since $\mathcal{G}; (u, w \triangleright z); (y, v \triangleright w); (x, y \triangleright z); (u, v \triangleright x') \vdash_R (w_1 = w_2)$ implies $\mathcal{G}; (x, y \triangleright z); (u, v \triangleright x') \vdash_R (w_1 = w_2)$, one just need to add the A application ahead to the sequence of structural rules that derives the former relation entailment to get a new sequence of rules to derive the latter one. □

A.13 The Proof of the Heuristic Method

In the following proofs we use the tree representation of a set of relational atoms. Given a labelled binary tree tr as defined in Section 6, we say another labelled binary tree tr' is a permutation of tr if they have the same root and same multiset of leaves. A permutation on tr is generally done by applying the rules E, A on $Rel(tr)$. Figure 11 gives some examples on tree permutations. In Figure 11, (b) is permuted from (a) by using E on $(d, e \triangleright b)$, whereas (c) is permuted from (a) by using A on the two relational atoms in the original tree.

Lemma A.13.1. *Let tr be a labelled binary tree with a root labelled with r and a multiset of labels L for the leaves. If there is a labelled binary tree tr' with the same root and leaves labels respectively, then there is a variant tr'' of tr' and a sequence σ of E, A rule applications such that $Rel(tr'') \subseteq \mathcal{S}(Rel(tr), \sigma)$.*

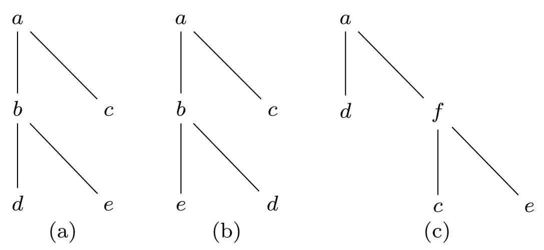

Figure 11: Examples of tree permutations.

Proof. Prove by induction on the width of the tree tr. We show that any distinct permutation(i.e., they are not variants of each other) of a tree can be achieved by using the rules E and A. Base case is when there are only two leaves in tr. In this case, there is only one relational atom in $Rel(tr)$, thus clearly there is only one distinct permutation of tr, which can be obtained by applying E on $Rel(tr)$.

The next case is when there are 3 leaves in the tree, meaning $Rel(tr)$ contains two relational atoms. In this case, it can be easily checked that there are 12 distinct permutations of tr, all of which can be derived by using E and A.

Inductive case, suppose the lemma holds for all trees with width less than n, consider a tree tr with width n. Suppose further that the root label of tr is r, it's two children are in the relational atom $(w_1, w_2 \triangleright r)$, and the multisets of leaves labels for the subtrees of w_1 and w_2 are L_1, L_2 respectively. Let tr' be a permutation of tr with the same root label and leaves labels, and in tr' the two children of the root label are in the relational atom $(w_3, w_4 \triangleright r)$. Suppose the multisets of leave labels for the subtrees of w_3, w_4 are L_3, L_4 respectively. Apparently, since $L_1 \cup L_2 = L_3 \cup L_4 = L$, every label in L_3 is either in L_1 or in L_2. Let $L' = L_1 \cap L_3$ and $L'' = L_2 \cap L_3$, then $L' \cup L'' = L_3$ and $(L_1 \setminus L') \cup (L_2 \setminus L'') = L_4$. By the induction hypothesis on the subtrees of w_1 and w_2, there exist w_5, w_6, w_7, w_8 s.t. $(w_5, w_6 \triangleright w_1), (w_7, w_8 \triangleright w_2)$ hold, and the subtrees of w_5, w_6, w_7, w_8 give the multisets of leaves $L', (L_1 \setminus L'), L'', (L_2 \setminus L'')$ respectively. Then we use the following derivation to permute the tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(w'', w''' \triangleright r); (w_6, w_8 \triangleright w'''); (w_5, w_7 \triangleright w''); \cdots}{(w', w_6 \triangleright r); (w'', w_8 \triangleright w'); (w_5, w_7 \triangleright w''); \cdots}\,A}{(w_6, w' \triangleright r); (w_8, w'' \triangleright w'); (w_5, w_7 \triangleright w''); \cdots}\,E \times 2}{(w_6, w' \triangleright r); (w_2, w_5 \triangleright w'); (w_8, w_7 \triangleright w_2); \cdots}\,A}{(w_6, w' \triangleright r); (w_2, w_5 \triangleright w'); (w_7, w_8 \triangleright w_2); \cdots}\,E}{(w_6, w_5 \triangleright w_1); (w_7, w_8 \triangleright w_2); (w_1, w_2 \triangleright r); \cdots}\,A}{(w_5, w_6 \triangleright w_1); (w_7, w_8 \triangleright w_2); (w_1, w_2 \triangleright r); \cdots}\,E$$

Now the subtrees of w'' and w''' has the same multisets of leaves as w_3 and w_4 respectively. Again by the induction hypothesis on the subtrees of w'' and w''', we obtain a tree tr'' which is a variant of tr'. □

Proof of Lemma 6.1.

Proof. The lemma restricts the labels of internal nodes to be free variables that are created after all the labels on the left hand side. Additionally, each free variable is only allowed to occur once in a tree. Therefore given a set $\mathcal{G}$ of relational atoms as the left hand side of those constraints, and any sequence σ of structural rule applications, the free variable labels for internal nodes can be assigned to any labels occur in $\mathcal{S}(\mathcal{G}, \sigma)$. By Lemma A.13.1, there exists a sequence σ of E, A applications which converts the tree on the left hand side to a tree which is a variant of the one on the right hand side, thus those constraints can be solved by assigning the free variables in the internal nodes to the corresponding labels. □

A.14 Proof of Formulae in the Conclusion

In this section we show the proofs of the four formulae in the conclusion. We extend LS_{BBI} in the obvious way to handle the additive connectives $\neg$ and $\vee$, where $\neg p = p \rightarrow \bot$ and $p \vee q = \neg(\neg p \wedge \neg q)$. Thus we obtain the left and right rules for $\neg, \vee$ as in the classical setting. To save space, we shall write r^n to mean the rule r is applied n times, and write $r_1; r_2$ to mean apply r_1 then apply r_2 on a sequent, when the order of rule applications does not matter.

1. To prove the formula $(F * F) \rightarrow F$, where $F = \neg(\top -\!\!* \neg\top^*)$, we use the following derivation in LS_{BBI}:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(w', w'' \triangleright \epsilon); (b', c' \triangleright w''); (b, c \triangleright w'); (b, c \triangleright a); \cdots}{(w', c' \triangleright w); (w, b' \triangleright \epsilon); \cdots}\,A}{(c', w' \triangleright w); (b, c \triangleright w'); (b', w \triangleright \epsilon); \cdots}\,E^2}{(b', w \triangleright \epsilon); (\epsilon, b \triangleright w); (c', c \triangleright \epsilon); \cdots}\,A}{(b, c \triangleright a); (b', b \triangleright \epsilon); (c', c \triangleright \epsilon); (\epsilon, \epsilon \triangleright \epsilon); \cdots}\,A}{(b, c \triangleright a); (b', b \triangleright \epsilon); (c', c \triangleright \epsilon); a : \top -\!\!* \neg\top^*; b' : \top, c' : \top \vdash}\,U}{(b, c \triangleright a); (b', b \triangleright b''); (c', c \triangleright c''); a : \top -\!\!* \neg\top^*; b' : \top, c' : \top; b" : \top^*; c'' : \top^* \vdash}\,\top^* L^2}{(b, c \triangleright a); (b', b \triangleright b''); (c', c \triangleright c''); a : \top -\!\!* \neg\top^*; b' : \top, c' : \top \vdash \_'' : \neg\top^*; c'' : \neg\top^*}\,\neg R^2}{(b, c \triangleright a); a : \top -\!\!* \neg\top^* \vdash b : \top -\!\!* \neg\top^*; c : \top -\!\!* \neg\top^*}\,-\!\!*\, R^2}{(b, c \triangleright a); b : \neg(\top -\!\!* \neg\top^*); c : \neg(\top -\!\!* \neg\top^*) \vdash a : \neg(\top -\!\!* \neg\top^*)}\,\neg L^2; \neg R}{\dfrac{a : F * F \vdash a : F}{\vdash a : (F * F) \rightarrow F}\,\rightarrow R}\,*L$$

   The correct relational atom that is required to split $a : \top -\!\!* \neg\top^*$ is $(w'', a \triangleright \epsilon)$. However, in the labelled sequent calculus we can only obtain $w'', w' \triangleright \epsilon$. Although w' and a both have exactly the same children, but the non-deterministic monoid allows the composition $b \circ c$ to be multiple elements, or even $\emptyset$ in $\mathcal{M}$. Thus we cannot conclude that $w' = a$. This can be solved by using P to replace w' by a, then use E to obtain $(w'', a \triangleright \epsilon)$ on the left hand side of the sequent, then the derivation can go through:

$$\dfrac{\dfrac{\dfrac{}{(w'', a \triangleright \epsilon); \cdots ; \vdash \epsilon : \top^*}\,\top^* R}{(w'', a \triangleright \epsilon); \cdots ; \epsilon : \neg\top^* \vdash}\,\neg L \qquad \dfrac{}{(w'', a \triangleright \epsilon); \cdots \vdash w'' : \top}\,\top R}{(w'', a \triangleright \epsilon); \cdots ; a : \top -\!\!* \neg\top^*; b' : \top, c' : \top \vdash}\,-\!\!*\, L$$

2. The trick to prove $(\neg\top^* -\!\!* \bot) \rightarrow \top^*$ is to create a relational atom $(w, w \triangleright w')$, as shown below.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{(\epsilon, \epsilon \triangleright w'); \cdots \vdash \epsilon : \top^*}\;\top^* R
    }{(w, w \triangleright w'); \cdots ; w : \top^* \vdash w : \top^*}\;\top^* L
  }{(w, w \triangleright w'); \cdots \vdash w : \neg \top^*; w : \top^*}\;\neg R
  \qquad
  \dfrac{}{(w, w \triangleright w'); \cdots ; w' : \bot \vdash w : \top^*}\;\bot L
}{
  \dfrac{
    (w, w \triangleright w'); w : \neg\top^* \mathrel{-\!\!*} \bot \vdash w : \top^*
  }{
    \dfrac{w : \neg\top^* \mathrel{-\!\!*} \bot \vdash w : \top^*}{\vdash w : (\neg\top^* \mathrel{-\!\!*} \bot) \to \top^*}\;\to R
  }\;T
}\;\mathrel{-\!\!*} L
$$

3. The proof for $(\top^* \wedge ((p * q) \mathrel{-\!\!*} \bot)) \to ((p \mathrel{-\!\!*} \bot) \vee (q \mathrel{-\!\!*} \bot))$ is as follows.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{}{\cdots ; c : q \vdash c : q; \cdots}\;id
      \qquad
      \dfrac{}{\cdots ; a : p \vdash a : p; \cdots}\;id
    }{(a, c \triangleright e); \cdots ; a : p; c : q \vdash e : p * q; \cdots}\;*R
    \qquad
    \dfrac{}{\cdots e : \bot \vdash \cdots}\;\bot L
  }{(e, \epsilon \triangleright e); (a, c \triangleright e); (a, \epsilon \triangleright b); (c, \epsilon \triangleright d); \epsilon : (p * q) \mathrel{-\!\!*} \bot; a : p; c : q \vdash b : \bot; d : \bot}\;\mathrel{-\!\!*} L
}{
  \dfrac{(a, c \triangleright e); (a, \epsilon \triangleright b); (c, \epsilon \triangleright d); \epsilon : (p * q) \mathrel{-\!\!*} \bot; a : p; c : q \vdash b : \bot; d : \bot}{
    \dfrac{(a, \epsilon \triangleright b); (c, \epsilon \triangleright d); \epsilon : (p * q) \mathrel{-\!\!*} \bot; a : p; c : q \vdash b : \bot; d : \bot}{
      \dfrac{\epsilon : (p * q) \mathrel{-\!\!*} \bot \vdash \epsilon : p \mathrel{-\!\!*} \bot; \epsilon : q \mathrel{-\!\!*} \bot}{
        \dfrac{w : \top^*; w : (p * q) \mathrel{-\!\!*} \bot \vdash w : p \mathrel{-\!\!*} \bot; w : q \mathrel{-\!\!*} \bot}{
          \dfrac{w : \top^* \wedge ((p * q) \mathrel{-\!\!*} \bot) \vdash w : (p \mathrel{-\!\!*} \bot) \vee (q \mathrel{-\!\!*} \bot)}{\vdash w : (\top^* \wedge ((p * q) \mathrel{-\!\!*} \bot)) \to ((p \mathrel{-\!\!*} \bot) \vee (q \mathrel{-\!\!*} \bot))}\;\to R
        }\;\wedge L; \vee R
      }\;\top^* L
    }\;\mathrel{-\!\!*} R^2
  }\;T
}\;U
$$

4. The proof for $\neg(\top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))))$ in LS_{BBI} is as follows.

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{
          \dfrac{}{(c, b \triangleright \epsilon); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C \vdash \epsilon : A}\;id
        }{(c, b \triangleright d); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C; d : \top^* \vdash d : A}\;\top^* L
      }{(c, b \triangleright d); (a, b \triangleright \epsilon); \epsilon : A; a : B; c : C \vdash d : \top^* \to A}\;\to R
    }{(a, b \triangleright \epsilon); \epsilon : A; a : B \vdash b : C \mathrel{-\!\!*} (\top^* \to A)}\;\mathrel{-\!\!*} R
  }{(a, b \triangleright \epsilon); \epsilon : A; a : B; b : \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}\;\neg L
}{
  \dfrac{\epsilon : A; \epsilon : B * \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}{
    \dfrac{w : \top^*; w : A; w : B * \neg(C \mathrel{-\!\!*} (\top^* \to A)) \vdash}{
      \dfrac{w : \top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))) \vdash}{\vdash w : \neg(\top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))))}\;\neg R
    }\;\wedge L
  }\;\top^* L
}\;*L
$$