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

Zhé Hóu, Alwen Tiu and Rajeev Goré

Logic and Computation Group, Research School of Computer Science
The Australian National University, Canberra, ACT 0200, Australia

Abstract. We present a labelled sequent calculus for Boolean BI (BBI), a classical variant of the logic of Bunched Implication. 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 conjecture that different semantics for BBI and some axioms in concrete models can be captured by adding extra structural rules.

1 Introduction

The logic of bunched implications (BI) was introduced to reason about resources using additive connectives $\wedge$, $\vee$, $\rightarrow$, $\top$, $\bot$, and multiplicative connectives $\top^*$, $*$, $-\!\!*$ [14]. 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 [17], 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) [10]. These semantics give different logics w.r.t. validity, i.e., $BBI_{ND}, BBI_{PD}, BBI_{TD}$ respectively, and all are undecidable [3, 10]. The purely syntactic proof theory of BBI also comes in three flavours: Hilbert calculi [16, 5], display calculi [1] and nested sequent calculi [15]. All are sound and complete w.r.t. the ND-semantics [5, 1, 15].

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 [9, 8]. They remark that "the adaptation of this tableaux system to BBI_{TD} should be straightforward (contrary to BBI_{ND})" [11]. We return to these issues in Section 7.

The structural rules of display calculi, especially the contraction rule on structures, are impractical for backward proof search. Nested sequents also face similar problems, and although Park et al. 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 [15]. The labelled tableaux of Larchey-Wendling and Galmiche compile all structural rules into PD-monoidal constraints, and are cut-fee complete for $\mathrm{BBI}_{\mathrm{PD}}$ using a potentially infinite counter-model construction [8]. But effective proof search is only a "perspective" and is left as further work [8, 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 [10] while "memory models" for BBI are restricted to have *indivisible units*: "the empty memory cannot be split into non-empty pieces" [3]. Our goal is to give a labelled proof system for BBI based upon the ND-semantics which easily extend to the PD- and TD-semantics, and also these other, more "practical", semantics.

Our labelled sequent calculus LS_{BBI} for BBI adopts some features from existing labelled tableaux for BBI [9] and existing labelled sequent calculi for modal logics [12]. Unlike these calculi, some LS_{BBI}-rules contain substitutions on labels. From a proof-search perspective, labelled calculi are no better than display calculi since they require extra-logical rules to explicitly encode the frame conditions of the underlying (Kripke) semantics. Such rules, which we refer to simply as structural rules, are just as bad as display postulates for proof search since we may be forced to explore all potential models. As a step towards our goal, we show that the applications of these structural rules can be localised around logical rules. Thus these structural rules are only triggered by applications of logical rules, leading to a purely syntax-driven proof search procedure for LS_{BBI}.

Our work is novel from two 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 obtain a constructive cut-elimination procedure. Compared to the nested sequent calculus of Park et al., our calculus is much simpler, and generally gives much shorter proofs. 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 give some structural rules which we conjecture will give cut-free labelled calculi for all the other semantics mentioned above. Detailed proofs of all claims except this conjecture are available in [7].

2 Syntax and semantics of BBI

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

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

The Kripke semantics of BBI employs a ternary relation of worlds based on a non-deterministic monoid structure, á la Galmiche and Larchey-Wendling [5]. A relational frame is a triple $(\mathcal{M}, \triangleright, \epsilon)$, where $\triangleright \subseteq \mathcal{M} \times \mathcal{M} \times \mathcal{M}$. Following [5], we write $a, b \triangleright c$ instead of $\triangleright(a, b, c)$ and also adopt a single unit ϵ, rather than a set of units [4]. 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	$\exists k, (a, k \triangleright d) \& (b, c \triangleright k) \Rightarrow \exists l, (a, b \triangleright l) \& (l, c \triangleright d)$.

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. We do not restrict this monoid to be cancellative, so $x, y \triangleright x$ does not imply $y = \epsilon$.

Let $(\mathcal{M}, \triangleright, \epsilon)$ be a relational frame and $v : Var \to \mathcal{P}(\mathcal{M})$ be a *valuation*. A *forcing relation* "$\Vdash$" between $m \in \mathcal{M}$ and BBI-formulae is defined as follows [5]:

$m \Vdash \top^*$	iff $m = \epsilon$	$m \Vdash P$	iff $P \in Var$ and $m \in v(P)$
$m \Vdash \bot$	iff never	$m \Vdash A \vee B$	iff $m \Vdash A$ or $m \Vdash B$
$m \Vdash \top$	iff always	$m \Vdash A \wedge B$	iff $m \Vdash A$ and $m \Vdash B$
$m \Vdash \neg A$	iff $m \nVdash A$	$m \Vdash A \to B$	iff $m \nVdash A$ or $m \Vdash B$

$m \Vdash A * B$ iff $\exists a, b.(a, b \triangleright m$ and $a \Vdash A$ and $b \Vdash B)$

$m \Vdash A \mathrel{-\!\!*} B$ iff $\forall a, b.((m, a \triangleright b$ and $a \Vdash A)$ implies $b \Vdash B)$

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

3 The labelled sequent calculus for BBI

The inference rules of LS_{BBI} are shown in Figure 1, where P is an atomic formula, A, B are formulae, w, x, y, z are in the set $LVar$ of label variables, and ϵ is the label constant. We define a mapping $\rho : \{\epsilon\} \cup LVar \to \mathcal{M}$ from labels to worlds. We overload the notation in an obvious way so that ϵ is the empty world in the semantics and the label constant, while $\triangleright$ is the ternary relation in the semantics and in the calculus. Therefore, from now on, we demand that the mappings from labels to worlds obeys $\forall \rho. \rho(\epsilon) = \epsilon$.

A sequent $\Gamma \vdash \Delta$ consists of a semi-colon separated multiset Γ of relational atoms and labelled formulae and a semi-colon separated multiset Δ of labelled formulae. Note that relational atoms can appear only in the left-hand-side Γ.

A labelled formula $w : A$ means formula A is true in world $\rho(w)$. A relational atom $(x, y \triangleright z)$, which we always write inside parentheses, 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.

Definition 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 $A \in \Gamma$ is true then so is some $B \in \Delta$.*

BBI-validity of a formula A corresponds to the sequent validity of $\vdash x : A$ where x is an arbitrary label. This notion of validity is common for BBI [10, 15] and CBI [2], but is stronger than BI-validity [16], where A is only required to be true at the world ϵ in all BI-models. Using labelled sequents, BI-validity informally corresponds to the sequent validity of $\vdash \epsilon : A$. For example, the formula $\top^*$ is BI-valid, but it is not BBI-valid.

In our sequents, the structural connective ";" means additive "and" in the antecedent and means additive "or" in the succedent. Traditional sequents use "," in this role, but our notation is consistent with sequent calculi for the family of Bunched Implication (BI) logics, where ";" is the additive structural connective and "," is the multiplicative structural connective. The "," connective does not appear explicitly in our sequents but is encoded implicitly in the relational atoms.

In each rule, the formula/relational atom shown explicitly in the conclusion is the *principal formula/relational atom*. In the cut rule, the cut-formula is $x : A$.

The semantics of $*$ involves an existential condition, so rules $*L$ and $*R$ incorporate existential and universal quantifiers respectively, conversely for the rules $-\!\!* L$ and $-\!\!* R$. Therefore, rules $*L$ and $-\!\!* R$ create a premise containing new relations, and the labels in the created relation must be fresh (except for the label of the principal formula). Rules $*R$ and $-\!\!* L$ create premises using existing relations from the conclusion. Further, in rules A and A_C, the label w must be fresh in the premise, as it represents a new partition of the original world. Contraction admissibility [7] requires the rule A_C, a special case of A with a built-in contraction on $(x, y \triangleright x)$. The rule $\top^* L$ utilises a substitution $[\epsilon/x]$ where $\Gamma[y/x]$ is the result of replacing every occurrence of x in Γ by y.

The *additive rules* ($\bot L$, $\top R$, $\wedge L$, $\wedge R$, $\to L$, $\to R$) 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}.

Note that we start (at the bottom) by labelling the formula with an arbitrary world a. Since provability is preserved by substitutions of labels (Lemma 1), provability of $\vdash a : F$ implies provability of $\vdash w : F$, for any world w. Thus, if a formula is provable, then it is true in every world in every model.

3.1 Soundness and completeness

Definition 2 (Sequent Falsifiability). *A sequent $\Gamma \vdash \Delta$ is falsifiable if some $(\mathcal{M}, \triangleright, \epsilon)$, v and ρ make every member of Γ true and every member of Δ false.*

Theorem 1 (Soundness). *The labelled sequent calculus LS_{BBI} is sound w.r.t. the non-deterministic monoidal Kripke semantics for BBI.*

For each rule in LS_{BBI}, we show that if the conclusion is falsifiable, then the premise is falsifiable. We prove the completeness of LS_{BBI} by showing that every derivation of a formula in the Hilbert system for BBI [5] can be mimicked in LS_{BBI}, possibly using cuts. Detailed proofs are available in our arXiv paper [7].

Identity and Cut:

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

Logical Rules:

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

$$\frac{}{\Gamma \vdash w : \top; \Delta}\ \top R \qquad \frac{}{\Gamma \vdash \epsilon : \top^*; \Delta}\ \top^* R$$

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

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

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

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

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

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:

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

Fig. 1. The (cut-free) labelled sequent calculus LS_{BBI} for Boolean BI.

Theorem 2 (Completeness). *The labelled sequent calculus LS_{BBI} is complete w.r.t. the non-deterministic monoidal Kripke semantics for BBI.*

3.2 Cut elimination

We now state the cut-elimination theorem for our labelled sequent calculus. The general proof outlined here is similar to the cut-elimination proof for labelled

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\epsilon, a \rhd a); (a, \epsilon \rhd a); a : A \vdash \epsilon : \top^*}\,\top^* R \quad \dfrac{}{(\epsilon, a \rhd a); (a, \epsilon \rhd a); a : A \vdash a : A}\,id}{(\epsilon, a \rhd a); (a, \epsilon \rhd a); a : A \vdash a : \top^* * A}\,{*R}}{(a, \epsilon \rhd a); a : A \vdash a : \top^* * A}\,E}{a : A \vdash a : \top^* * A}\,U}{\vdash a : A \to (\top^* * A)}\,{\to R}$$

Fig. 2. An example derivation in LS_{BBI}.

systems for modal logic [12], 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 our arXiv paper [7].

Given a derivation Π, its *height* $ht(\Pi)$ is defined as the length of the longest branch in the derivation tree of Π. The substitution lemma shows that provability is preserved under arbitrary substitutions of labels.

Lemma 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 $ht(\Pi') \leq ht(\Pi)$.*

The admissibility of weakening and contraction on both formulae and relational atoms follows unsurprisingly from our design of the calculus, as does the invertibility of the inference rules. Note that the rule A_C exists for avoiding contraction on relational atoms when applying the rule A: see [7] for details.

Suppose an application of the *cut* rule has premise derivations Π_1 and Π_2 and a cut-formula $x : A$ of size $|A|$. The *cut height* is $ht(\Pi_1) + ht(\Pi_2)$ and the complexity of such a *cut* rule is $(|A|, ht(\Pi_1) + ht(\Pi_2))$. If there are multiple branches in Π_1, then $ht(\Pi_1)$ shall be the height of the longest branch, similarly for $ht(\Pi_2)$. The strict ordering for both parts of the pair is $>$ on natural numbers.

Theorem 3 (Cut-elimination). *If $\Gamma \vdash \Delta$ is derivable in LS_{BBI}, then it is also derivable in LS_{BBI} without using the cut rule.*

Proof. By induction on the complexity 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 less complexity. The argument for termination is similar to the cut-elimination proof for *G3ip* [13]. 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. □

4 Localising structural rules

To obtain an effective proof search procedure for LS_{BBI}, we need to restrict the use of structural rules, which in LS_{BBI}, can permute upwards through all rules except for id, $\top^* R$, $*R$, and $-\!\!* L$. The other logical rules do not rely on relational atoms, so we can apply them whenever possible. This allows us to design a more compact proof system where applications of structural rules are separated into a special entailment relation for relational atoms. We localise the structural rules Eq_1 and Eq_2 first, and then localise the other structural rules.

Let r be an instance of a structural rule where the substitution used in the rule instance is θ: which is the identity substitution except when r is Eq_1 or Eq_2. We can view r (upwards) as a function that takes a set of relational atoms (in the conclusion of the rule) and outputs another set (in the premise). We write $r(\mathcal{G}, \theta)$ for the output relational atoms of an instance of r with substitution θ and with conclusion containing $\mathcal{G}$. Let σ be a sequence of instances of structural rules $[r_1(\mathcal{G}_1, \theta_1); \cdots ; r_n(\mathcal{G}_n, \theta_n)]$. Given a set of relational atoms $\mathcal{G}$, the result of the (backward) application of σ to $\mathcal{G}$, denoted by $\mathcal{S}(\mathcal{G}, \sigma)$, is defined as:

$$\mathcal{S}(\mathcal{G}, \sigma) = \begin{cases} \mathcal{G} & \text{if} \quad \sigma = [\,] \\ \mathcal{S}(\mathcal{G}\theta \cup r(\mathcal{G}', \theta), \sigma') & \text{if} \quad \mathcal{G}' \subseteq \mathcal{G} \text{ and } \sigma = [r(\mathcal{G}', \theta); \sigma'] \\ \text{undefined} & \quad \text{otherwise} \end{cases}$$

Given a $\sigma = [r_1(\mathcal{G}_1, \theta_1); \cdots ; r_n(\mathcal{G}_n, \theta_n)]$, we denote with $subst(\sigma)$ the composite substitution $\theta_1 \circ \cdots \circ \theta_n$, where $t(\theta_1 \circ \theta_2)$ means $(t\theta_1)\theta_2$.

Definition 3. *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 Eq_1 or Eq_2 structural rules such that $\mathcal{S}(\mathcal{G}, \sigma)$ is defined, and $u\theta = v\theta$, where $\theta = subst(\sigma)$.*

The equality entailment does not fully capture the reflexivity, transitivity, and symmetry of equality. Rather, the structural rule E is used when symmetry is required to derive an equality. As a second step, we isolate the rest of the structural rules into a separate entailment relation, as we did with Eq_1 and Eq_2.

Definition 4. *Let $\mathcal{G}$ be a set of relational atoms. The entailment relation $\vdash_R$ has the following two forms:*

1. *$\mathcal{G} \vdash_R (w_1 = w_2)$ holds iff there is a sequence σ of E, U, A, A_C applications so that $\mathcal{S}(\mathcal{G}, \sigma) \vdash_E (w_1 = w_2)$.*
2. *$\mathcal{G} \vdash_R (w_1, w_2 \triangleright w_3)$ holds iff there is a sequence σ of E, U, A, A_C applications 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 $\mathcal{S}(\mathcal{G}, \sigma) \vdash_E (w_3 = w_3')$.*

Thus we can move all the structural rules into $\vdash_R$, giving an intermediate system LS^{sf}_{BBI}, with changed logical rules shown in Figure 3. We write $\mathcal{G}||\Gamma \vdash \Delta$ to emphasise that the left hand side of a sequent is partitioned into relational atoms $\mathcal{G}$ and labelled formulae Γ. Note that the entailment $\vdash_R$ is not a premise, but a side condition for the rule to be applicable.

Theorem 4. *A sequent $\Gamma \vdash \Delta$ is derivable in LS_{BBI} iff it is derivable in LS^{sf}_{BBI}.*

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

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

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

$\dagger$: σ is a derivation of $\mathcal{G} \vdash_R (x, y \triangleright w)$ $\qquad$ $\ddagger$: σ is a derivation of $\mathcal{G} \vdash_R (x, w \triangleright z)$

Fig. 3. Changed rules in LS^{sf}_{BBI}.

5 Mapping proof search to constraint solving

The intermediate system LS^{sf}_{BBI} is essentially a variant of LS_{BBI} that packages structural rules at certain points in the proof search. We can further separate proof search into two stages: guessing the shape of the derivation tree, and then checking that each entailment $\vdash_R$ can be proved. The latter involves guessing a relational atom to use in the $*R$ or $\mathrel{-\!\!*} L$ rule which also satisfies the equality constraints in the id and $\top^* R$ rules. We formalise this via a symbolic proof system where the relational atoms in the rules $*R, \mathrel{-\!\!*} L$ are selected lazily via the introduction of *free variables*, which are essentially existential variables that must be instantiated to concrete labels satisfying all the constraints in the derivation.

Free variables help to make the right decisions when applying $*R$ and $\mathrel{-\!\!*} L$ rules. That is, suppose the id rule (or analogously the $\top^* R$ rule) requires a free variable $\mathbf{x}$ to be equal to a label w, we can satisfy this by globally assigning w to $\mathbf{x}$. In this way, the search space is reduced, and many applications of structural rules are guided by the result of id and $\top^* R$ rules. See Section 6 for an example.

In our symbolic system $FVLS_{BBI}$, free variables are denoted by $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. We use $\mathbf{u}, \mathbf{v}, \mathbf{w}$ for either labels or free variables, and a, b, c for ordinary labels. A *symbolic sequent* is a sequent 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 emphasise that it contains no free variables. The symbolic proof system $FVLS_{BBI}$ is given in Figure 4. The rules are mostly similar to LS^{sf}_{BBI}, but lacking the entailment relations $\vdash_R$. Instead, constraints containing new free variables are introduced when applying $*R$ and $\mathrel{-\!\!*} L$ backwards. Notice also that in $FVLS_{BBI}$, the $*R$ and $\mathrel{-\!\!*} L$ rules do not compute the set $\mathcal{S}(\mathcal{G}, \sigma)$. So the relational atoms in $FVLS_{BBI}$ are those that are created by $*L, \mathrel{-\!\!*} R, \top^* L$. We refer to a derivation in $FVLS_{BBI}$ as a *symbolic derivation*.

An *equality constraint* is an expression of the form $\mathcal{G} \vdash^?_R (\mathbf{u} = \mathbf{v})$, and a *relational constraint* is of the form $\mathcal{G} \vdash^?_R (\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$. 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 $\mathcal{G}$ of $\mathfrak{c}$. We write $\mathcal{G} \vdash^?_R C$ for either an equality or relational constraint. We write $fv(\mathfrak{c})$ for the set of free variables in $\mathfrak{c}$, and $fv(\mathcal{C})$ to denote the set of free variables in a set of constraints $\mathcal{C}$.

$$\frac{}{\mathcal{G}||\Gamma; \mathbf{w}_1 : P \vdash \mathbf{w}_2 : P; \Delta}\ id \qquad \frac{}{\mathcal{G}||\Gamma; \mathbf{w} : \bot \vdash \Delta}\ \bot L \qquad \frac{}{\mathcal{G}||\Gamma \vdash \mathbf{w} : \top; \Delta}\ \top R$$

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

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

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

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

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

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

$\dagger$: a, b do not occur in the conclusion of $*L$ $\ddagger$: a, c do not occur in the conclusion of $\mathrel{-\!\!*} R$
$\sharp$: $\mathbf{x}, \mathbf{y}$ do not occur in the conclusion of $*R$ $\natural$: $\mathbf{x}, \mathbf{z}$ do not occur in the conclusion of $\mathrel{-\!\!*} L$

Fig. 4. Labelled sequent calculus $FVLS_{BBI}$ for Boolean BI.

Definition 5 (Constraint systems). *A* constraint system *is a pair* $(\mathcal{C}, \preceq)$ *of a set of constraints and a well-founded partial order on elements of* $\mathcal{C}$ *satisfying* ***Monotonicity****:* $\mathfrak{c}_1 \preceq \mathfrak{c}_2$ *implies* $\mathcal{G}(\mathfrak{c}_1) \subseteq \mathcal{G}(\mathfrak{c}_2)$*. It is* well-formed *if it also satisfies* ***Unique variable origin****:* $\forall \mathbf{x}$ *in* $\mathcal{C}$*, there exists a unique minimum (w.r.t.* $\preceq$*) constraint* $\mathfrak{c}(\mathbf{x}) = \mathcal{G}_x \vdash^?_R (\mathbf{u}, \mathbf{v} \triangleright \mathbf{w})$ *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* $\mathfrak{c}' \preceq \mathfrak{c}(\mathbf{x})$*. Such a* $\mathfrak{c}(\mathbf{x})$ *is the origin of* $\mathbf{x}$*.*

From now on, we use $\mathfrak{c}(\mathbf{x})$ for the origin (constraint) of $\mathbf{x}$, as defined above. We use $\mathbb{C}$ 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. Note that the labels for constraints correspond to those in Figure 4.

Definition 6. *To a given symbolic derivation* Π*, we associate a set of constraints* $\mathcal{C}(\Pi)$ *as follows where the lowest rule instance of* Π *is:*

id	$\mathcal{C}(\Pi) = \{\mathcal{G} \vdash^?_R (\mathbf{w}_1 = \mathbf{w}_2)\}$
$\top^* R$	$\mathcal{C}(\Pi) = \{\mathcal{G} \vdash^?_R (\mathbf{w} = \epsilon)\}$
$*R$	$\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \cup \mathcal{C}(\Pi_2) \cup \{\mathcal{G} \vdash^?_R (\mathbf{x}, \mathbf{y} \triangleright \mathbf{w})\}$ *where the left premise derivation is* Π_1 *and the right-premise derivation is* Π_2
$\mathrel{-\!\!*} L$	$\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \cup \mathcal{C}(\Pi_2) \cup \{\mathcal{G} \vdash^?_R (\mathbf{x}, \mathbf{w} \triangleright \mathbf{y})\}$ *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* $\{\Pi_1, \ldots, \Pi_n\}$*, then* $\mathcal{C}(\Pi) = \mathcal{C}(\Pi_1) \cup \cdots \cup \mathcal{C}(\Pi_n)$.

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 rule applications in the derivation tree of Π then naturally induces a partial order on $\mathcal{C}(\Pi)$. That is, let $\preceq^{\Pi}$ be an ordering on $\mathcal{C}(\Pi)$ defined via: $\mathfrak{c}_1 \preceq^{\Pi} \mathfrak{c}_2$ iff $r(\mathfrak{c}_1)$ is applied below $r(\mathfrak{c}_2)$ on the same branch. Obviously $\preceq^{\Pi}$ is a partial order. The following property of $\mathcal{C}(\Pi)$ is easy to verify.

Lemma 2. *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.*

Given a symbolic derivation Π, let $\mathbb{C}(\Pi)$ be a constraint system $(\mathcal{C}(\Pi), \preceq^{\Pi})$ defined as above. From Lemma 2, if $\mathbb{C}(\Pi) \neq \{\ \}$, 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 the solvability of a constraint system. This requires to capture that (ternary) relational atoms created by the solution must be accumulated across different constraints, in order to guarantee the soundness of $FVLS_{BBI}$. A *free variable substitution* θ is a mapping from free variables to free variables or labels with finite domain. We denote with $dom(\theta)$ the domain of θ. Given θ and a set V of free variables, $\theta \restriction V$ is the substitution obtained from θ by restricting the domain to V as shown below left. Given θ and θ' such that $dom(\theta') \subseteq dom(\theta)$, we define $\theta \setminus \theta'$ as the substitution as shown below right:

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

Definition 7 (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 structural rules s.t.*

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

The minimum constraints of a well-formed constraint system are simple.

Definition 8 (Restricting a constraint system). *Let $\mathbb{C} = (\mathcal{C}, \preceq)$ be a well-formed constraint system, and $\mathfrak{c}$ be a minimum (simple) constraint in $\mathbb{C}$. Let (θ, σ) be a solution to $\mathfrak{c}$ and $\mathcal{G}' = \mathcal{S}(\mathcal{G}(\mathfrak{c}), \sigma)$. Define a function f on constraints:*

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

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

Lemma 3. *The pair $(\mathcal{C}', \preceq') = \mathbb{C} \restriction (\mathfrak{c}, \theta, \sigma)$ is a well-formed constraint system.*

Definition 9 (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.*

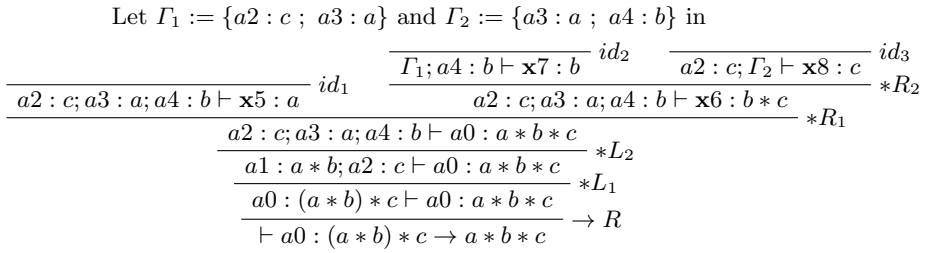

Fig. 5. A symbolic derivation for $((a * b) * c) \to (a * (b * c))$.

If $n \geq 1$ *then there must exist some minimum (simple) constraint in* $\mathbb{C}$*. For any minimum constraint* $\mathfrak{c}_i$*, let* $\theta_i = \theta \restriction fv(\mathfrak{c}_i)$*, 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* $\mathbb{C} \restriction (\mathfrak{c}_i, \theta_i, \sigma_i)$*.*

In Definition 9, suppose a constraint system $\mathbb{C} = (\{\mathfrak{c}_1, \cdots, \mathfrak{c}_n\}, \preceq)$ has a solution $(\theta, \{\sigma_1, \cdots, \sigma_n\})$. For each constraint $\mathfrak{c}_i$ in $\mathbb{C}$, let $\mathfrak{c}'_i$ be the simple constraint obtained from $\mathfrak{c}_i$ in the process of restricting $\mathbb{C}$. Then there is a solution (θ_i, σ_i) to $\mathfrak{c}'_i$, where $\theta_i = \theta \restriction fv(\mathfrak{c}'_i)$, and $\sigma_i \in \{\sigma_1, \cdots, \sigma_n\}$.

Theorem 5 (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}*.*

The proof [7] uses induction on the height of symbolic derivations. Intuitively, the proof progressively "grounds" a symbolic derivation, root-upwards. At each inductive step we show that grounding the premises corresponds to restricting the constraint system induced by the symbolic derivation.

We prove the completeness of $FVLS_{BBI}$ by showing that for every cut-free derivation Π of a (ground) sequent in LS^{sf}_{BBI}, there is a symbolic derivation Π' of the same sequent such that $\mathcal{C}(\Pi')$ is solvable. Obviously, 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. So the key is to recover the omitted relational atoms in each sequent from the constraint system while solving constraints. The full proof is in our arXiv paper [7].

Theorem 6 (Completeness). *If a sequent has a* LS^{sf}_{BBI} *derivation* Π*, then it has a symbolic derivation* Π' *such that* $\mathcal{C}(\Pi')$ *is solvable.*

6 A heuristic and experimental results

A heuristic. Suppose we want to prove $((a*b)*c) \to (a*(b*c))$. Using $FVLS_{BBI}$, we build a symbolic derivation as in Figure 5 (right associativity for connectives is assumed). The constraints in the derivation are listed below, with the corresponding rules that generate them:

$$
\begin{array}{ll}
id_3\colon & (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a2 = \mathbf{x}8) \\
id_2\colon & (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a4 = \mathbf{x}7) \\
*R_2\colon & (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (\mathbf{x}7, \mathbf{x}8 \rhd \mathbf{x}6) \\
id_1\colon & (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a3 = \mathbf{x}5) \\
*R_1\colon & (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (\mathbf{x}5, \mathbf{x}6 \rhd a0).
\end{array}
$$

From the constraints generated by id rules, we already know how $\mathbf{x}5, \mathbf{x}7, \mathbf{x}8$ should be assigned. In the following, we shall write $(a1, a2 \rhd a0); (a3, a4 \rhd a1)$ as $\mathcal{G}$, $(a3, \mathbf{x}6 \rhd a0); (a2, a4 \rhd \mathbf{x}6)$ as C, $\mathbf{1}$ as the identity substitution, and $\emptyset$ as an empty sequence of rule applications. Now it is much easier to solve the constraint system with the known information. The last constraint can be solved by $([a3/\mathbf{x}5, w/\mathbf{x}6], A((a1, a2 \rhd a0); (a3, a4 \rhd a1), \mathbf{1}))$, which generates $(a3, w \rhd a0); (a2, a4 \rhd w)$. The resultant constraint system is restricted to the following:

$$
\begin{array}{ll}
id_3\colon & (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a2 = \mathbf{x}8) \\
id_2\colon & (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a4 = \mathbf{x}7) \\
*R_2\colon & (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (\mathbf{x}7, \mathbf{x}8 \rhd w) \\
id_1\colon & (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a3 = a3).
\end{array}
$$

Now the last constraint is trivially solved by $(\mathbf{1}, \emptyset)$. The second last constraint can be solved by $([a4/\mathbf{x}7, a2/\mathbf{x}8], E((a2, a4 \rhd w), \mathbf{1}))$, which generates $(a4, a2 \rhd w)$. The remaining constraints are restricted as below.

$$
\begin{array}{ll}
id_3\colon & (a4, a2 \rhd w); (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a2 = a2) \\
id_2\colon & (a4, a2 \rhd w); (a3, w \rhd a0); (a2, a4 \rhd w); (a1, a2 \rhd a0); (a3, a4 \rhd a1) \vdash^{?}_{R} (a4 = a4)
\end{array}
$$

These constraints are trivially solvable by $(\mathbf{1}, \emptyset)$. Therefore the overall solution to the original (first) constraint system is $([a3/\mathbf{x}5, w/\mathbf{x}6, a4/\mathbf{x}7, a2/\mathbf{x}8], A((a1, a2 \rhd a0); (a3, a4 \rhd a1), \mathbf{1}) \cdot E((a2, a4 \rhd w), \mathbf{1}))$, the reader can check that this solution is compliant with our definitions in Section 5.

But there is a simpler way to see that the label w must exist: the two ternary relational atoms in $\mathcal{G}$ manifest that $a0$ can be split into $a2, a3, a4$. This is exactly what C says. For any variant of $\mathcal{G}$ that describes the same splitting of $a0$ as C, the "internal" node $\mathbf{x}6$ can always be assigned to either an existing label or a label generated by the associativity rule. In the example, $\mathbf{x}6$ cannot be matched to any existing label, so we can assign $\mathbf{x}6$ to be a fresh label globally, and add C to the l.h.s. of the successor constraints in the partial order $\prec$. Similarly for any variant of C with the same splitting of $a0$. The next lemma extends this idea.

Lemma 4. *Given constraints $\mathfrak{c}_1 \lessdot \cdots \lessdot \mathfrak{c}_n$ with $\mathcal{G} = \mathcal{G}(\mathfrak{c}_1) = \cdots = \mathcal{G}(\mathfrak{c}_n)$ s.t. the r.h.s. of $\mathfrak{c}_1$ to $\mathfrak{c}_n$ form a binary tree where every internal node is some free variable $\mathbf{x}$ with $\mathfrak{c}_1 \preceq \mathfrak{c}(\mathbf{x})$, and the other nodes are non-ϵ labels: if $\mathcal{G}' \subseteq \mathcal{G}$ and $\mathcal{G}'$ forms a binary tree with the same root and leaves, then $\mathfrak{c}_1, \cdots, \mathfrak{c}_n$ are solvable.*

Experimental results. 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. [15]. (1) BBeye: the OCaml prover from Park et al. based upon nested sequents [15]; (2) Naive (Vamp): translates a

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

Table 1. Initial experimental results.

BBI formula into a first-order formula using the standard translation, then uses Vampire 2.6 [6] to solve it; (3) $FVLS_{BBI}$ Heuristic: backward proof search in $FVLS_{BBI}$, using the heuristic-based method to solve the set of constraints.

The results are in Table 1. In the BBeye (opt) column, 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 the latter is not stable. When the tested formulae involve more interaction between structural rules, 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 [15] 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.

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

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

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

Fig. 6. Some auxiliary structural rules.

7 Conclusions, extensions and further 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 now resides in guessing which relational atom 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 already 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. Although we do not have a counter-model construction procedure for our labelled systems, this aspect has been studied by Larchey-Wendling using labelled tableaux [8]. The possibility to adapt his method to BBI_{ND} using our calculus is also a future work.

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 $(w_{11}, w_{12} \rhd w_{13}) \wedge \cdots \wedge (w_{i1}, w_{i2} \rhd w_{i3}) \Rightarrow (w'_{11}, w'_{12} \rhd w'_{13}) \wedge \cdots \wedge (w'_{j1}, w'_{j2} \rhd w'_{j3}) \wedge (x_{11} = x_{12}) \wedge \cdots \wedge (x_{k1} = x_{k2})$, we create a rule:

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

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., $(a, b \rhd c) \wedge (a, b \rhd d) \Rightarrow (c = d)$; (2) TD-semantics: the composition of any two elements is always defined as a singleton, i.e., $\forall a, b, \exists c$ s.t. $(a, b \rhd c)$; (3) indivisible unit: (cf. Section 1) $(a, b \rhd \epsilon) \Rightarrow (a = \epsilon) \wedge (b = \epsilon)$; and (4) cancellative: 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 rules P, T, IU, C respectively in Figure 6.

The formula $(F * F) \to F$, where $F = \neg(\top \mathrel{-\!\!*} \neg\top^*)$, differentiates BBI_{ND} and BBI_{PD} [10] and is provable using $LS_{BBI}{+}P$. Using $LS_{BBI}{+}T$, we can prove $(\neg\top^* \mathrel{-\!\!*} \bot) \to \top^*$, which is valid in BBI_{TD} but not in BBI_{PD} [10], and also $(\top^* \wedge ((p * q) \mathrel{-\!\!*} \bot)) \to ((p \mathrel{-\!\!*} \bot) \vee (q \mathrel{-\!\!*} \bot))$, which is valid in separation models iff the composition is total [4]. These additional rules preserve cut-elimination.

Oddly, the formula $\neg(\top^* \wedge A \wedge (B * \neg(C \mathrel{-\!\!*} (\top^* \to A))))$, 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 tableaux (this formula must also be valid in BBI_{PD}). We are currently investigating this phenomenon.

References

1. J. Brotherston. A unified display proof theory for bunched logic. *ENTCS*, 265:197–211, September 2010.
2. J. Brotherston and C. Calcagno. Classical BI: Its semantics and proof theory. *LMCS*, 6(3), 2010.
3. J. Brotherston and M. Kanovich. Undecidability of propositional separation logic and its neighbours. In *LICS*, pages 130–139, 2010.
4. J. Brotherston and M. Kanovich. Undecidability of propositional separation logic and its neighbours. *submitted to the Journal of ACM*, 2013.
5. D. Galmiche and D. Larchey-Wendling. Expressivity properties of boolean BI through relational models. In *FSTTCS*, pages 358–369, 2006.
6. K. Hoder and A. Voronkov. Comparing unification algorithms in first-order theorem proving. KI'09, pages 435–443. Springer-Verlag, 2009.
7. Z. Hóu, A. Tiu, and R. Goré. A labelled sequent calculus for BBI: Proof theory and proof search. *arXiv:1302.4783*, 2013.
8. D. Larchey-Wendling. The formal strong completeness of partial monoidal boolean BI. *submitted to the Journal of Logic and Computation*, 2012.
9. D. Larchey-Wendling and D. Galmiche. Exploring the relation between intuitionistic BI and boolean BI: An unexpected embedding. *MSCS*, 19(3):435–500, 2009.
10. D. Larchey-Wendling and D. Galmiche. The undecidability of boolean BI through phase semantics. *LICS*, 0:140–149, 2010.
11. D. Larchey-Wendling and D. Galmiche. Non-deterministic phase semantics and the undecidability of boolean BI. *ACM TOCL*, 14(1), 2013.
12. S. Negri. Proof analysis in modal logic. *JPL*, 34(5-6):507–544, 2005.
13. S. Negri and J. von Plato. *Structural Proof Theory*. CUP, 2001.
14. P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *BSL*, 5(2):215–244, 1999.
15. J. Park, J. Seo, and S. Park. A theorem prover for boolean BI. In *POPL*, POPL '13, pages 219–232, New York, NY, USA, 2013. ACM.
16. D. J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series. Kluwer Academic Publishers, 2002.
17. J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, LICS '02, pages 55–74. IEEE Computer Society, 2002.