

[Electronic Notes in Theoretical Computer Science 265 (2010) 197–211](http://dx.doi.org/10.1016/j.entcs.2010.08.012)

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

A Unified Display Proof Theory for Bunched Logic

# James Brotherston[1](#_bookmark0)

*Dept. of Computing Imperial College London, UK*

Abstract

We formulate a unified display calculus proof theory for the four principal varieties of bunched logic by combining display calculi for their component logics. Our calculi satisfy cut-elimination, and are sound and complete with respect to their standard presentations. We show that the standard sequent calculus for BI can be seen as a reformulation of its display calculus, and argue that analogous sequent calculi for the other varieties of bunched logic seem very unlikely to exist.

*Keywords:* bunched logic, display calculus, proof theory, cut-elimination, substructural logic

# Introduction

*Bunched logics*, originating in O’Hearn and Pym’s BI [[18](#_bookmark44)], are a relatively recent addition to the menagerie of substructural logics and are increasingly attracting interest amongst computer science researchers as well as logicians. Of their better- established cousins, bunched logics most resemble *relevant logics* [[21](#_bookmark45)] in that they feature both multiplicative (or intensional) and additive (or extensional) logical connectives, with the difference between the two types characterised as a matter of which structural principles are admitted by each. However, while in relevant logics certain of the additive connectives are barred in order to exclude various philosoph- ically controversial theorems from the logic, in bunched logics one simply takes a *complete* set of additive connectives in addition to a set of multiplicatives. Thus bunched logics can be seen as the result of freely combining a standard (additive) propositional logic with a (multiplicative) linear logic. This simple-minded treat- ment of the additives gives rise to a Tarskian *resource interpretation* of formulas

1 Research supported by an EPSRC Postdoctoral Fellowship. Thanks to Rajeev Gor´e, Greg Restall and the East London Massive (as was) for illuminating discussions and advice, and to the anonymous referees for suggestions on improving the paper.

1571-0661 © 2010 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2010.08.012

in bunched logics, which are read as true or false relative to resources: roughly speaking, the additives have their standard propositional meanings while the mul- tiplicatives denote resource composition properties [[20](#_bookmark46)]. In computer science, such resource readings of bunched logic have very successfully been exploited to obtain logics for program analysis. Most notably, *separation logic* [[24](#_bookmark50)], which is based upon an interpretation of resources as portions of heap memory, has spawned a host of program analysis applications that discover and reason about the structure of heap memory during program execution (recent examples include [[7](#_bookmark33),[8](#_bookmark34),[10](#_bookmark36)]). Bunched logic has also been variously employed in addressing other computing problems such as polymorphic abstraction [[9](#_bookmark35)], tree update [[6](#_bookmark32)] and typed reference update and dis- posal [[2](#_bookmark27)].

In this paper, we examine bunched logic from the general proof-theoretic per- spective. While there has been considerable interest in the semantics of bunched logics, justified in no small part by the computational significance of the result- ing models [[11](#_bookmark37),[12](#_bookmark38),[14](#_bookmark40),[20](#_bookmark46)], their proof theory by contrast has received comparatively little attention. As observed by Pym [[19](#_bookmark47)], it is natural to consider four principal varieties of bunched logic, characterised by the presence or otherwise of classical negation in the additive and multiplicative fragments or, equivalently, by the un- derlying additive and multiplicative algebras (see Figure [1](#_bookmark2)). However, to date there has been no proof-theoretical analysis corresponding to this general characterisa- tion. On the one hand, O’Hearn and Pym’s original bunched logic BI is known to possess both a complete natural deduction proof system satisfying normalisation, and a complete sequent calculus satisfying cut-elimination [[19](#_bookmark47)]. On the other hand, analogues of these syntactic proof systems for well-known variants of BI such as Boolean BI (BBI) have been conspicuously absent from the literature [2](#_bookmark1) . Instead, proof systems for BBI are usually obtained in a crude manner by adding a suffi- ciently powerful axiom or inference rule to the corresponding proof system for BI. Such additions typically break normalisation and cut-elimination properties, which is less than ideal from the theoretical point of view but also from a practical per- spective, since separation logic and many of the aforementioned related program analysis tools are based on BBI. However, extending the BI sequent calculus to BBI without breaking cut-elimination is highly problematic.

An alternative route to a disciplined proof theory for general bunched logics is suggested by Belnap’s *display logic* [[1](#_bookmark28)], which was historically employed as a device

for giving consecution calculi *a` la* Gentzen for relevant and modal logics. The

distinguishing feature of display calculi is the *display property* : any consecution can always be rearranged so that a given part appears alone on the appropriate side of the proof turnstile. To ensure this property we need both a richer form of consecution than that of typical Gentzen-style sequents, and a set of auxiliary “display” rules for rearranging them in the required fashion. Compensation for this extra complexity comes in the form of an elegant, symmetric presentation of the calculus, analogous to that of Gentzen’s sequent calculi. Furthermore, cut-

2 A tableau proof system for a variant of BBI based on functional rather than the usual relational monoidal semantics is given in [[16](#_bookmark42)], but it does not correspond directly to a cut-free sequent calculus, owing to its reliance on semantic constraint labels.

CBI

(Boolean, de Morgan)

*undecidable* [[4](#_bookmark30)]

¬

~

dMBI

(Heyting, de Morgan)

BBI

(Boolean, Lambek)

*undecidable* [[4](#_bookmark30),[17](#_bookmark43)]

~

¬

BI

(Heyting, Lambek)

*decidable* [[11](#_bookmark37)]

Fig. 1. The bunched logic family. The (additive, multiplicative) subtitles indicate the underlying additive

and multiplicative algebras. The arrows denote the addition of either additive (¬) or multiplicative (∼) classical negation.

elimination is guaranteed for any display calculus whose rules obey a set of easily verifiable syntactic conditions.

In earlier work with Calcagno, we showed that the “classical” bunched logic CBI could be naturally presented as a display calculus [[3](#_bookmark29)]. In this paper, we obtain a *uniﬁed* display calculus proof theory for all four principal bunched logics in Figure [1](#_bookmark2). First, we formulate display calculi for the elementary logics which characterise the additive and multiplicative components of the various logics. Since Belnap’s orig- inal display apparatus does not adapt to the intuitionistic components (because it relies on the presence of classical negation), we instead exploit the residual re- lationship between conjunction and implication to obtain a display property, *`a la* Gor´e [[13](#_bookmark39)] and Restall [[23](#_bookmark48)]. Then, we obtain display calculi for the bunched logics by combining the display calculi for the corresponding elementary components. This composition of the elementary calculi preserves several of their desirable structural properties, chiefly cut-elimination and soundness / completeness with respect to standard presentations of the corresponding logics. Additionally, we establish a translation between cut-free proofs in our display calculus for BI and those in its standard sequent calculus, showing that the latter system can be seen as a refined version of the former. The fact that our display calculi for the other bunched logics seemingly cannot be similarly optimised into sequent presentations — due to their essential use of *unary* structural connectives as well as the usual binary ones — goes some way to explaining why well-behaved sequent calculi for these logics have been so elusive.

The remainder of this paper is organised as follows. In Section [2](#_bookmark3) we define the four main bunched logics in Figure [1](#_bookmark2) in terms of their elementary component logics. Section [3](#_bookmark8) introduces the apparatus of display logic. In Section [4](#_bookmark12) we give display calculi for the bunched logics via display calculi for their elementary component logics. In Section [5](#_bookmark20) we compare the resulting display calculus for BI with its bunched sequent calculus. Section [6](#_bookmark25) concludes.

# From elementary logics to bunched logics

In this section, we define the four principal bunched logics (cf. Figure [1](#_bookmark2)) as free combinations of well-known elementary logics.

We assume a fixed infinite set u of propositional variables. *Formulas* are con- structed from propositional variables using the logical connectives given in Figure [2](#_bookmark4): any *P* ∈ u is a formula, and so is the result of applying a logical connective to the appropriate number of formulas. We restrict the syntax of formulas in a particular logic by stipulating which formula connectives are permitted to occur. We write *F, G, H*, etc., to range over formulas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Additive symbol:*  *Multiplicative symbol:* | T  T∗ | ⊥  ⊥∗ | ¬  ~ | ∧  ∗ | ∨  ∨∗ | →  —∗ |
| *Arity:* | 0 | 0 | 1 | 2 | 2 | 2 |

Fig. 2. Logical connectives.

We regard a *logic* L as being specified by: (a) the set of logical connectives which may occur in formulas of the logic; and (b) a basic proof system for entailments of the form *F* ▶ *G*, where *F* and *G* are formulas. We write *F* E▶ *G* to abbreviate two axioms *F* ▶ *G* and *G* ▶ *F* . We specify four well-known elementary logics, which form the principal components of the bunched logics in Figure [1](#_bookmark2), as follows:

* + Intuitionistic logic, IL, has as logical connectives T, ⊥, Λ, V and →. Classical logic, CL, adds the negation ч. We present IL and CL in Figure [3](#_bookmark5).
  + Lambek multiplicative logic, LM (a.k.a. multiplicative intuitionistic linear logic), has as logical connectives the multiplicative T∗, ∗ and —∗. De Morgan multiplica- tive logic, dMM (a.k.a. multiplicative classical linear logic), extends these by ⊥∗,

~ and V∗ . We present LM and dMM in Figure [4](#_bookmark6).

*F* ▶ *F F* ▶ T ⊥ ▶ *F Gi* ▶ *G*1 ∨ *G*2 (*i* ∈ {1*,* 2}) *G*1 ∧ *G*2 ▶ *Gi* (*i* ∈ {1*,* 2})

*F* ▶ *H G* ▶ *H*

*F* ▶ *G F* ▶ *H*

*F* ▶ *G* → *H*

*F* ∧ *G* ▶ *H*

*F* ▶ *G G* ▶ *H*

*F* ∨ *G* ▶ *H*

*F* ▶ *G* ∧ *H*

*F* ∧ *G* ▶ *H*

*F* ▶ *G* → *H*

*F* ▶ *H*

............ .................... ................... .................... .................... .......

¬*F* E▶ *F* → ⊥ ¬¬*F* ▶ *F*

Fig. 3. IL and CL. The axioms below the dotted line are present in CL only.

*F* ▶ *F F* ∗ (*G* ∗ *H*) E▶ (*F* ∗ *G*) ∗ *H F* ∗ *G* ▶ *G* ∗ *F F* ∗ T∗ E▶ *F*

*F*1 ▶ *G*1 *F*2 ▶ *G*2

*F* ∗ *G* ▶ *H*

*F* ▶ *G* —∗ *H*

*F* ▶ *G G* ▶ *H*

*F*1 ∗ *F*2 ▶ *G*1 ∗ *G*2

*F* ▶ *G* —∗ *H*

*F* ∗ *G* ▶ *H*

*F* ▶ *H*

. . ........... .................... .................... ................... ................

⊥∗ E▶ ∼T∗ *F* ∨∗ *G* E▶ ∼(∼*F* ∗ ∼*G*) ∼*F* E▶ *F* —∗ ⊥∗ ∼∼*F* ▶ *F*

Fig. 4. LM and dMM. The axioms below the line are present in dMM only.

We write S = {IL*,* CL*,* LM*,* dMM} for this set of elementary logics. By the *free combination* L1 + L2 of two logics L1*,* L2 ∈ S, we mean the logic whose logical connectives and presentation are the unions of, respectively, the logical connectives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Additive* | *Multiplicative* | *Arity* | *Antecedent meaning* | *Consequent meaning* |
| ∅ | ∅ | 0 | truth | falsity |
|  | *b* | 1 | negation | negation |
| ; | *,* | 2 | conjunction | disjunction |
| ⇒ | ( | 2 | undefined | implication |

Fig. 5. Structural connectives.

and the presentations of L1 and L2. The bunched logics B = {BI*,* BBI*,* dMBI*,* CBI} can then be defined very straightforwardly in terms of their elementary components:

* + - BI, a.k.a. the logic of bunched implications (cf. [[18](#_bookmark44),[20](#_bookmark46)]), is given by IL + LM;
    - BBI, a.k.a. Boolean BI (cf. [[12](#_bookmark38)]), is given by CL + LM;
    - dMBI, standing for “de Morgan BI”, is given by IL + dMM;
    - CBI, a.k.a. Classical BI (cf. [[3](#_bookmark29)]), is given by CL + dMM.

Our definition of the logics S ∪ B above will be taken as the baseline with respect to which our display calculi for these logics are later proven correct. This has the benefit of freeing our analysis from unnecessary semantic considerations. We note that our definitions of B can be seen to be in agreement with those found elsewhere in the literature, as well as the characterisation in Figure [1](#_bookmark2). For example, our presentations of BI and BBI agree with their counterparts in [[20](#_bookmark46)] and [[12](#_bookmark38)] respectively. (To our knowledge, dMBI has not appeared in the literature before,

while CBI was presented in [[3](#_bookmark29)] via a display calculus, which we will reconstruct as part of our unified proof theory for B.)

# Display calculus fundamentals

In this section we present the basic notions that we require in order to specify a

*display calculus* in the spirit of Belnap [[1](#_bookmark28)].

*Structures* are constructed from formulas using the structural connectives given by Figure [5](#_bookmark7): any formula is a structure, and so is the result of applying a structural connective to the appropriate number of structures. We write *W, X, Y, Z*, etc., to range over structures. If *X* and *Y* are structures then *X* ▶ *Y* is called a *consecution*.

There is a classification of the structure occurrences in a consecution into *antecedent*

and *consequent* parts, which generalises the simple left-right division of sequent calculus.

Definition 3.1 (Antecedent / consequent part) *Substructure occurrences in a structure X are classiﬁed as either* positive *or* negative *in X, as follows:*

* + - *X is positive in X;*
    - *if Z is negative (positive) in Y then Z is positive (negative) in Y and bY ;*
    - *if Z is positive (negative) in X*1 *or X*2 *then Z is positive (negative) in X*1 ; *X*2

*and X*1 *, X*2*;*

* + - *if Z is negative (positive) in X*1 *or positive (negative) in X*2*, then Z is positive (negative) in X*1 ⇒ *X*2 *and X*1 ( *X*2*.*

*Z is said to be an* antecedent (consequent) part *of a consecution X* ▶ *Y if it is positive (negative) in X or negative (positive) in Y .*

Consecutions are interpreted as entailments between formulas as follows.

Definition 3.2 (Consecution validity) *For any structure X we deﬁne the for- mulas* Ψ*X and* Υ*Y by mutual structural induction as follows:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ψ*F*  Ψ∅  Ψ *X* | = *F*  = T  = чΥ*X* | Ψ∅ Ψ*bX* | = T∗  = ~Υ*X* | Υ*F*  Υ∅  Υ *X* | = *F*  = ⊥  = чΨ*X* | Υ∅ Υ*bX* | = ⊥∗  = ~Ψ*X* |
| Ψ*X*;*Y* | = Ψ*X* Λ Ψ*Y* | Ψ*X,Y* | = Ψ*X* ∗ Ψ*Y* | Υ*X*;*Y* | = Υ*X* V Υ*Y* | Υ*X,Y* | = Υ*X* V∗ Υ*Y* |
| Ψ*X*⇒*Y* | = *undefined* | Ψ*X*(*Y* | = *undefined* | Υ*X*⇒*Y* | = Ψ*X* → Υ*Y* | Υ*X*(*Y* | = Ψ*X —*∗ Υ*Y* |

*X* ▶ *Y is said to be* valid *in a logic* L *iff* Ψ*X* ▶ Υ*Y is provable in* L*.*

We remark that, in any given display calculus, we restrict the form of conse- cutions by stipulating which of the structural connectives may appear as the main (i.e. outermost) connective of an antecedent or consequent part. In doing so, we en- sure that the restrictions on the structural connectives match the available formula connectives, so that validity of consecutions is always well defined. In particular, neither ⇒ nor ( will ever be permitted to appear as the main connective of an

antecedent part of a consecution.

The defining feature of any true display calculus is the availability of an equiv- alence relation on consecutions, called a *display-equivalence*, that facilitates their rearrangement into an equivalent consecution in which a given antecedent (conse- quent) part appears as the *entire* antecedent (consequent).

Definition 3.3 (Display-equivalence) *Let* ≡*D be the least equivalence generated by a set of rules of the form C <>D C*j*, where C, C*j *are consecutions. We say that*

≡*D is a* display-equivalence *if, for any antecedent (consequent) part Z of X* ▶ *Y , one can construct a structure W such that X* ▶ *Y* ≡*D Z* ▶ *W (X* ▶ *Y* ≡*D W* ▶ *Z). The process of rearranging X* ▶ *Y into Z* ▶ *W or W* ▶ *Z via display-equivalence is called* displaying *Z.*

A *display calculus* DLL for a logic L is then specified by the following:

Antecedent / consequent structural connectives: The structural connec- tives that are permitted to appear as the main connective of an antecedent / consequent part of a consecution, respectively.

Display postulates: A set of rules of the form *C <>D C*j such that the least equivalence ≡*D* generated by the rules is a display-equivalence.

Logical rules: Proof rules for the formula connectives, typically divided into pairs of left- and right-introduction rules for each logical connective in the manner familiar from sequent calculus (though, like in sequent systems, some connectives may have only one introduction rule). Note that, since we can appeal to the

display-equivalence ≡*D*, these rules may be written so that the formula introduced by a rule is displayed (alone) in its conclusion.

Structural rules: Proof rules for the structural connectives. We write a rule with a double-line between premise and conclusion to indicate that it is *symmetric*,

Antecedent structure connectives: ∅ ;

Consequent structure connectives: ⇒

Display postulates: *X* ; *Y* ▶ *Z <>D X* ▶ *Y* ⇒ *Z <>D Y* ; *X* ▶ *Z*

Logical rules:

⊥ ▶ *X*

(⊥L)

∅ ▶ *X* T ▶ *X*

(TL)

*F* ; *G* ▶ *X F* Λ *G* ▶ *X*

(ΛL)

*F* ▶ *X G* ▶ *X F* V *G* ▶ *X*

(VL)

*X* ▶ *F G* ▶ *Y F* → *G* ▶ *X* ⇒ *Y*

(→L)

(TR)

*X* ▶ *F X* ▶ *G*

(ΛR)

*X* ▶ *Fi i* ∈ {1*,* 2}

*X* ; *F* ▶ *G*

(→R)

*X* ▶ T

Structural rules:

∅ ; *X* ▶ *Y*

*X* ▶ *F* Λ *G*

*W* ; (*X* ; *Y* ) ▶ *Z*

*X* ▶ *F*1 V *F*2

*X* ▶ *Z*

(VR)

*X* ▶ *F* → *G*

*X* ; *X* ▶ *Y*

======= (∅L)

*X* ▶ *Y*

============= (AAL) (*W* ; *X*); *Y* ▶ *Z*

*X* ; *Y* ▶ *Z*

(WkL) (CtrL)

*X* ▶ *Y*

Fig. 6. Specification of DLIL.

i.e., that the premise and conclusion may be exchanged.

In addition to the logical and structural proof rules given by their specification, all of our display calculi share a common set of *identity rules*:

*P* ▶ *P*

(Id)

*X* ▶ *F F* ▶ *Y X* ▶ *Y*

(Cut)

*X*j ▶ *Y* j

*X* ▶ *Y*

*X* ▶ *Y* ≡*D X*j ▶ *Y* j (≡*D*)

where *P* ranges over propositional variables. We remark that a display calculus so specified is not guaranteed to obey any particular proof-theoretic properties over and above the availability of display-equivalence; as is well-known, display calculi may fail to enjoy cut-elimination, interpolation, or decidability. However, cut-elimination is guaranteed for display calculi with sufficiently well-behaved logical and structural rules [[1](#_bookmark28)].

# Display calculi for bunched logics

In this section we give display calculi for the elementary logics S (see Section [2](#_bookmark3)) and combine them to obtain display calculi for the bunched logics ¥. We give display calculus specifications for the elementary logics IL, CL, LM and dMM in Figures [6](#_bookmark11), [7](#_bookmark13), [8](#_bookmark14) and [9](#_bookmark16) respectively.

Some remarks on our formulation of these elementary display calculi are in order. Firstly, the display postulates for the classical logics CL and dMM essentially follow Belnap [[1](#_bookmark28)]. These postulates do not adapt to the intuitionistic IL and LM because they lack suitable involutive negations necessary to interpretand *b*. Instead, we employ the meta-level implications ⇒ and (, and exploit their residual connections

with the conjunctions to obtain suitable display postulates (an idea also employed

by Gor´e [[13](#_bookmark39)] and Restall [[23](#_bookmark48)]).

Secondly, because the structural connectives ∅ and ‘;’ may occur only in an- tecedent positions in DLIL consecutions, we must use structure-free formulations of the rules (VL) and (⊥L) (for convenience, we do the same for (ΛR) and (TR), and use the same rules for DLCL). Also, we could have written a single version of (→L)

Antecedent structure connectives: ∅ ;

Consequent structure connectives: ∅ ;

Display postulates: *X* ; *Y* ▶ *Z <>D X* ▶ *Y* ; *Z <>D Y* ; *X* ▶ *Z*

*X* ▶ *Y* ; *Z <>D X* ; *Y* ▶ *Z <>D X* ▶ *Z* ; *Y*

*X* ▶ *Y <>D Y* ▶ *X <>D X* ▶ *Y*

Logical rules:

Structural rules:

*X* ▶ ∅

*X* ▶ ⊥

(⊥R)

*F* ▶ *X*

ч*F* ▶ *X*

(чL)

*X* ▶ *F X* ▶ ч*F*

(чR)

*X* ▶ *Y* ; ∅

======= (∅R)

*X* ▶ *Y*

*X* ▶ *F G* ▶ *Y*

*F* → *G* ▶ *X* ; *Y*

*X* ▶ *F* ; *G*

(→L) (VR)

*X* ▶ *F* V *G*

Fig. 7. Specification of DLCL. The logical and structural rules extend those of DLIL (Figure [6](#_bookmark11)), except for (VR) and (→L) which replace their DLIL counterparts.

Antecedent structure connectives: ∅ *,*

Consequent structure connectives: (

Display postulates: *X, Y* ▶ *Z <>**D X* ▶ *Y* ( *Z <>D Y, X* ▶ *Z*

Logical rules:

Structural rules:

∅ ▶ *X*

T∗ ▶ *X*

(T∗L)

*F , G* ▶ *X*

*F* ∗ *G* ▶ *X*

(∗L)

*X* ▶ *F G* ▶ *Y F* —∗ *G* ▶ *X* ( *Y*

(—∗L)

∅ *, X* ▶ *Y*

======== (∅L)

*X* ▶ *Y*

∅ ▶ T∗

(T∗R)

*X* ▶ *F Y* ▶ *G*

*X , Y* ▶ *F* ∗ *G*

*X , F* ▶ *G*

(∗R) (—∗R)

*X* ▶ *F* —∗ *G*

Fig. 8. Specification of DLLM.

*W ,* (*X , Y* ) ▶ *Z*

============= (MAL)

(*W , X*) *, Y* ▶ *Z*

common to both DLIL and DLCL with conclusion *X* ; *F* → *G* ▶ *Y* , and similarly for (—∗L) in DLLM and DLdMM. We use separate versions in order to maintain the pleasant property that the formula introduced by a logical rule is always displayed in its conclusion.

Finally, we note that the right-hand analogues of the structural rules (AAL), (CtrL) and (WkL) are derivable in DLCL. Likewise, the right-hand analogue of (MAL) is derivable in DLdMM.

Now we obtain display calculi for ¥ by defining, for L1*,* L2 ∈ S:

DLL1+L2 =def DLL1 + DLL2

where DLL1 + DLL2 is the display calculus whose antecedent and consequent struc- ture connectives, display postulates, and logical and structural rules are in each case those of DLL1 plus those of DLL2 . We observe that DLCBI as presented here is equivalent to its earlier formulation in [[3](#_bookmark29)], while DLBI, DLBBI and DLdMBI are new. However, DLdMBI can be seen to be very nearly equivalent to Restall’s display calcu- lus for the well-known relevant logic RW (a.k.a. C) obtained from R by removing the multiplicative contraction rule [[23](#_bookmark48)]. The two calculi differ only because RW

lacks the additive intuitionistic → and ⊥ of dMBI (which can however be added conservatively).

We now demonstrate that each of our specifications does indeed give rise to a true display calculus, in the sense that the display property holds.

Antecedent structure connectives: ∅ *b ,*

Consequent structure connectives: ∅ *b ,*

Display postulates: *X , Y* ▶ *7 <>D X* ▶ *bY , 7 <>D Y , X* ▶ *7*

*X* ▶ *Y , 7 <>D X , bY* ▶ *7 <>D X* ▶ *7 , Y*

*X* ▶ *Y <>D bY* ▶ *bX <>D bbX* ▶ *Y*

Logical rules:

Structural rules:

⊥∗ ▶ ∅

(⊥∗L)

*bF* ▶ *X*

~*F* ▶ *X*

(~L)

*F* ▶ *X t* ▶ *Y F* V∗ *t* ▶ *X , Y*

(V∗ L)

*X* ▶ *F t* ▶ *Y F* —∗ *t* ▶ *bX , Y*

(—∗L)

*X* ▶ *Y ,* ∅

======== (∅R)

*X* ▶ *Y*

*X* ▶ ∅

*X* ▶ ⊥∗

(⊥∗R)

*X* ▶ *bF X* ▶ ~*F*

(~R)

*X* ▶ *F , t X* ▶ *F* V∗ *t*

(V∗ R)

Fig. 9. Specification of DLdMM. The logical and structural rules extend those of DLLM above, except for

(—∗L) which replaces its DLLM counterpart.

Proposition 4.1 (Display) *For all* L ∈ S ∪ ¥*, the least equivalence* ≡*D induced by the display postulates of* DLL *is a display-equivalence for* DLL*.*

Proof. (Sketch) The required display property (Defn. [3.3](#_bookmark10)) follows from the fact that, for any consecution *X* ▶ *Y* , the display postulates of DLL facilitate the display of each of the immediate substructures of *X* and *Y* (as the antecedent or consequent as appropriate); it follows by induction that arbitrary substructures of *X* ▶ *Y* can be displayed. This fact may be verified essentially by eye for each L ∈ S, noting that the structure of consecutions and display postulates in DLIL and DLCL is isomorphic to that of DLLM and DLdMM respectively. It follows immediately that the immediate substructures of *X* ▶ *Y* can be displayed for each L ∈ ¥, using the display postulates from the appropriate component calculus. 2

Proposition 4.2 (Soundness) *For all* L ∈ S ∪ ¥*, if a consecution of* DLL *is provable in* DLL *then it is valid.*

Proof. (Sketch) It suffices to prove that each rule of DLL is locally sound in that, if all of its premises are valid, then so is its conclusion. In practice this means deriving the rule in L under the translation from consecutions to formula entailments given by Defn. [3.2](#_bookmark9). This may straightforwardly, if somewhat tediously, be carried out for each of the elementary logics L ∈ S. For L ∈ ¥, the local soundness property is then immediate since it is obvious that, if any DLL*i* rule is L*i*-derivable under translation for *i* ∈ {1*,* 2}, then the rules of DLL1 + DLL2 are derivable under translation in L1 + L2. 2

Lemma 4.3 (Identity) *For all* L ∈ S ∪ ¥*, and for any formula F of* L*, the con-* *secution F* ▶ *F is provable in* DLL*.*

Proof. (Sketch) By structural induction on *F* , distinguishing a case for every pos- sible logical connective of L. In the case *F* = *P* ∈ u we are immediately done by (Id). The connective cases follow straightforwardly by applying the left- and right-introduction rules for the connective together with the induction hypothesis

(for the additives the structural rules are also required). 2

Lemma 4.4 *For all* L ∈ S ∪ ¥*, and for any consecution X* ▶ *Y of* DLL*, the consecutions X* ▶ Ψ*X and* Υ*Y* ▶ *Y are* DLL*-provable, where* Ψ— *and* Υ— *are the functions given in Deﬁnition* [*3.2*](#_bookmark9)*.*

Proof. (Sketch) By induction on the number of structural connectives in *X* ▶ *Y* , distinguishing a case for every possible structural connective of *X* and *Y* in DLL. In the base case *X* and *Y* are formulas of L, and we are done by Lemma [4.3](#_bookmark17). We show a typical connective case, *Y* = (*X*j ⇒ *Y* j), in which case DLL contains DLIL and we have Υ*Y* = Ψ*X*′ → Υ*Y* ′ . As *X*j ⇒ *Y* j is a consequent part of *X* ▶ *Y* by the case assumption, *X*j is an antecedent part and *Y* j a consequent part, so *X*j ▶ *Y* j is a consecution of DLL and thus by induction hypothesis *X*j ▶ Ψ*X*′ and Υ*Y* ′ ▶ *Y* j are both DLL-provable. Thus by applying the DLIL rule (→L) we obtain the required DLL proof of Ψ*X*′ → Υ*Y* ′ ▶ *X*j ⇒ *Y* j. 2

Theorem 4.5 (Completeness) *For all* L ∈ S ∪ ¥*, if a consecution of* DLL *is valid then it is provable in* DLL*.*

Proof. (Sketch) Suppose that *X* ▶ *Y* is DLL-valid, i.e. that Ψ*X* ▶ Υ*Y* is L-provable. To show that *X* ▶ *Y* is DLL-provable, it suffices by Lemma [4.4](#_bookmark18) and (Cut) to show that Ψ*X* ▶ Υ*Y* is DLL-provable. In practice this simply entails showing that each proof rule of L is DLL-derivable, which is a routine exercise for each L ∈ S. The result then follows immediately for all L ∈ ¥ because it is clear that, if DLL1 and DLL2 can derive every rule of L1 and L2 respectively, then DLL1 + DLL2 can derive every rule of L1 + L2. 2

We call a display calculus proof *cut-free* if it contains no instances of (Cut).

Theorem 4.6 (Cut-elimination) *For all* L ∈ S ∪ ¥*, any* DLL *proof of X* ▶ *Y*

*can be transformed into a cut-free proof of X* ▶ *Y .*

Proof. (Sketch) Given the display property (Proposition [4.1](#_bookmark15)), it suffices to verify for each L ∈ S ∪ ¥ that the proof rules of DLL meet Belnap’s well-known condi- tions C1–C8 guaranteeing cut-elimination [[1](#_bookmark28)]. Moreover, since conditions C1–C7 are properties of individual rules rather than sets of rules, these may be verified for

all L ∈ S ∪ ¥ in one pass, simply by examining each rule appearing in Figures [6](#_bookmark11)–[9](#_bookmark16). Condition C8 (elimination of principal cuts) depends essentially on the form of the logical rule pairs (•L) and (•R) for each logical connective • (though the structural rules are also sometimes required to eliminate principal cuts in DLIL and DLCL). C8 may be easily verified for each L ∈ S, and then follows immediately for each L ∈ ¥ since every principal cut in DLL is a principal cut in one of its elementary component calculi. 2

We remark that cut-free proofs in our display calculi enjoy a traditional subfor- mula property, as can be seen by inspection of the proof rules. However, the anal- ogous “substructure” property does not hold: the premise of a rule instance may contain structures which are not substructures of any structure in the conclusion [3](#_bookmark19) .

3 Of course exactly the same is true, e.g., of any sequent calculus with a contraction rule.

Thus, like in linear logic, cut-elimination does not necessarily entail decidability or interpolation. Indeed, in collaboration with Kanovich we have recently shown both BBI and CBI, among other formalisms, to be undecidable [[4](#_bookmark30)] [4](#_bookmark22) . Cut-elimination in display calculi remains meaningful for the reason that it eliminates the infinite branching points provided by the cut rule; as is clear by inspection of the proof rules, for any consecution there are only finitely many ways of applying a proof rule backwards [5](#_bookmark23) . Thus an exhaustive proof search is finitely branching, albeit possibly non-terminating. Moreover, for certain display calculi it is possible to ensure that such a proof search is indeed terminating [[23](#_bookmark48)] — although it is sadly impossible to generally determine which ones [[15](#_bookmark41)].

# Relationship to bunched sequent calculi

Of all the bunched logics ¥, only BI is known to possess a sequent calculus with cut-elimination, given by Pym [[19](#_bookmark47)]. Thus it is natural to compare this calculus, LBI, with our display calculus DLBI. The sequents of LBI are of the form Γ ▶ *F* where *F* is a BI-formula and Γ is a *bunch*, given by:

Γ ::= *F* | ∅ | ∅ | Γ; Γ | Γ *,* Γ

where *F* ranges over BI-formulas. A *coherent equivalence* ≡ is defined on bunches as the least congruence closed under associativity and commutativity of the semicolon and comma, and under the equations (∅ ; *F* ) ≡ *F* ≡ (∅ *, F* ). The right-introduction rules for the logical connectives have standard intuitionistic formulations. The left- introduction rules, and the structural rules, are written so as to apply to formulas occurring at arbitrary positions within a bunch, using the notation Γ(Δ) for a bunch Γ with a distinguished sub-bunch occurrence Δ. For example, the rules for —∗ in LBI are:

Δ ▶ *F*1 Γ(*F*2) ▶ *F*

Γ(Δ *, F*1 —∗ *F*2) ▶ *F*

(—∗L)

Γ *, F* ▶ *G*

Γ ▶ *F* —∗ *G*

(—∗R)

We demonstrate a correspondence between cut-free proofs in LBI and in DLBI.

Lemma 5.1 *There is an injective, constructive map from* LBI *proofs to* DLBI

*proofs. Moreover, this map preserves cut-freeness of proofs.*

Proof. (Sketch) First note that every LBI sequent is a DLBI consecution, as bunches are exactly the structures that can occur as antecedent parts of the lat- ter. We show that each of the proof rules of LBI is derivable in DLBI. The left- introduction rules can be seen in DLBI as a macro for first displaying the active part of the conclusion, then applying the corresponding left-introduction rule of DLBI and finally reversing the original display process to restore the bunch context. E.g.,

4 An alternative proof of the undecidability of BBI was produced independently [[17](#_bookmark43)].

5 Provided that one is careful to prevent arbitrary “stacking” ofand *b*, cf. [[15](#_bookmark41),[23](#_bookmark48)].

we derive the (—∗L) rule of LBI as follows:

Γ(*F*2) ▶ *F*

Δ ▶ *F*1 *F*2 ▶ *X*

(≡*D*)

Δ *, F*1 —∗ *F*2 ▶ *X*

Γ(Δ *, F*1 —∗ *F*2) ▶ *F*

(—∗L)

(≡*D*)

where *X* is a placeholder for the consequent structure that results from displaying *Z* in Γ(*Z*) ▶ *F* . The other left-rules are similar. The right-introduction rules of LBI have direct equivalents in DLBI, and applications of the rule for coherent equivalence

≡ are translated into DLBI as combinations of the display-equivalence rule (≡*D*), the associativity rules (AAL) and (MAL) and the unit rules (∅L) and (∅L). 2

Lemma 5.2 *There is a constructive map from* DLBI *proofs to* LBI *proofs. More- over, this map preserves cut-freeness of proofs.*

Proof. (Sketch) For any DLBI consecution *X* ▶ *Y* define its *display-normal form* *X* ▶ *Y* ’ to be the consecution obtained by applying the transformations (*X* ▶ *Y* ⇒ *Z*) '→ (*X* ; *Y* ▶ *Z*) and (*X* ▶ *Y* ( *Z*) '→ (*X , Y* ▶ *Z*) until no further transformations are possible. Note that for any DLBI consecution *X* ▶ *Y* we have that *X* is a bunch and *X* ▶ *Y* ’ is a unique LBI sequent of the form Γ(*X*) ▶ *F* . We show that each proof rule of DLBI is derivable in LBI under the translation −’. For example, in the case of the DLBI rule (—∗L) we have:

*X* ▶ *F* ’ *G* ▶ *Y* ’

=

*F* —∗ *G* ▶ *X* ( *Y* ’

*X* ▶ *F* Γ(*G*) ▶ *H*

Γ(*X , F* —∗ *G*) ▶ *H*

and we are immediately done since the translated rule instance is simply the (—∗L) rule of LBI. The other rules are similar. Note that for the display rule (≡*D*) we simply treat each display postulate individually: applications of display postulates either collapse under −’ or boil down to the commutativity of the comma or semicolon, which is handled by the bunch equivalence ≡. 2

Proposition 5.3 *Any cut-elimination procedure for* DLBI *may be constructively transformed into a cut-elimination procedure for* LBI*, and vice versa.*

Proof. Immediate from Lemmas [5.1](#_bookmark21) and [5.2](#_bookmark24). 2

While Lemma [5.1](#_bookmark21) demonstrates that a cut-free LBI proof is essentially a cut- free DLBI proof with some display steps omitted, Lemma [5.2](#_bookmark24) indicates the converse: any cut-free DLBI proof can be viewed as a cut-free LBI proof by first bringing each consecution into a “display-normal form”. We suggest that this normal form probably does not exist for the display calculi for the other bunched logics in any meaningful sense (and so Lemma [5.2](#_bookmark24) does not adapt), because of the seemingly es- sential presence of the structural negationsand/or *b* in these calculi. For example, if we consider the DLBBI consecution *F , G* ▶ *H*, then it is clear that there is no

display-rearrangement of this consecution which could be called “bunched-like” in

the sense thatdoes not occur in it. Thus any cut-free sequent calculus for BBI *without* such a unary negative structuring must represent cut-free DLBBI proofs in a rather non-trivial way, and it appears quite plausible that attempts to formulate such a calculus are fundamentally doomed — an observation borne out by our own experience and that of others [[19](#_bookmark47)]. Similar remarks apply to dMBI and CBI. (Of course, this does not rule out other, less syntax-directed approaches such as labelled deduction based on tableaux [[11](#_bookmark37),[16](#_bookmark42)] or hybrid logics [[22](#_bookmark49)].)

# Conclusion

Our main contribution in this paper is a unified proof theory for the principal varieties of bunched logic, based on display calculus (incidentally substantiating O’Hearn and Pym’s suggestion that this apparatus might apply to BI [[18](#_bookmark44)]). In particular, we provide the first cut-free proof system for BBI, which underlies sepa- ration and spatial logics employed in program analysis. Evidence for the canonicity of our unified proof theory is provided by cut-elimination for each of our calculi, as well as soundness and completeness with respect to the basic presentations of the corresponding logics.

The fact that each bunched logic can individually be presented as a display calculus is relatively unsurprising in the light of the earlier display calculus for CBI presented in [[3](#_bookmark29)], and the intuitionistic display technology, based on residual pairs of connectives, to be found in [[13](#_bookmark39),[23](#_bookmark48)]. As well as realising these calculi explicitly, we make two additional contributions in this paper. First, we obtain our proof theory in a unified and economical way, by first formulating and then combining calculi for the elementary additive and multiplicative components of the bunched logics. Our treatment takes advantage of the compositionality of key structural properties of display calculi: given that the properties hold for two “elementary” display calculi, it is straightforward to establish that the same properties hold of the display calculus obtained by combining them. Second, having formulated our display calculi, we are in a position to immediately establish a translation between cut-free proofs in our display calculus for BI and those in its standard bunched sequent calculus. By doing so, we observe not only that this sequent calculus can be seen as an optimised display calculus, but also that the display calculi for the other bunched logics cannot be pared down to a sequent calculus in the same way. Both observations provide strong evidence that our formulation of the proof theory of bunched logics in terms of display calculi is indeed canonical.

Though complete cut-free proof systems for bunched logic are of clear theoreti- cal interest, from the practical perspective it remains to be seen whether our proof theory will find application in automated theorem-proving tools. The need for such tools is quite real, e.g., in the setting of separation logic, which is based on BBI, but since both separation logic and BBI are fundamentally undecidable [[4](#_bookmark30)], com- promises are clearly necessary [6](#_bookmark26) . We suggest that our work might be applied in two

6 In fact, separation logic is significantly more complicated than pure BBI, as it must also account for specific properties of the heap-like models on which it is based.

main directions. First, the display property intuitively corresponds to “pointing” or “focusing” in a proof attempt, where one selects part of a subgoal to work on. Thus our display calculi might well find application in semi-automated proof assistants, where the proof search is partially or wholly guided by humans. Second, it might be possible to obtain useful fully-automated but incomplete proof search tools by imposing constraints on the use of structural rules. A further possibility might be to look at obtaining *deep inference* calculi, which abandon the distinction between log- ical connectives and structural ones [[5](#_bookmark31)], for bunched logics by attempting to extract formula-rewriting rules from their cut-free display calculi. Our approach may also open new avenues for display-based proof theories for other logics with relevance for computer science.

# References

1. Nuel D. Belnap, Jr. Display logic. *Journal of Philosophical Logic*, 11:375–417, 1982.
2. Josh Berdine and Peter O’Hearn. Strong update, disposal and encapsulation in bunched typing. In

*Proceedings of MFPS*, ENTCS. Elsevier, 2006.

1. James Brotherston and Cristiano Calcagno. Classical BI (A logic for reasoning about dualising resource). In *Proceedings of POPL-36*, pages 328–339, 2009.
2. James Brotherston and Max Kanovich. Undecidability of propositional separation logic and its neighbours. To appear in *Proceedings of LICS-25*, 2010.
3. Kai Bru¨nnler. Deep inference and its normal form of derivations. In *Proceedings of CiE*, volume 3988 of *LNCS*, pages 65–74, 2006.
4. Cristiano Calcagno, Philippa Gardner, and Uri Zarfaty. Context logic as modal logic: Completeness and parametric inexpressivity. In *Proceedings of POPL-34*, 2007.
5. Bor-Yuh Evan Chang and Xavier Rival. Relational inductive shape analysis. In *Proceedings of POPL- 35*, 2008.
6. Wei-Ngan Chin, Cristina David, Huu Hai Nguyen, and Shengchao Qin. Enhancing modular OO verification with separation logic. In *Proceedings of POPL-35*, 2008.
7. Matthew Collinson, David Pym, and Edmund Robinson. Bunched polymorphism. *Mathematical Structures in Computer Science*, 18(6):1091–1132, 2008.
8. Dino Distefano and Matthew Parkinson. jStar: Towards practical verification for Java. In *Proceedings of OOPSLA*, pages 213–226. ACM, 2008.
9. D. Galmiche, D. Mery, and D. Pym. The semantics of BI and resource tableaux. *Mathematical Structures in Computer Science*, 15:1033–1088, 2005.
10. Didier Galmiche and Dominique Larchey-Wendling. Expressivity properties of Boolean BI through relational models. In *Proceedings of FSTTCS*, 2006.
11. Rajeev Gor´e. Gaggles, Gentzen and Galois: How to display your favourite substructural logic. *Logic* *Journal of the IGPL*, 6(5):669–694, 1998.
12. Samin Ishtiaq and Peter W. O’Hearn. BI as an assertion language for mutable data structures. In

*Proceedings of POPL-28*, 2001.

1. Marcus Kracht. Power and weakness of the modal display calculus. In Heinrich Wansing, editor, *Proof Theory of Modal Logic*, pages 93–121. Kluwer Academic Publishers, 1996.
2. Dominique Larchey-Wendling and Didier Galmiche. Exploring the relation between intuitionistic BI and Boolean BI: An unexpected embedding. *Mathematical Structures in Computer Science*, 19:1–66, 2009.
3. Dominique Larchey-Wendling and Didier Galmiche. The undecidability of Boolean BI through phase semantics. To appear in *Proceedings of LICS-25*, 2010.
4. P.W. O’Hearn and D. J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215– 244, June 1999.
5. David Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series. Kluwer, 2002.
6. David Pym, Peter O’Hearn, and Hongseok Yang. Possible worlds and resources: The semantics of BI.

*Theoretical Computer Science*, 315(1):257–305, 2004.

1. Stephen Read. *Relevant Logic: A Philosophical Examination*. Basil Blackwell, 1987.
2. Jason Reed. A Hybrid Logical Framework. PhD thesis, Carnegie Mellon University, 2009.
3. Greg Restall. Displaying and deciding substructural logics 1: Logics with contraposition. *Journal of Philosophical Logic*, 27:179–216, 1998.
4. John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings of 17th LICS*, 2002.