THE BULLETIN OF SYMBOLIC LOGIC
Volume 5, Number 2, June 1999

THE LOGIC OF BUNCHED IMPLICATIONS

PETER W. O'HEARN AND DAVID J. PYM

Abstract. We introduce a logic **BI** in which a multiplicative (or linear) and an additive (or intuitionistic) implication live side-by-side. The propositional version of **BI** arises from an analysis of the proof-theoretic relationship between conjunction and implication; it can be viewed as a merging of intuitionistic logic and multiplicative intuitionistic linear logic. The naturality of **BI** can be seen categorically: models of propositional **BI**'s proofs are given by bicartesian doubly closed categories, i.e., categories which freely combine the semantics of propositional intuitionistic logic and propositional multiplicative intuitionistic linear logic. The predicate version of **BI** includes, in addition to standard additive quantifiers, multiplicative (or intensional) quantifiers $\forall_{\mathbf{new}}$ and $\exists_{\mathbf{new}}$ which arise from observing restrictions on structural rules on the level of terms as well as propositions. We discuss computational interpretations, based on sharing, at both the propositional and predicate levels.

§1. Introduction.

1.1. On restricting the structural rules. Perhaps the most lasting impression of linear logic, much more than the formal system itself, will be its revealing of the computational significance of the structural rules of Weakening and Contraction [15]. Logically, their absence leads to the decomposition of conjunction into *additive* (&) and *multiplicative* ($\otimes$) forms, which can be given a sequential natural deduction presentation as follows:

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \,\&\, \psi}\,\&I \qquad \frac{\Gamma \vdash \varphi_1 \,\&\, \varphi_2}{\Gamma \vdash \varphi_i}\,(i = 1, 2)\,\&E$$

$$\frac{\Gamma \vdash \varphi \quad \Delta \vdash \psi}{\Gamma, \Delta \vdash \varphi \otimes \psi}\,\otimes I \qquad \frac{\Gamma, \varphi, \psi \vdash \chi \quad \Delta \vdash \varphi \otimes \psi}{\Gamma, \Delta \vdash \chi}\,\otimes E$$

If we have the rules of Weakening and Contraction

$$\frac{\Gamma \vdash \psi}{\Gamma, \varphi \vdash \psi}\,W \quad \frac{\Gamma, \varphi, \varphi \vdash \psi}{\Gamma, \varphi \vdash \psi}\,C$$

then these rules for $\otimes$ and & define the same connective, but without them the connectives are distinct.

Received August 21, 1998; revised December 9, 1998.

1079-8986/99/0502-0003/$4.00

This splitting of connectives has a long history [12], but the possibilities opened up by restricting structural rules were given a new perspective by vivid "resource interpretations" of linear logic. The leading example is perhaps the *number-of-uses* reading in which a proof of a linear implication $\varphi \multimap \psi$ determines a function that *uses* its argument exactly once. Like $\otimes$, the linear implication is "multiplicative", which is to say that it combines separate contexts in its elimination, i.e., binary, rule.

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \multimap \psi} \multimap I \qquad \frac{\Gamma \vdash \varphi \multimap \psi \quad \Delta \vdash \varphi}{\Gamma, \Delta \vdash \psi} \multimap E$$

However, an important message of linear logic is that, in order to obtain an expressive system, one cannot stay in a pretty-but-weak system: it appears to be crucial to allow access to structurals in some manner. This is done in linear logic in a very specific way, via the "!" modality, which admits a recovery of conventional intuitionistic (or additive) implication $\varphi \to \psi$ as $!\varphi \multimap \psi$. The number-of-uses reading of implication is extended to this modality by reading $!\varphi$ as "as many φs as required".

But notice a curious asymmetry here. Removal of the structural rules enables conjunction to be split straightaway into additive and multiplicative parts, while the treatment of additive implication uses an additional notion, a modality. It seems natural to ask: could a direct decomposition of implication be given, as was done for conjunction?

The answer is yes. In particular, access to the structurals can be recovered in another, rather different way, not involving a modality. The technical cost of this conceptual symmetry is that we must work with a more richly structured notion of sequent, entailing a more delicate analysis of the proof-theoretic relationship between implication and conjunction.

Our purpose in this paper is to explore this alternative route to combining multiplicative and additive implication. We introduce a logic **BI**, the logic of *bunched implications*, by giving a proof theory, a semantics of the proof theory, and a semantics of truth. We also consider "resource interpretations" which begin to expose the computational consequences of some of the choices made.

In presenting this material, we will proceed at a high level: we concentrate on ideas rather than formal results; indeed, there are no proofs at all given in this paper. We have two reasons for adopting this approach. Firstly, that the material covered is too broad to be reasonably given, in full detail, in a short article such as this. Secondly, we hope that this style of presentation will help to expose some of **BI**'s main features and properties in a direct fashion; many of these features can, we feel, be appreciated without becoming embroiled in too many formal details and proofs. Companion papers by each of us [28, 34, 33] explore technical aspects of **BI**'s semantics, proof theory and applications in more detail.

1.2. A proof-theoretic introduction. Recall that implication is inextricably bound up with conjunction, or at least with antecedent-forming operations used to formulate sequents. This connection goes so far that it is sometimes said that an introduction rule

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$$

can be regarded, proof-theoretically, as defining the meaning of $\to$. However far we may wish to take this point, it is clear that the character of the implication in a logic is married to, and in a sense determined by, that of the comma or conjunction.

If, as is the case in **BI**, we have two forms of implication then we are faced with the question of which of them to use in the introduction rule,

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \mathbin{?} \psi}.$$

That is, should the conclusion $\varphi \mathbin{?} \psi$ in this rule be a multiplicative or additive implication?

The connection between introduction rules and implications suggests a way out of this impasse: If an antecedent-forming operation determines the behaviour of an implication, and we have two implications, then we should have two antecedent-forming operations. So, we postulate an additional context-forming operation ";", and stipulate that Contraction and Weakening are possible for ";" but not for ",". The introduction rules then become

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \mathrel{-\!\!*} \psi} \qquad \text{and} \qquad \frac{\Gamma; \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}.$$

In this scheme, the antecedents are no longer sequences, but are trees with propositions as leaves and internal nodes labelled by "," or ";", or in short, *bunches* [13, 6, 36].

(Aside: We have chosen to use a different symbol, $\mathrel{-\!\!*}$, for the multiplicative implication, in order to avoid confusion with $\multimap$. Although many of the formal properties in the multiplicative connectives are the same, the way that they mesh with additive connectives gives rise to different behaviour. In particular, some of the common readings of $\multimap$, such as the number-of-uses reading, do not directly carry over to $\mathrel{-\!\!*}$.)

It is all very well to postulate proof rules in this way, but what meaning or significance, if any, does the resulting logic have? We argue herein that **BI** possesses two very natural semantics. The first, a semantics of the proof theory, arises from *doubly closed categories* (DCCs), in which a single category admits two closed structures or function spaces. It shows clearly the difference between **BI** and linear logic, where two closed categories are usually used. We shall return to this point in §§3.2 and 8.

The second semantics is a Kripke-style semantics of formulæ, which combines Kripke's semantics of intuitionistic logic and Urquhart's semantics of multiplicative, intuitionistic linear logic. This second semantics gives **BI** more of a genuine status as a logic. That is, it gives us a way to read the formulæ as propositions that are true or false, relative to a given world.

The resource interpretations of **BI** are stated in terms of a notion of *sharing*. We explain this in two areas: imperative programming, where the "resource" is a portion of the computer's store; and logic programming, where the "resource" refers to variables occurring in answer substitutions, and their locations on the paths followed during proof-search.

We also consider the vexed question of multiplicative quantifiers. In **BI**, the quantifiers come in two varieties:

$$\begin{array}{lll} \text{Additive quantifiers} & \forall & \exists \\ \text{Multiplicative quantifiers} & \forall_{\mathbf{new}} & \exists_{\mathbf{new}}. \end{array}$$

The additive ones are exactly the quantifiers found in classical, intuitionistic and linear logic. The multiplicative ones are new, being *intensional* quantifiers of the kind long-sought in relevant logic.

Throughout the course of the paper we concentrate, for the most part, on propositional **BI**; quantifiers are considered briefly in §§6 and 7. The general point, however, is that structural restrictions make just as much sense on level of terms in predicate logic as they do on the level of propositions. The quantifiers reflect this, giving rise to a form of "resource sensitivity" over individuals or term denotations in a predicate logic. In fact, predicate **BI** is a rich, subtle and, we claim, elegant logic which makes many more distinctions between connectives than propositional **BI**. However, this theory is beyond the scope of this introductory article; the details of predicate **BI** can be found in [33].

1.3. The significance of BI. We must emphasize that the tools used to define and analyze **BI** have been available for some time. For one, bunches have been used in sequent calculi for relevant logics [13]; the mere fact of existence of a logic like **BI** would come as no surprise to relevantists. For another, many of the categorical fundamentals of **BI** had, with the benefit of hindsight, already been laid down even earlier, in a classic paper of Day [10]. One of our main aims here is to stress the semantic link-up, as providing a theoretical justification for the informal derivation of bunched structure from proof-theoretic considerations, as sketched in this section.

The logical significance of **BI** can be seen to reside in the clarity of its semantics. DCCs directly combine cartesian closed categories and symmetric monoidal closed categories, i.e., of models of intuitionistic logic and multiplicative intuitionistic linear logic [24, 23, 42], structures which exist independently of logic, in mathematics. Indeed, DCCs also arise naturally; see §§3 and 3.1. Similarly, **BI**'s Kripke semantics is a direct combination of Kripke's semantics for intuitionistic logic and Urquhart's semantics for multiplicative intuitionistic linear logic. This latter semantics provides a link to **BI**'s computational significance. An elementary examination of the nature of resources, at both the propositional and predicate levels, leads one to consider a semantics of this type. This point of view is discussed in [34, 33, 32].

We say more on the connection to relevant logic at the end of the paper in §8, where we also give a detailed comparison to linear logic, and discuss **BI**'s place in the general context of logics that limit use of the structural rules.

§**2. Propositional BI.** In this section, we give a presentation of **BI** in sequential natural deduction form, i.e., a sequent presentation based on introduction and elimination rules.

Connectives.

Additive connectives	$1 \wedge \to \bot \vee$
Multiplicative connectives	$I \ast \mathrel{-\!\!\ast}$

The additive connectives correspond to those of intuitionistic logic, **IL**, whereas the multiplicative connectives correspond to those of multiplicative, intuitionistic linear logic, or **MILL** (we refer to [43] for background information on the two systems).

Bunches.

$$
\begin{array}{rclr}
\Gamma & ::= & \varphi & \text{propositional assumption} \\
& \mid & \{\}_m & \text{multiplicative unit} \\
& \mid & \Gamma, \Gamma & \text{multiplicative combination} \\
& \mid & \{\}_a & \text{additive unit} \\
& \mid & \Gamma; \Gamma & \text{additive combination.}
\end{array}
$$

We write $\Gamma(\Delta)$ to indicate a bunch in which Δ appears as a subtree. We write $\Gamma(\Delta')$ for the similar tree where Δ' replaces Δ. (So $\Gamma(-)$ is a kind of positional notation indicating an incomplete tree that can be completed by placing various trees in its hole.) We don't present a more compact representation of bunches using, say, nested multisets or sequences instead of binary operators; the real point of bunches is to let us get the implications right. We use an equivalence on trees instead of worrying about representation.

Coherent equivalence: $\Gamma \equiv \Gamma'$**.** $\equiv$ is the least equivalence relation on bunches satisfying

1. Commutative monoid equations for $\{\}_a$ and ;
2. Commutative monoid equations for $\{\}_m$ and ,
3. Congruence: if $\Delta \equiv \Delta'$ then $\Gamma(\Delta) \equiv \Gamma(\Delta')$.

Note that ";" and "," do not distribute over one another.

Judgements. Logical consequences are of the form

$$\Gamma \vdash \varphi$$

where Γ is a bunch and φ a formula. The rules for deriving judgements now follow.

Identity and structure.

$$\frac{}{\varphi \vdash \varphi}\ Id \qquad \frac{\Gamma \vdash \varphi}{\Delta \vdash \varphi} \equiv \ (\text{where } \Delta \equiv \Gamma)$$

$$\frac{\Gamma(\Delta) \vdash \varphi}{\Gamma(\Delta; \Delta') \vdash \varphi}\ W \qquad \frac{\Gamma(\Delta; \Delta) \vdash \varphi}{\Gamma(\Delta) \vdash \varphi}\ C$$

Multiplicatives.

$$\frac{}{\{\}_m \vdash I}\ II \qquad \frac{\Gamma(\{\}_m) \vdash \chi \quad \Delta \vdash I}{\Gamma(\Delta) \vdash \chi}\ IE$$

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \mathrel{-\!\!*} \psi} \mathrel{-\!\!*} I \qquad \frac{\Gamma \vdash \varphi \mathrel{-\!\!*} \psi \quad \Delta \vdash \varphi}{\Gamma, \Delta \vdash \psi} \mathrel{-\!\!*} E$$

$$\frac{\Gamma \vdash \varphi \quad \Delta \vdash \psi}{\Gamma, \Delta \vdash \varphi * \psi}\ {*I} \qquad \frac{\Gamma(\varphi, \psi) \vdash \chi \quad \Delta \vdash \varphi * \psi}{\Gamma(\Delta) \vdash \chi}\ {*E}$$

Additives.

$$\frac{}{\{\}_a \vdash 1}\ 1I \qquad \frac{\Gamma(\{\}_a) \vdash \chi \quad \Delta \vdash 1}{\Gamma(\Delta) \vdash \chi}\ 1E$$

$$\frac{\Gamma; \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}\ {\to}I \qquad \frac{\Gamma \vdash \varphi \to \psi \quad \Delta \vdash \varphi}{\Gamma; \Delta \vdash \psi}\ {\to}E$$

$$\frac{\Gamma \vdash \varphi \quad \Delta \vdash \psi}{\Gamma; \Delta \vdash \varphi \wedge \psi}\ {\wedge}I \qquad \frac{\Gamma(\varphi; \psi) \vdash \chi \quad \Delta \vdash \varphi \wedge \psi}{\Gamma(\Delta) \vdash \chi}\ {\wedge}E$$

$$\frac{\Gamma \vdash \varphi_i}{\Gamma \vdash \varphi_1 \vee \varphi_2}\ (i = 1, 2)\ {\vee}I \qquad \frac{\Gamma \vdash \varphi \vee \psi \quad \Delta(\varphi) \vdash \chi \quad \Delta(\psi) \vdash \chi}{\Delta(\Gamma) \vdash \chi}\ {\vee}E$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \varphi}\ {\bot}E$$

We do not include a multiplicative disjunction (such as fission, or par) in **BI**, not because it is straightforward, but because it is not. Such a disjunction would come along naturally with a calculus which has bunched multiple conclusions.

Notice that the introduction and elimination rules for additive and multiplicative implications, conjunctions and their units are identical in form. The difference between them is the antecedent-combining operations they use. Because ";" admits Weakening and Contraction, rules where additive maintenance is explicit are admissible:

$$\frac{\Gamma \vdash \varphi \to \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi} \qquad \frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}.$$

The question of whether these rules can replace the more explicit ones above is delicate [28, 34].

LEMMA 1. *The Cut rule is admissible*:

$$\frac{\Gamma(\varphi) \vdash \psi \quad \Delta \vdash \varphi}{\Gamma(\Delta) \vdash \psi}\ \mathrm{Cut}$$

Notice that this formulation of cut covers both the ";" and "," cases in the construction of bunches. A presentation of **BI** as a sequent calculus, in which elimination rules are replaced by left-rules, can be found in [34, 33], in which are detailed the connection to the sequential natural deduction formulation, including its presentation as the $\alpha\lambda$-calculus, cut-elimination and normalization. For now, the following essential result suffices:

PROPOSITION 2. ***BI**'s natural deduction system is, with respect to the evident reductions and commuting conversions, strongly normalizing. It has also the subject reduction property.*

BI contains **MILL** and **IL** as sublanguages; the precise connection to them can be stated as follows:

PROPOSITION 3.

1. **BI** *is conservative over* **IL**. *That is,* $\varphi_1; \ldots; \varphi_n \vdash \varphi$ *is provable in* **BI** *if and only if* $\varphi_1, \ldots, \varphi_n \vdash \varphi$ *is provable in* **IL**, *where each* φ_i *and* φ *is a formula built up using additive connectives only.*
2. **BI** *is conservative over* **MILL**. *That is,* $\varphi_1, \ldots, \varphi_n \vdash \varphi$ *is provable in* **BI** *if and only if* $\varphi_1^*, \ldots, \varphi_n^* \vdash \varphi^*$ *is provable in* **MILL**, *where each* φ_i *and* φ *is a formula built up using multiplicative connectives only and* $(-)^*$ *replaces* $*$ *by* $\otimes$ *and* $-\!\!*$ *by* $\multimap$.

An important point is that conservativity does not extend to **MAILL**, which is **MILL** extended with additive conjunction and disjunction. The reason is that **BI** admits distribution of additive conjunction over disjunction, by which we mean $\varphi \wedge (\psi \vee \chi) \vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$ and its converse. To prove distribution in **BI** one can simply use the usual intuitionistic proof, which refers to ";" but not ",". **MAILL**, on the other hand, does not admit distribution. We will return to this point in §8.

That the conservativity properties in the proposition hold can be seen straightforwardly from a semantics of **BI**'s proof theory.

§3. Doubly closed categories. In order to explain the structure of a proof theory, it is usually useful to give a categorical description of it, in which the connectives enjoy universal properties and satisfy naturality conditions. To do this for **BI**, note that the semantics of proofs for **IL** is given using cartesian closed categories and that for **MILL** using symmetric monoidal closed categories [24, 23, 42]. In each case, the introduction rule for implication corresponds to an adjunction where the internal hom is a right adjoint: to a cartesian product, for **IL**, and a tensor product, for **MILL**.

To model **BI** we simply ask for a category that has all the structure necessary to model *both* **MILL** and **IL**.

> A *doubly closed category*, or DCC for short, is a category equipped with two monoidal closed structures. A DCC is called *cartesian* if one of the closed structures is cartesian and the other symmetric monoidal and *bicartesian* if, in addition, it has finite coproducts.

In brief, models of proofs for **BI** are given using bicartesian DCCs.

To see how DCC structure gives rise to bunches, consider the two adjunctions

$$[H * E, F] \cong [H, E \mathrel{-\!\!*} F] \qquad [H \wedge E, F] \cong [H, E \to F]$$

where $*$ is a symmetric monoidal product and $\wedge$ a cartesian (or categorical) product.

To mimic these adjunctions in the proof theory one would like to have a context extension corresponding to $*$ and another to $\wedge$. This leads directly to the rules

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \mathrel{-\!\!*} \psi} \qquad \frac{\Gamma; \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$$

and to the tree-like structure of antecedents.

Suppose, for a given DCC, we have an assignment of an object $[\![\mathrm{p}]\!]$ to each propositional letter. We call this an *interpretation*. Any interpretation extends in the obvious way to each formula $[\![\varphi]\!]$, and we can assign an object $[\![\Gamma]\!]$ to each bunch by mapping "," to $*$ and ";" to $\wedge$ in the DCC. (In our notation we are using letters E, F H to refer directly to the objects of a DCC, with $[\![\varphi]\!]$ indicating an object determined by a formula.)

A form of soundness and completeness for these interpretations can be stated as follows [34, 33]:

PROPOSITION 4. *$\Gamma \vdash \varphi$ is provable in* **BI** *if and only if, for every DCC, $\mathcal{D}$, and interpretation $[\![-]\!]$ in $\mathcal{D}$, the hom set $\mathcal{D}[[\![\Gamma]\!], [\![\varphi]\!]]$ is non-empty.*

A stronger notion of soundness and completeness is also possible [34, 33]. Given interpretations for propositional letters, any **BI** proof of a judgement $\Gamma \vdash \varphi$ determines a morphism $[\![\Gamma]\!] \to [\![\varphi]\!]$ in a DCC. The stronger form connects the equality induced by this DCC semantics to an equational theory for proofs.

(Note: The assignment of morphisms to derivations should be evident, except that the interpretation of the elimination rule for $\vee$ we need to use distributivity of both $*$ and $\wedge$ over $\vee$. We get that because $E * (-)$ and $E \wedge (-)$ are both left adjoints, and so they preserve coproducts.)

The conservativity of **BI** over **IL** can now be seen immediately in terms of DCCs. For suppose $\mathcal{C}$ is a categorical model of **IL**, i.e., a bicartesian closed category. Then we can regard it as a bicartesian DCC, in which the two closed structures are the same. When we restrict to the additive fragment of **BI**, this shows that the denotations of **BI** proofs are exactly the same as those in the model of **IL**.

The DCC semantics suggests some, at first sight, unusual properties. In particular, a morphism from E to F can variously be viewed as a map $I \to E \mathrel{-\!\!*} F$ or $1 \to E \to F$ using the adjunctions and, indeed, we have the following isomorphisms of hom sets:

$$[1, E \to F] \cong [E, F] \cong [I, E \mathrel{-\!\!*} F].$$

In **BI**, these isomorphisms are realized by the following deductions:

$$\dfrac{\dfrac{\varphi \vdash \psi}{\{\}_a ; \varphi \vdash \psi}}{\{\}_a \vdash \varphi \to \psi} \qquad \dfrac{\dfrac{\varphi \vdash \psi}{\{\}_m , \varphi \vdash \psi}}{\{\}_m \vdash \varphi \mathrel{-\!\!*} \psi}$$

$$\dfrac{\dfrac{\{\}_a \vdash \varphi \to \psi \quad \overline{\varphi \vdash \varphi}}{\{\}_a ; \varphi \vdash \psi}}{\varphi \vdash \psi} \qquad \dfrac{\dfrac{\{\}_m \vdash \varphi \mathrel{-\!\!*} \psi \quad \overline{\varphi \vdash \varphi}}{\{\}_m , \varphi \vdash \psi}}{\varphi \vdash \psi}.$$

Although these observations may make the difference between $\mathrel{-\!\!*}$ and $\to$ appear rather thin, the two implications are not interconvertible:

$$\varphi \mathrel{-\!\!*} \psi \nvdash \varphi \to \psi \qquad \text{and} \qquad \varphi \to \psi \nvdash \varphi \mathrel{-\!\!*} \psi.$$

An example of a model will help to confirm these remarks.

EXAMPLE 5. **Set** $\times$ **Set** is bicartesian closed, with coproducts and cartesian closed structure defined pointwise from their corresponding versions in **Set**. A symmetric monoidal closed structure is given by

$$\begin{aligned} I &= (1, 0) \\ (E_0, E_1) * (F_0, F_1) &= ((E_0 \times F_0) + (E_1 \times F_1),\ (E_0 \times F_1) + (E_1 \times F_0)) \\ (E_0, E_1) \mathrel{-\!\!*} (F_0, F_1) &= ((E_0 \to F_0) \times (E_1 \to F_1),\ (E_0 \to F_1) \times (E_1 \to F_0)). \end{aligned}$$

More generally, if $\mathcal{M}$ is a commutative monoid, considered as a discrete monoidal category, then $\mathbf{Set}^{\mathcal{M}}$ is a bicartesian DCC; in this example, $\mathcal{M}$ is the two-element commutative monoid $\{0, 1\}$ with addition modulo two.

Example 5 does not appear to convey any particularly useful computational ideas but we can use it to make a few remarks.

1. It is a non-degenerate model, in that I is not a terminal object and $*$ is not cartesian product. So the definition of DCCs does not somehow induce a collapse of the specified structure.
2. There are no maps in the model from 1 to I.
3. $((0, 1) \to (1, 0)) = (1, 0)$ and $((0, 1) \mathrel{-\!\!*} (1, 0)) = (0, 1)$. This, combined with the fact that there are no maps between $(0, 1)$ and $(1, 0)$ in either direction, implies that there are no maps from $((0, 1) \to (1, 0))$ to $((0, 1) \mathrel{-\!\!*} (1, 0))$ or back, confirming the remark above that $\to$ and $\mathrel{-\!\!*}$ are not convertable to one another in the linear version of the bunched language.
4. There is no functor $! : \mathbf{Set} \times \mathbf{Set} \to \mathbf{Set} \times \mathbf{Set}$ admitting an isomorphism $!E \mathrel{-\!\!*} F \cong E \to F$, thus indicating that a DCC is not simply a model of linear

logic in disguise. To see the remark, consider that $(1, 0) \to (2, 2) = (2, 1)$ but that, for any C is, $C \mathrel{-\!\!*} (2, 2) = (X, Y)$ for sets X and Y of the same cardinality. Therefore, for any "!" we try to pick, $!E \mathrel{-\!\!*} (2, 2)$ cannot be $(2, 1)$.

3.1. Day's construction. We can generate a rich class of models of **BI** using a general construction due to Brian Day [10]. He shows that any monoidal (not necessarily closed) category $(\mathcal{C}, *, I)$ induces a monoidal closed structure on the functor category $\mathbf{Set}^{\mathcal{C}^{op}}$, and that when $(\mathcal{C}, *, I)$ is symmetric monoidal so is $\mathbf{Set}^{\mathcal{C}^{op}}$.

The construction is as follows. The unit I of the monoidal structure is $\mathcal{C}[-, I]$. Given functors E and F, the formula for the tensor product is written using co-ends:

$$(E * F)Z = \int^{X,Y} EX \times FY \times \mathcal{C}[Z, X * Y].$$

The formula for $\mathrel{-\!\!*}$ uses an end:

$$(E \mathrel{-\!\!*} F)Z = \int_X \mathbf{Set}^{\mathcal{C}^{op}}[EX, F(Z * X)] \cong \mathbf{Set}^{\mathcal{C}^{op}}[E, F(Z * -)].$$

The formulæ for $(E * F)Z$ and $(E \mathrel{-\!\!*} F)Z$ are both contravariant in Z, giving the morphism parts of the functors. It is often possible to give an explicit description of the tensor product without using co-ends at all. In fact, we already did this with the $\mathbf{Set} \times \mathbf{Set}$ example above.

Two observations are useful for working with the tensor product. The first is that we have a form of pairing operation: given $a \in EX$ and $b \in BY$ we can form an element $[a, b] \in (E * F)(X * Y)$. To see how this element is defined, consider that the co-end $E * F(Z)$ can be described as a quotient of quintuples $(X, Y, f : Z \to_{\mathcal{C}} X * Y, a \in EX, b \in FY)$. The pair $[a, b]$ is then the equivalence class of $(X, Y, \mathrm{id}_{X*Y}, a, b)$.

The second is a representation result which characterizes maps out of a tensor products: natural transformations $E * F \to H$ are in bijection with families of functions

$$EX \times FY \to H(X * Y)$$

natural in X and Y. To see why this is true, consider the definition of $\mathrel{-\!\!*}$, and the isomorphism $[E * F, H] \cong [E, F \mathrel{-\!\!*} H]$: the multi-map characterization is, essentially, forced by $\mathrel{-\!\!*}$.

To obtain models of **BI**, we can combine this structure for multiplicatives with the standard fact that $\mathbf{Set}^{\mathcal{C}^{op}}$ is bicartesian [24]. Finite products are determined pointwise, from those in **Set**, and the additive exponent can be represented as

$$(A \to B)Z = \mathbf{Set}^{\mathcal{C}^{op}}[\mathcal{C}[-, Z] \wedge A, B].$$

PROPOSITION 6 ([10]). *If $\mathcal{C}$ is a symmetric monoidal category, then $\mathbf{Set}^{\mathcal{C}^{op}}$ is a bicartesian DCC.*

It is worth remarking that Day describes his results in much greater generality, in the context of enriched categories, so this gives us many more models than those mentioned in the proposition.

In a separate paper [11], Day also shows that the Yoneda embedding preserves closed structure; this is an analogue of the standard fact that Yoneda preserves CCC structure. From this we may conclude that **BI** is conservative over **MILL**, again not only on the level of provability but also on the semantics of proofs.

3.2. Non-interpretations. We have seen how DCCs expose some of the workings of **BI**'s proof theory: They can equally be used to indicate how the theory does *not* work by considering some non-examples of models.

First, CPOs (pointed, ω-complete partial orders) admit constructions that are reminiscent of **BI**. If E and F are CPOs then the strict function space $E \multimap F$, continuous function space $E \rightarrow F$, smash product $E \otimes F$ and cartesian product $E \times F$ are all CPOs. However, this does not give us a DCC, because $(\rightarrow, \times, 1)$ is the cartesian closed structure in the category of continuous functions, while $(\multimap, \otimes, I)$ gives monoidal closed structure in the category of strict continuous functions. A DCC requires that a single category admit both closed structures.

The situation with CPOs, where two closed categories are involved, is typical of linear logic. For example, in the coherence space model [15] we have a monoidal exponent $\multimap$ and a cartesian exponent $\rightarrow$, but the former is closed for the category of linear maps and the latter for the category of stable maps.

That, for example, the coherence space model does not work for **BI** may seem just a technical curiosity, but the reasons for why it does not run rather deep and relate to "resource interpretations" of the connectives [28]. For example, the judgement

$$\{\}_m \vdash \varphi \mathrel{-\!\!*} ((\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \psi)$$

has a proof that uses the φ premiss twice:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{x:\varphi; f\colon \varphi\rightarrow\varphi\rightarrow\psi \vdash f\,x : \varphi\rightarrow\psi} \qquad \cfrac{\vdots}{x:\varphi; f\colon \varphi\rightarrow\varphi\rightarrow\psi \vdash x:\varphi}}{x : \varphi; f\colon \varphi \rightarrow \varphi \rightarrow \psi \vdash (f\ x)\,x : \psi}\ C, \rightarrow E}{x : \varphi \vdash \alpha f\,.\,(f\ x)\,x : ((\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \psi)}\ \rightarrow I}{\{\}_m, x : \varphi \vdash \alpha f\,.\,(f\ x)\,x : ((\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \psi)}\ \equiv}{\{\}_m \vdash \lambda x\,.\,\alpha f\,.\,(f\ x)\,x : \varphi \mathrel{-\!\!*} ((\varphi \rightarrow \varphi \rightarrow \psi) \rightarrow \psi)}\ \mathrel{-\!\!*} I$$

Here we have used a λ-calculus-like notation for proof-terms, where α stands for additive abstraction and λ for multiplicative (or λinear). In the key, top-pictured, step we use the admissible rule for $\rightarrow$ elimination (or equivalently we use $\rightarrow E$ followed by Contraction). It is at this step that the way that $\rightarrow$ admits sharing between $(f\ x)$ and its argument x appears.

The point of this example is the two occurrences in the body $(f\ x)\,x$ of the argument x to a $-\!\!*$-typed function. This serves to illustrate that the idea that a multiplicative, or "linear", function uses its argument exactly once does not directly carry over to **BI**. For this reason, we have chosen not to adopt the symbol $\multimap$, with the associated readings it tends to carry, for **BI**'s multiplicative implication.

If the number-of-uses reading—which is often considered to be characteristic for a logic, like linear logic, which restricts Contraction—does not carry over, then what meaning do the **BI** connectives have? And what is the justification for this judgement?

We can answer these questions in two ways. First, we claim that the Kripke resource semantics, given later, provides a natural truth-based semantics for **BI**. We do not have to appeal to λ-calculus to make sense of it (although we certainly can do so). Secondly, although **BI** can be understood in terms of its truth semantics, we can also offer a resource-based interpretation of proofs which does justify these judgements. We call this the (propositional) *sharing interpretation*.

§4. The sharing interpretation, I. To set the stage, the background idea is of functional programming data such as pairs, functions and so on, but with an additional, intensional, notion of resource which can be accessed during a computation. The resources are distributed, in the sense that a value may have access to some, to none or to all of them.

We concentrate on the two implications.

- $\varphi -\!\!*\, \psi$: functions that may not share resources with their arguments.
- $\varphi \to \psi$: functions that may share resources with their arguments.

To illustrate how this informal reading of connectives works, recall the "unusual" example of a proof-term $\lambda x\,.\,\alpha f\,.\,(f\ x)\,x$, in which a multiplicative assumption is used twice. From the point of view of the sharing interpretation, the subterm $(f\ x)$ in the proof-term is allowed to share with its argument x, because $f\ x$ is of additive function type $\varphi \to \psi$. That is why $(f\ x)\,x$ is reasonable: There is no requirement that an argument to a $-\!\!*$-typed function be used just once, only that it does not share with other variables in the proof-term. The kind of thing that would be disallowed by the sharing interpretation is an application $(f\ x)\,x$, where f has type $\varphi -\!\!*\, \varphi -\!\!*\, \psi$.

The reading extends to other connectives in a straightforward way. For example, $\varphi * \psi$ is the type of non-sharing pairs, and $\varphi \wedge \psi$ is the type of pairs that might share.

The sharing interpretation is inspired by John Reynolds's work on syntactic control of interference and Idealized Algol ([37, 38], and also the relevant [29]). These are programming languages that use affine and intuitionistic

λ-calculus respectively, together with imperative features such as the assignment statement $x := e$. The calculi give the "functional data" of the sharing interpretation, and the computer's store gives the "intensional" component; see [28].

A concrete model that corresponds to the informal reading can be obtained as an instance of Day's construction. Let $\mathcal{I}$ be the category of finite sets and injective functions. We think of an object X here as a possible world which identifies a finite set of cells in the computer memory. These worlds are used to capture the intensional part of the sharing interpretation, that part which refers to resource instead of merely to values.

The (object parts of the) functors for implication display the basic character of the semantics: Given $E, F : \mathcal{I} \to \mathbf{Set}$,

$$(E \mathrel{-\!\!*} F)(X) = \mathbf{Set}^{\mathcal{I}}[E(-), F(X + -)]$$
$$(E \to F)(X) = \mathbf{Set}^{\mathcal{I}}[E(X + -), F(X + -)]$$

where $+$ is the evident functor on $\mathcal{I}$ given by disjoint union in **Set**. Notice how these clauses correspond directly to the sharing interpretation. For $\mathrel{-\!\!*}$, the absence of X in $E(-)$ indicates how a function (which lives in world X) and argument must access disjoint resources. For $\to$, the presence of X in $E(X + -)$ indicates how a procedure may share store with its argument. The other connectives are defined as in Day's construction.

We can relate these definitions further to the informal reading by defining a functor **cell**, which plays the role of the type of storage cells: It is the inclusion functor from $\mathcal{I}$ to **Set**. The value $\mathbf{cell}\, X = X$ of **cell** at world X is thought of as the set of cells associated with that world.

Now, consider any element $p \in \mathbf{cell} \mathrel{-\!\!*} (\mathbf{cell} \mathrel{-\!\!*} E)X$. The definition of $\mathrel{-\!\!*}$ says that, for an arbitrary Y and $a \in \mathbf{cell}\, Y$, $p[Y]a$ is an element of $(\mathbf{cell} \mathrel{-\!\!*} E)(X + Y)$. Unpacking further, this says that, for an arbitrary Z and $b \in \mathbf{cell}\, Z$, $(p[Y]a)[Z]b \in E(X + Y + Z)$. The crucial point is that a is in the X-component and b in the Z-component: so, from the point of view of the summed-up world $X + Y + Z$, they must be different cells.

(An aside: Strictly speaking, the sharing interpretation presented in this section works as well for the affine variant of **BI**, where "," admits Weakening. In terms of the model, $\mathbf{Set}^{\mathcal{I}}$ is an affine DCC, in that the unit $I = \mathcal{I}[\{\}, -]$ is a terminal object. A linear variant of this model can be obtained by replacing $\mathcal{I}$ by the category of finite sets and bijections. The informal reading of connectives is then adjusted by saying that an additive function accesses the same resources as its argument.)

Our intention in this section has been to explain how there is a resource interpretation of **BI** proofs that is consistent with the fact that $\mathrel{-\!\!*}$-typed functions in **BI** (or its calculus of proof-terms) can use their arguments multiple times. We have seen that such an interpretation is possible, and given a specific model corresponding to it. We have also hinted at a connection to imperative programming, but a full study of that connection would take us too far afield here; we refer to the companion paper [28] for more information.

§5. Kripke resource semantics. Thus far we have considered **BI** from proof- or type-theoretic perspectives, with a semantic focus on the interpretation of proofs. However, **BI** can also be understood from a model-theoretic point of view, in which we consider a notion of truth for formulæ, which are thus seen to denote propositions in the established sense. We are not looking for an algebraic semantics here, where one takes (say) a Heyting algebra with enough structure to model the multiplicatives; this would just be a collapsed version of the DCC semantics, and would not be very informative.

A more penetrating semantics of **MILL** was given by Urquhart [44]. The semantics uses possible worlds, arranged as a commutative monoid, which Urquhart justifies in terms of a philosophical analysis of a notion of "piece of information". Recall also that Kripke gave a possible world semantics of **IL**, which can be understood as using functor categories $\mathbf{Set}^{\mathcal{P}}$, where $\mathcal{P}$ is a preorder [22]. We can obtain a semantics of **BI**, which combines these two semantics directly, by working in categories $\mathbf{Set}^{\mathcal{C}^{\mathrm{op}}}$, where $\mathcal{C}$ is a symmetric monoidal category. For simplicity, we describe this for the special case that $\mathcal{C}^{\mathrm{op}} = \mathcal{M}$ is a partially-ordered commutative monoid. (This means that the multiplication of the monoid must be monotone, but note that it need not be given by meet or join, and the unit need not be a least or greatest element.)

Given such a monoid $\mathcal{M} = (M, \cdot, e, \sqsubseteq)$, the semantics of **BI** is defined via a forcing relation $m \models \varphi$, for a world $m \in M$ and formula φ. All propositions will satisfy the familiar monotonicity property from intuitionistic logic:

Kripke Monotonicity: $n \models \varphi$ *and* $m \sqsubseteq n$ *implies* $m \models \varphi$.

In more detail, an interpretation must specify the forcing relation $m \models \mathrm{p}$ for propositional variables p in a way compatible with Kripke Monotonicity. The clauses for the additive connectives are then

$$
\begin{array}{lll}
m \models 1 & & \text{always} \\
m \models \varphi \wedge \psi & \text{if and only if} & m \models \varphi \text{ and } m \models \psi \\
m \models 0 & & \text{never} \\
m \models \varphi \vee \psi & \text{if and only if} & m \models \varphi \text{ or } m \models \psi \\
m \models \varphi \rightarrow \psi & \text{if and only if} & \forall n \sqsubseteq m \ (n \models \varphi \text{ implies } n \models \psi).
\end{array}
$$

The multiplicatives follow Urquhart's semantics of **MILL**.

$$
\begin{array}{lll}
m \models I & \text{if and only if} & m \sqsubseteq e \\
m \models \varphi * \psi & \text{if and only if} & \exists n, n' \in M \\
 & & (m \sqsubseteq n \cdot n' \text{ and } n \models \varphi \text{ and } n' \models \psi) \\
m \models \varphi \mathrel{-\!\!*} \psi & \text{if and only if} & \forall n \in M \ (n \models \varphi \text{ implies } n \cdot m \models \psi).
\end{array}
$$

If the order on the monoid $\mathcal{M}$ is discrete then the clauses for $*$ and I reduce to Urquhart's semantics for **MILL**. The clause for $-\!\!*$ is precisely the one used by him. Given this definition, formulated in the category $\mathbf{Set}^{\mathcal{M}}$, we can proceed to establish soundness and completeness theorems, with a model-existence result being established via a term model constructed using normal natural deductions [34, 33].

We can read the clauses informally in terms of resources. For example, the formula for $*$ says, roughly, that a world forces a multiplicative conjunction if and only if there is a *partition of that world* into components which force the constituents of the formula. This narrative about partitioning suggests a sense in which multiplicative (or intensional, in the relevant terminology) connectives depend on the *internal structure* of worlds. On top of this, one can understand the formula for $-\!\!*$ as follows: if m is the cost of the function and if n is the cost of (any) argument, then the cost of obtaining the result of applying the function to the argument is $m \cdot n$. The clauses for additives should be understood in terms of conservation, rather than splitting, of information. For example, to establish an additive conjunction one can make use of the same information when showing each conjunct.

The reader might have recognized a similarity to Day's construction in the treatment of the multiplicatives. This can be made precise as follows: Given an interpretation of propositional letters, we can define an object $[\![\mathrm{p}]\!]$ in $\mathbf{Set}^{\mathcal{M}}$, where

$$[\![\mathrm{p}]\!]m = \begin{cases} \{*\} & \text{if } m \models \mathrm{p} \\ \{\} & \text{if } m \not\models \mathrm{p} \end{cases}$$

with the morphism part being evident. $[\![-]\!]$ extends to all formulæ inductively, using the DCC structure of $\mathbf{Set}^{\mathcal{M}}$. The connection between the two semantics can then be stated as follows:

PROPOSITION 7. *For all formulæ φ and worlds m, $[\![\varphi]\!]m$ is non-empty if and only if $m \models \varphi$.*

An important point is that $[\![\varphi * \psi]\!]m$ can have more than one element if $\mathcal{M}$ is not connected; not every $[\![\varphi]\!]$ is a subobject of 1. Thus, Urquhart's form of semantics is not exactly the same as Day's, but it is closely related to it and can be recovered from it [34, 33].

We regard the DCC and Kripke resource semantics as complementary. DCCs provide a proof-theoretic view, emphasizing adjoint relationships between implications and conjunctions. The Kripke semantics exists independently, and emphasizes a notion of truth, parameterized by worlds. The world structure can be understood as axiomatizing some minimal assumptions about resources, in a way similar to the considerations of Urquhart, but with some key points of divergence [34]. Day's construction provides the link between the two views.

One last comment. In his work Urquhart was aiming to model relevant logics which admit Contraction, and for this he included an equation $m \cdot m = m$. This lead to a mismatch with the relevant logic **R**, where $(\varphi \mathrel{-\!\!*} \psi \vee \chi) \wedge (\psi \mathrel{-\!\!*} \chi) \mathrel{-\!\!*} (\varphi \mathrel{-\!\!*} \chi)$ holds in Urquhart's semantics but is not provable in **R**. We do not have such a mismatch here for two reasons. First, we do not accept the equation, because **BI** does not admit Contraction for ",". Second, even if we were to add Contraction to **BI** we would not insist on $m \cdot m$ being equal to m, but only greater or equal. In this respect, our semantics is related to the ternary relation semantics of Routley and Meyer [14, 36]; but taking a commutative monoid, along the lines of Urquhart, as primitive seems a natural choice in the light of Day's construction.

§**6. Quantifiers.** The predicate version of **BI** has the familiar intuitionistic quantifiers $\forall$ and $\exists$. It also has intensional quantifiers, obtained by observing structural restrictions on the level of terms as well as propositions. This is formulated using a notion of *bunch of variables*, which is separate from bunches of propositions. We describe this here using a single-sorted version of predicate **BI**.

Bunches of variables. x is used to range over variables, and X over bunches of variables.

$$X ::= x \mid X; X \mid X, X \mid \{\}_m \mid \{\}_a.$$

Bunches of variables are subject to the linearity restriction: any variable appears at most once in a bunch.

Judgements. We consider terms- and propositions-in-context, with a syntax of the form

$$X \vdash t : \mathbf{Term} \qquad \text{and} \qquad X \vdash \varphi : \mathbf{Prop}$$

which assert that a term or proposition is well-formed in context X [25]. Constants and predicate letters are given by schematic judgements and, as in the bunched logic itself, Contraction and Weakening are allowed for ";" but not for ",". We omit a formal definition and move on to consider the quantifiers.

Logical judgements have the form

$$(X)\, \Gamma \vdash \varphi$$

asserting that φ is a consequence of Γ, where the terms and atomic predicates in the sequent are well-formed according to X.

Multiplicatives.

$$\frac{(X, x)\ \Gamma \vdash \varphi}{(X)\ \Gamma \vdash \forall_{\mathbf{new}} x.\varphi}\ \forall_{\mathbf{new}} I$$

$$\frac{(X)\ \Gamma \vdash \forall_{\mathbf{new}} x.\varphi \quad Y \vdash t : \mathbf{Term}}{(X, Y)\ \Gamma \vdash \varphi[t/x]}\ \forall_{\mathbf{new}} E$$

$$\frac{(X, Y)\ \Gamma \vdash \varphi[t/x] \quad Y \vdash t : \mathbf{Term}}{(X)\ \Gamma \vdash \exists_{\mathbf{new}} x.\varphi}\ \exists_{\mathbf{new}} I$$

$$\frac{(X)\ \Gamma \vdash \exists_{\mathbf{new}} x.\varphi \quad (Y, x)\ \Delta, \varphi \vdash \psi}{(X, Y)\ \Gamma, \Delta \vdash \psi}\ \exists_{\mathbf{new}} E$$

Additives.

$$\frac{(X; x)\ \Gamma \vdash \varphi}{(X)\ \Gamma \vdash \forall x.\varphi}\ \forall I$$

$$\frac{(X)\ \Gamma \vdash \forall x.\varphi \quad X \vdash t : \mathbf{Term}}{(X)\ \Gamma \vdash \varphi[t/x]}\ \forall E$$

$$\frac{(X)\ \Gamma \vdash \varphi[t/x] \quad X \vdash t : \mathbf{Term}}{(X)\ \Gamma \vdash \exists x.\varphi}\ \exists I$$

$$\frac{(X)\ \Gamma \vdash \exists x.\varphi \quad (X; x)\ \Delta; \varphi \vdash \psi}{(X)\ \Gamma; \Delta \vdash \psi}\ \exists E$$

The idea of the introduction rule for $\forall_{\mathbf{new}}$ is that we may infer $\forall_{\mathbf{new}} x.\varphi$ in the usual way for universal quantification, except that the variable x must sit in multiplicative combination with all of the other variables. In the elimination rule we must be careful not to substitute an arbitrary term for x, but only one that is respective of the multiplicative relationship between x and other variables in X. In particular, t cannot contain any of these other variables appearing in X; this requirement is implemented by the linearity restriction. The universal introduction rules have the usual restriction that x must not occur in Γ; similarly, the existential elimination rules have the restriction that x must not occur in Δ or ψ.

A consequence of these observations is that it is not exactly reasonable to read $\forall_{\mathbf{new}}$ literally as "for all". Rather, we must take into account that in $\forall_{\mathbf{new}} x.\varphi$ the multiplicative relationship between x and other variables must be observed. In terms of the sharing interpretation, we would read it as "for all x that don't share resources with other variables in φ", or more briefly, for all *new* x. The nomenclature for $\forall_{\mathbf{new}}$ can be considered to come from the relationship it has with new storage variables in Idealized Algol. Similar considerations apply to $\exists_{\mathbf{new}}$.

It is important to avoid the trap of thinking of $\forall_{\mathbf{new}}$ as an infinite multiplicative conjunction; this idea is sometimes mentioned, by analogy with the view of additive quantification as an infinite additive conjunction. Rather, $\forall_{\mathbf{new}}$ is closely allied to the multiplicative implication $-\!\!*$, although a version allied to multiplicative conjunction is possible under restricted circumstances. It would be interesting to formulate a dependent function type, along the lines of [20], which generalizes both of them.

For the additive universal quantifier one might have expected here that the elimination rule would be formulated as

$$\frac{(X)\ \Gamma \vdash \forall x.\varphi \quad X \vdash t : \mathbf{Term}}{(X)\ \Gamma \vdash \varphi[t/x]}\ \forall E$$

Such a rule is admissible if we have Contraction for ";" on the level of terms. Similar considerations apply to the additive existential quantifier.

We now extend the Kripke Resource Semantics to quantifiers. To do this, we must first define a notion of *environment*, which specifies a binding of variables to individuals. For this, we suppose that we are given a functor D in $\mathbf{Set}^{\mathcal{M}}$; this functor is the "domain of individuals". As in the possible worlds semantics of intuitionistic logic, the use of a functor instead of a constant set allows different collections of individuals to exist at different worlds.

EXAMPLE 8. Consider the monoid $\mathcal{M} = (N, 0, +, \leq)$ of natural numbers, under addition. Then D might be taken to be the functor sending n to the set $\{0, \ldots, n-1\}$, with the morphism part being inclusion. Here, we may regard the set $D(n)$ as specifying the collection of memory locations active at world n, borrowing from the sharing interpretation in terms of imperative programming.

Continuing with the definitions, given such a functor D we may define the *environment functor* $[\![X]\!]$, for X a bunch of variables, as follows:

$$\begin{array}{ll} [\![x]\!] = D & \\ [\![\{\}_m]\!] = I & [\![X, Y]\!] = [\![X]\!] * [\![Y]\!] \\ [\![\{\}_a]\!] = 1 & [\![X; Y]\!] = [\![X]\!] \wedge [\![Y]\!]. \end{array}$$

Here, the $*$ on the right is Day's tensor product, and $\wedge$ is cartesian product of functors. As a result, each $[\![X]\!]$ is a functor, so that when $n \sqsubseteq m$ and $a \in [\![X]\!]m$ we obtain an element $[\![X]\!](n \sqsubseteq m)a \in [\![X]\!]n$.

The forcing relation for predicate **BI** is of the form

$$(X)\ u \mid m \models \varphi$$

where $m \in M$ is a world as before, φ is a formula, possibly involving quantifiers and predicate letters, and $u \in [\![X]\!]m$ is an *environment* appropriate to a bunch of variables X, at world m. (We emphasise here that an environment is an element of a set, and not a map in $\mathbf{Set}^{\mathcal{M}}$.)

The functorial action of environments enables us to formulate an extension of Kripke Monotonicity:

$$(X)\ u \mid n \models \varphi \text{ and } m \sqsubseteq n \text{ implies } (X)\ ([\![X]\!](m \sqsubseteq n)u) \mid m \models \varphi.$$

We now move on to give the semantics of the quantifiers. The semantics of the additive quantifiers is standard, and the multiplicative quantifiers use the monoid structure, as follows:

$$
\begin{array}{lll}
(X)\; u \mid m \models \forall x \,.\, \varphi & \text{if and only if} & \forall n \sqsubseteq m \,.\, \forall d \in Dn \,. \\
 & & \quad (X; x)\; (\llbracket X \rrbracket (n \sqsubseteq m) u, d) \mid n \models \varphi \\
(X)\; u \mid m \models \exists x \,.\, \varphi & \text{if and only if} & \exists d \in Dm \,.\, (X; x)\; (u, d) \mid m \models \varphi \\
(X)\; u \mid m \models \forall_{\mathbf{new}} x \,.\, \varphi & \text{if and only if} & \forall n \,.\, \forall d \in Dn \,. \\
 & & \quad (X, x)\; [u, d] \mid n \cdot m \models \varphi \\
(X)\; u \mid m \models \exists_{\mathbf{new}} x \,.\, \varphi & \text{if and only if} & \exists n \,.\, \exists d \in Dn \,. \\
 & & \quad (X, x)\; [u, d] \mid n \cdot m \models \varphi.
\end{array}
$$

In the multiplicative cases, we have used the pairing operation $[u, d]$ for Day's tensor product (§3.1). In each case, it takes an environment $u \in \llbracket X \rrbracket m$ and an element $d \in Dn$ and forms the element $[u, d] \in (\llbracket X \rrbracket * D) m \cdot n$. (Notice from the definition of environment functors that this functor $\llbracket X \rrbracket * D$ is in fact equal to $\llbracket X, x \rrbracket$, so the definitions are type correct.)

The difference between $\forall$ and $\forall_{\mathbf{new}}$ is that the former considers elements $d \in Dn$ only in accessible worlds where $n \sqsubseteq m$, where for the latter we look to a completely separate world n, and then use the monoid operation $\cdot$ to combine it with m.

The multiplicative existential is different in another respect from its additive counterpart. The additive $\exists$, from intuitionistic logic, is often described as being "local", in that the definition stays at the same world, where for $\forall$ you travel to accessible worlds to find elements. The multiplicative $\exists_{\mathbf{new}}$ does not stay at the same world when it looks for an element that exists, but nor does it travel along accessible lines; it hops to an arbitrary world, accessible or not, to find an element, and then considers the resulting formula in an environment formed by multiplicative combination.

The connection of multiplicative quantifiers to "newness" is clear if we go back to §4, and the sharing interpretation. Hopping to a world multiplicatively apart from the current one guarantees, in the sharing model, that the element obtained doesn't interfere with any identifiers evaluated at the current world. This non-interference property is one of the characteristic properties of local state in imperative languages.

We conclude this section with one further point, first mentioned in the introduction. The inclusion of bunched structure on the level of individuals raises the question of how this information will be incorporated in the rules for propositional connectives. For each connective, we must choose whether to follow multiplicative or additive maintenance on the level of individuals. The most common cases for multiplicative connectives are in fact cross cases, in which a multiplicative connective allows the same variable to be shared between its components (this approach is taken, for example, in predicate versions of linear logic). In **BI**, we also have the more radical possibility of multiplicative maintenance. A full discussion of predicate **BI** is beyond the scope of the present article: the details of predicate **BI**, including its proof theory, Kripke and categorical semantics, in both presheaves and fibred (indexed) categories, can be found in [33]. However, we illustrate some of the main points in the next section, under the heading of logic programming.

§7. **The sharing interpretation, II.** The background idea is now of logic programming, with proof-search as the primitive concept. Again, we have an intensional notion of resource. The resources here are variables occurring in answer substitutions. These will be distributed in the sense that access to variables will be localized to specific branches in the process of proof-search. On the level of the formulæ themselves, this gives rise to a new form of modularity in logic programs.

The proof-theoretic account of logic programming is simple and elegant. A *program* is expressed as an antecedent (or context), $\mathcal{P}$, of a sequent and a *goal* is expressed as a succedent, typically existentially quantified. A goal $\exists x.G$ is often written in the Prolog style as $G(X)$, with X described as a "logical variable". The whole sequent, $\mathcal{P}$?- $\exists x.G$, is interpreted as a request to calculate a pair (Φ, σ) in which σ is an *answer substitution* for X and Φ is a *proof* of the sequent $\mathcal{P} \vdash G\sigma$.

Operationally, we must describe how to execute a program when it is supplied with a goal. Recall first that inference rules can be read as *reduction operators*, from conclusion to premisses. (Kleene [21] explains this in the case of the classical predicate calculus.) Such operators are the basic units of proof-search, or backward chaining, just as inference rules are the basic units of deduction, or forward chaining.

A semantics based on *goal-directed proof-search* is computationally appealing. Fix a program, $\mathcal{P}$, and proceed, informally, as follows:

• Given a complex goal, G, we first reduce G by applying, as a reduction operator, the introduction rule which corresponds to the outermost connective of G. This reduction process is repeated until all remaining goal formulæ are *atomic*. For example, given the goal $G_1 \wedge (G_2 \to G_3)$, we construct the tree

$$\dfrac{\dfrac{\vdots}{\mathcal{P} \text{ ?- } G_1} \qquad \dfrac{\dfrac{\vdots}{\mathcal{P}; G_2 \text{ ?- } G_3}}{\mathcal{P} \text{ ?- } G_2 \to G_3}\,{\to}I}{\mathcal{P} \text{ ?- } G_1 \wedge (G_2 \to G_3)}\,\wedge I.$$

Note that upper rightmost step adds G_2 to the program;

• Given an atomic goal, A, we invoke the program, using a *resolution* step. Suppose the program includes a proposition of the form $\forall x.G \to B$, in which B is atomic, such that there is a substitution σ for x such that $B\sigma = A$.

Then we can immediately proceed to the *sub-goal* $G\sigma$:

$$\frac{\mathcal{P} \;?\text{-}\; G\sigma}{\mathcal{P} \;?\text{-}\; A} \quad (\forall x.G \to B \text{ in } \mathcal{P},\, B\sigma = A).$$

This operational semantics has several desirable features. Most importantly, it is *not very non-deterministic*, thereby reducing the need for backtracking to an acceptable level. More details of this model of computation can be found in [27, 35, 26, 17].

Proofs constructed according to this goal-directed strategy are called *uniform proofs* [27, 35]. Uniform proofs are not complete for all of **BI** (or indeed for all of intuitionistic logic). For example, consider the following, trivially provable, sequent:

$$G \vee H \vdash G \vee H.$$

To see that there is no uniform proof of this sequent, consider the $\vee I$ rule:

$$\frac{\Gamma \vdash G}{\Gamma \vdash G \vee H} \qquad \frac{\Gamma \vdash H}{\Gamma \vdash G \vee H}.$$

Any attempt to reduce the right-hand side of $G \vee H \vdash G \vee H$ first forces a premature choice between G and H leading, for example, to

$$G \vee H \vdash G$$

which is clearly not provable.

However, uniform proofs are complete for a *clausal hereditary Harrop* fragment of **BI**. The basic idea is to restrict the classes of propositions permitted to occur in each side of a sequent, *program clauses*, P, on the left and *goals*, G, on the right. These two classes (here, for simplicity, we make do with a simplified form and suppress some operational issues) are defined by mutual induction as follows, where A ranges over atomic propositions:

$$\begin{array}{rrl} \textit{Program clauses}\ P & ::= & A \mid \forall x.G \to A \mid \forall_{\mathbf{new}} x.G \multimap\!\!* A \\ \textit{Goals}\ G & ::= & A \mid G \wedge G \mid G * G \mid G \vee G \\ & & \mid P \to G \mid P \multimap\!\!* G \mid \\ & & \mid \exists x.G \mid \exists_{\mathbf{new}} x.G. \end{array}$$

A sequent is said to be hereditary Harrop if it is of the form $\mathcal{P} \vdash G$, where $\mathcal{P}$ is a bunch of program clauses and G is a goal. We get a version of the resolution step for each of $\forall x.G \to A$ and $\forall_{\mathbf{new}} x.G \multimap\!\!* A$. In fact, there are also the cross cases, one in which $\forall_{\mathbf{new}}$ is paired with $\to$ and one in which $\forall$ is paired with $\multimap\!\!*$. The latter case is the one taken in linear logic and so in Lolli [18] and Lygon [17, 35].

The proof that uniform proofs are complete for hereditary Harrop sequents relies most naturally on a presentation of **BI** as a sequent calculus [34], in which the elimination rules are replaced by "left rules" which introduce connective to the left-hand side of a sequent. Provided we restrict our attention to normal proofs, such a presentation is equivalent to the natural deduction one. The details of these arguments are beyond the scope of our present purposes. However, it will be helpful to consider the resolution rule itself in a bit more detail. Recall that our treatment of quantifiers required the introduction of a collection of variables to sequents, $(X)\,\Gamma \vdash G$ denoting that X is the set of first-order variables occurring in Γ and G. Resolution makes explicit use of these variables.

In predicate **BI**, the elimination rule for $-\!\!*$ [33] can be replaced with the left rule

$$\frac{(X)\,\Gamma \vdash E \qquad (Y)\,\Delta, F \vdash G}{(X, Y)\Gamma, E -\!\!*F \vdash G} -\!\!*\, L.$$

Similarly, the $\forall_{\mathbf{new}}E$ rule can be replaced with the left rule

$$\frac{(X, x)\,\Gamma, E(x) \vdash F}{(X)\,\Gamma, \forall_{\mathbf{new}} x.E \vdash F} \,\forall_{\mathbf{new}} L.$$

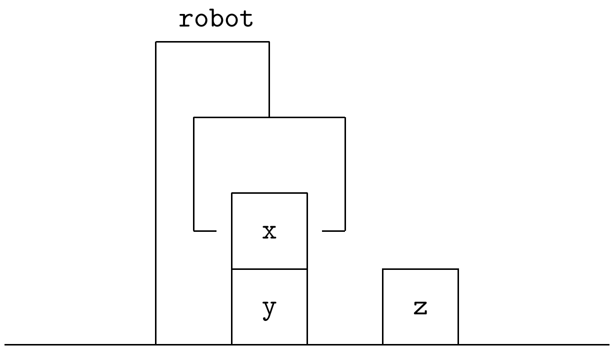
Taking these two together, and adding a substitution, the resolution rule for the multiplicative clause $\forall_{\mathbf{new}} x.G -\!\!*A$ can be expressed as the following derived rule:

$$(1) \qquad \frac{(X)\,\Gamma \vdash G \qquad \dfrac{}{(Y)\,A\sigma \vdash B\sigma}\,Id\ \ (A\sigma = B\sigma)}{(X, Y)\,\Gamma, \forall_{\mathbf{new}} y.G -\!\!*A \vdash B}\,\mathit{Resolution}.$$

A similar argument describes the case for the additive clause.

Whilst the behaviour of the additive clauses is familiar from Prolog-like languages, the multiplicative clauses provided by **BI** have a different interpretation which provides an account of *local variables* and *interference* in logic programming. An example will illustrate these ideas.

Consider a simple version of the famous "blocks world" [40], in which a robotic arm is supposed to manipulate blocks, here labelled x, y and z. Suppose further that we wish to remove from the computation—imagine perhaps that it is a construction process—any components that may be defective. A sketch of a possible program $\mathcal{P}$ follows the diagram:



```
empty.          % the robot's arm is empty
clear(X).       % X supports nothing else
support(X).     % X is supported by the ground
on(Y,X).        % Y is on top of X

defect(X).      % X is defective

...

ready *- ...    % the robot's arm is in position and
                % other conditions hold

defect(X) <- remove(X) & clear(X).   % we write & for
                                     % additive conjunction
remove(X) *- empty * ready.

...
```

Here we intend the logical variable X in the program clause for "remove" to be quantified not by $\forall$, as it would be in a Prolog-like language, but by $\forall_{\mathbf{new}}$. Suppose that quantification in the clause for defect is the usual, Prolog-like $\forall$.

Suppose the block x is possibly defective, i.e., that we have reached the sub-goal *defect*(x). By invoking the clause "defect", in the Prolog-like style, we obtain the sub-goal *remove*(x). Now we are ready to invoke the clause for "remove", as described by the resolution rule in (1). The substitution σ, a fragment of the ultimate answer substitution for the computation, replaces the bound variable X with x, thereby creating a match with the sub-goal *remove*(x). As there are no other variables required, the bunch Y is empty. Thus we get the following instance of resolution:

$$\frac{(W)\ \mathcal{P} \vdash G \qquad \overline{(x)\ \mathit{remove}(x) \vdash \mathit{remove}(x)}\ \mathit{Id}}{(W, x)\ \mathcal{P}, \forall_{\mathbf{new}} X.G \multimap\!\!* \mathit{remove}(X) \vdash \mathit{remove}(\mathrm{x})}\ \mathit{Resolution}.$$

where $G = \mathit{empty} * \mathit{ready}$.

Here it follows immediately that x, the possibly defective block, can take no further part in the computation, which proceeds up the left-hand branch. Had we taken a version of "remove" in the Prolog-like style, x could have persisted on the left-hand branch, thereby requiring additional code to enforce its absence from further constructions.

Thus the use of x in this setting is *local* to the invocation of the clause for "remove" and x cannot be shared by "remove" and other clauses. This is in contrast to the situation arising from additive resolutions, in which the variables used in (fragments of) answer substitutions are global and can be shared. More generally, given two clauses

$$\forall_{\mathbf{new}} x . G \mathrel{-\!\!*} A(x) \qquad \text{and} \qquad \forall_{\mathbf{new}} x . G' \mathrel{-\!\!*} A'(x)$$

the use of $\forall_{\mathbf{new}}$ can be interpreted as enforcing their non-interference, i.e., the fragments of the answer substitution generated by their invocations are disjoint. This behaviour arises partly from the form of $\mathrel{-\!\!*} L$, in which the bunch of variables (X, Y) is distributed to the left and right branches of the proof-search. Semantically, for models in $\mathbf{Set}^{\mathcal{M}}$, this behaviour can be seen in (a case of) the forcing relation for implication in predicate **BI**:

$$\begin{aligned} &(X, Y)\,[u_X, u_Y] \mid m_X \cdot m_Y \models G \mathrel{-\!\!*} A \qquad \text{if and only if} \\ &\qquad \forall n\, \forall Z\, \forall v \in [\![Z]\!]n\; (Z, Y)[v, u_Y] \mid n \cdot m_Y \models G \\ &\qquad\qquad \text{implies } (X, Z)\,[u_X, v] \mid m_X \cdot n \models A. \end{aligned}$$

Here, we require that $Y \vdash G : \mathbf{Prop}$ and $X \vdash A : \mathbf{Prop}$, and we are using the pairing operation of Day's tensor product. The arbitrary nature of the environment $v \in [\![Z]\!]n$, used in the clause for G, reflects the fact that the answer substitution on G's branch of the computation remains to be calculated. The contribution of the quantifiers is thus to regulate the scope of the variables. Again, this is reflected in the semantic clause for the quantifier:

$$\begin{aligned} &(X)\, u \mid m \models \forall_{\mathbf{new}} x \,.\, A \qquad \text{if and only if} \\ &\qquad\qquad \forall n \,.\, \forall d \in Fn \,.\, (X, x)\,[u, d] \mid n \cdot m \models A. \end{aligned}$$

Here the environment for the instantiated clause is formed by taking Day's product of an environment for the whole clause with the interpretation of the substitution determined by the resolution step. These topics will be developed further in [32].

§8. BI in context.

Relation to relevant logic. **BI** should be considered a relevant logic, where we follow Stephen Read [36] in considering "relevant logic" broadly, as referring to a variety of logics which control the use of structural rules, and not necessarily to ones that possess Contraction. In this context, the problem that originally gave rise to bunches involved not implication *per se*, but distribution of additive conjunction over disjunction (see [14]); that is,

$$\varphi \wedge (\psi \vee \chi) \vdash (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$$

and its converse. The difficulty is that distribution fails in standard sequent calculi for logics that omit structural rules, whereas it is present in the main relevant logic **R**. Bunches were used to give a more delicate treatment of structural rules, which was crucial in accounting for distribution.

Distribution of $\wedge$ over $\vee$ has been a source of debate within and without the relevant logic community (see [7] for a lively account). But for our purposes the debate is not a serious issue. Simply, distribution is valid in Kripke resource semantics, which we regard as a natural semantics, and we must accept it when working in a situation justified by the semantics. Similarly, distribution follows at once in the semantics of proofs, using the fact that $E \wedge (-)$ is a left adjoint. This is not to suggest that one wants distribution in all circumstances; the use of linear typing to provide a type structure for domain theory [31] is a prime example of where one does not.

From this historical perspective, the relationship of (propositional) **BI** to **R** can be stated as follows: to get to **BI** from Dunn's sequent calculus $\mathbf{NR}^+$ for positive **R** first remove multiplicative Contraction, then add the unit I of multiplicative conjunction (fusion), and finally the intuitionistic falsity $\perp$ and implication $\rightarrow$. **BI** is also, we believe, closely related to Belnap's scheme of display logic, which gives a method of mixing several logics together [6, 4].

While we acknowledge this historical debt to relevant logic, the *fact* that **BI** could have been obtained in this way says nothing about its *significance*. That is, why would one ever consider modifying a presentation of **R** in this way? Our account of **BI** has, therefore, had somewhat of a revisionist bent. In a sense, the aim has been to show where **BI**, and bunches themselves, *could have* come from.

To recap, we began by noting a curious asymmetry. The decomposition of conjunction (and even disjunction, with the relevantists fission or linear logic's par) into multiplicative and additive components is decades old, but a similarly direct decomposition of implication had not been emphasized. It seemed natural to ask, then, whether such a decomposition could be given; implication, after all, lies at the very heart of logic. We then argued how this question, together with proof-theoretic considerations on implication, leads to the use of two forms of context extension. Evidence that the original question was worth asking was then provided by the semantics, particularly the semantics in terms of doubly closed categories, which display **BI** as a naturally occurring logic.

In contrast, it seems fair to say that (with some exceptions [36, 41]) bunches have appeared mainly as a technical device in the work of relevantists. And from the outside the view is often less charitable. For example, in his history of substructural logics [12], Došen mentions "complications of nonstandard Gentzen systems" and "technically unwholesome properties", comments that bear directly on bunched formalisms, or on features (such as distribution) that necessitate them. A certain regret is even evident in [4], when extra constants or connectives are entered into a Gentzen formalism in order to account for a particular logic.

We would argue, however, that bunches do have a good conceptual status. One may arrive at them from a consideration of the proof-theoretic relationship between conjunction and implication. And an alternative, perhaps more forceful, route is obtained from doubly closed categories, where the adjoint relationship between conjunction and implication leads inexorably to bunched structure. From this point of view bunches are not purely technical, and neither are they particularly unwholesome; rather, bunched structure is, in certain situations, semantically inevitable.

We have also sketched how **BI** admits "resource interpretations", based on sharing. Our results here are more tentative, and it will take time to evaluate their consequences. But we do believe that there are good possibilities for applications, some of which are described in more detail in companion papers and other papers under development. Apart from the specifics of **BI**, the bunch-based approach to controlling structural rules opens up further possibilities for controlling the scope and dynamic extent of information dependencies, in both programs and specifications.

We conclude this part with two general points about relevant logics and bunches. First, we would suggest that useful analyses of existing relevant logics, or identification of interesting variants, might be obtained from attention to categorical proof theory, particularly by focusing on structural relationships between antecedents and logical connectives. And second, other relevant logics might benefit from convincing resource interpretations of proofs (as opposed to just formulæ). Such interpretations are not easy to find, but there is reason to believe that they should be different from those for substructural logics not based on bunches (because of the different semantic properties, exemplified by the distribution).

Relation to linear logic. We have already acknowledged the leading influence of linear logic in the Introduction; here we would like to discuss in detail several of the differences. The point is to aid comprehension by comparison to a well known system, and particularly to go past the surface similarity (where both logics combine multiplicative and additive implications) between them. The comparison is made to intuitionistic linear logic, with "!", and not to smaller fragments.

First, the informal readings given to the connectives are different. Where the linear implication $\varphi \multimap \psi$ is often considered to be about using an argument once, the multiplicative implication $\varphi \mathrel{-\!\!*} \psi$ does not readily admit such a reading. The sharing interpretation, however, provides a reading of connectives consistent with the existence of $-\!\!*$-typed functions which use their arguments multiple times.

Second, bunches are *not* similar to the zoned contexts used in some presentations of linear logic (e.g., [16, 5]). In particular, ";" and "," can be nested in a bunch, and ";" (just like ",") is internalized as a connective in the logic, while the zone marker ";" does not internalize in this way. (Uses of bunches in linear logic, such as [39], do not appear to be aimed at providing any analysis of implication. Moreover, the relationship of such uses to prior work in relevant logic is unclear.)

Third, linear logic admits a decomposition of $\to$ as $!\varphi \multimap \psi$, while **BI** admits no similar decomposition of $\to$ in terms of $-\!\!*$.

Fourth, in common with **R**, **BI** is a distributive logic, where additive conjunction distributes over additive disjunction. Linear logic does not admit distribution.

Fifth, categorical models of linear logic are based on two closed categories, where for **BI** we use a semantics of proofs based on a single category with two closed structures on it. In fact, from a categorical point of view, **BI** appears as a natural progression (which isn't to say the only one). Models of linear logic were first described in terms of a single closed category, and a comonad "!" [42]. Later, the intuitionistic and linear structures were given equal status, in the form of two separate closed categories, one symmetric monoidal and the other cartesian, with functors between them [8, 5]. **BI** also accepts the linear and intuitionistic aspects on an equal footing, but takes the further step of asking that the two closed structures reside in the same category.

Sixth, the Tarski-style models of **BI**, which explain formulæ in terms of a notion of truth, are obtained by combining existing semantics of intuitionistic and relevant logics. We know of no comparable semantics of linear logic, where one combines standard semantics for the multiplicatives and additives.

Seventh, linear logic has a number of vivid computational readings, including: the number-of-uses reading, based on the original coherence space model; an eager and lazy evaluation reading, based on a strict function model; a concurrency reading of proofs [1]; and a logic programming interpretation [35, 18]. In these comparitively early days for **BI**, we have just two related computational readings: one based on imperative programming and the other on logic programming.

Eighth, **BI** has multiplicative quantifiers $\forall_{\mathbf{new}}$ and $\exists_{\mathbf{new}}$, alongside the usual additive quantifiers. Despite many rumblings on this topic over the years, we have not been able to locate a worked-out predicate logic which has multiplicative quantifiers. That is, apart from Ambler's system for the existential only [2], the formulation of which is somewhat more complex than ours. Also relevant is the theory of multiplicative dependent function types in [20], together with its fibrational semantics [19]; incidentally, it can be regarded as relying on a version of bunches appropriate to dependent type theory.

Acknowledgments. We are grateful to Pablo Armelin, Guy McCusker, Peter Schroeder-Heister and Edmund Robinson for useful discussions about this work. The partial support of the UK EPSRC is gratefully acknowledged.

REFERENCES

[1] S. ABRAMSKY, *Computational interpretations of linear logic*, ***Theoretical Computer Science***, vol. 111 (1993), no. 1–2, pp. 3–57.

[2] S. AMBLER, *First order linear logic in symmetric monoidal closed categories*, ***Ph.D. thesis***, University of Edinburgh, 1992.

[3] A. R. ANDERSON and N. D. BELNAP, ***Entailment: the logic of relevance and necessity, volume I***, Princeton University Press, 1975.

[4] A. R. ANDERSON, J. M. DUNN, and N. D. BELNAP, ***Entailment: the logic of relevance and necessity***, vol. II, Princeton University Press, 1992.

[5] A. BARBER and G. D. PLOTKIN, *Dual intuitionistic linear logic*, submitted, October 1997.

[6] N. D. BELNAP, *Display logic*, ***Journal of Philosophical Logic***, vol. 11 (1982), pp. 375–414.

[7] ———, *Life in the undistributed middle*, ***Substructural logics*** (K. Došen and P. Schroeder-Heister, editors), Oxford University Press, 1993, pp. 31–42.

[8] P. N. BENTON, *A mixed linear and non-linear logic: proofs, terms and models*, ***Proceedings of computer science logic '94, Kazimierz, Poland***, Lecture Notes in Computer Science, no. 933, Springer-Verlag, Berlin, 1995.

[9] S. Brookes, M. Main, A. Melton, and M. Mislove (editors), *Mathematical foundations of programming semantics, eleventh annual conference*, Electronic Notes in Theoretical Computer Science, vol. 1, Elsevier Science, 1995, Tulane University, New Orleans, Louisiana.

[10] B. J. DAY, *On closed categories of functors*, ***Reports of the midwest category seminar*** (S. Mac Lane, editor), Lecture Notes in Mathematics, vol. 137, Springer-Verlag, Berlin-New York, 1970, pp. 1–38.

[11] ———, *An embedding theorem for closed categories*, ***Category seminar, Sydney*** (G. M. Kelly, editor), Lecture Notes in Mathematics, no. 420, Springer-Verlag, Berlin-New York, 1974, pp. 55–64.

[12] K. DOŠEN, *A historical introduction to substructural logics*, ***Substructural logics*** (K. Došen and P. Schroeder-Heister, editors), Oxford University Press, 1993, pp. 1–30.

[13] J. M. DUNN, *Conseqution formulation of positive R with co-tenability and t*, in [3].

[14] ———, *Relevant logic and entailment*, ***Handbook of philosophical logic*** (D. Gabbay and F. Guenthner, editors), vol. III: Alternatives in Classical Logic, Synthese Library, no. 166, D. Reidel, Dordrecht, Holland, 1986, pp. 117–224.

[15] J.-Y. GIRARD, *Linear logic*, ***Theoretical Computer Science***, vol. ? (1987), pp. 1–102.

[16] ———, *On the unity of logic*, ***Annals of Pure and Applied Logic***, vol. 59 (1993), pp. 201–217.

[17] J. A. HARLAND, D. J. PYM, and M. WINIKOFF, *Programming in Lygon: an overview*, ***Proceedings of AMAST '96*** (Berlin) (M. Wirsing and M. Nivat, editors), Lecture Notes in Computer Science, no. 1101, Springer-Verlag, 1996, pp. 391–405.

[18] J. S. HODAS and D. MILLER, *Logic programming in a fragment of intuitionistic linear logic*, ***Information and Computation***, vol. 110 (1994), no. 2, pp. 327–365.

[19] S. S. ISHTIAQ and D. J. PYM, *Kripke resource models of a dependently-typed, bunched λ-calculus*, in preparation, available at `http://www.dcs.qmw.ac.uk/~pym`.

[20] ———, *A relevant analysis of natural deduction*, ***Journal of Logic and Computation***, vol. 8 (1998), no. 6, pp. 809–838.

[21] S. C. KLEENE, ***Mathematical logic***, Wiley and Sons, 1968.

[22] S. A. KRIPKE, *Semantical analysis of intuitionistic logic I*, ***Formal systems and recursive functions*** (J. N. Crossley and M. A. E. Dummett, editors), North-Holland, Amsterdam, 1965, pp. 92–130.

[23] Y. LAFONT, *The linear abstract machine*, ***Theoretical Computer Science***, vol. 59 (1988), pp. 157–180.

[24] J. LAMBEK and P. J. SCOTT, ***Introduction to higher-order categorical logic***, Cambridge University Press, Cambridge, England, 1986.

[25] P. MARTIN-LÖF, *On the meanings of the logical constants and the justifications of the logical laws*, ***Technical Report 2***, Scuola di Specializziazione in Logica Matematica, Università di Siena, 1982.

[26] D. MILLER, *A logical analysis of modules in logic programming*, ***Journal of Logic Programming***, vol. 6 (1989), no. 1 and 2, pp. 79–108.

[27] D. MILLER, G. NADATHUR, F. PFENNING, and A. ŠČEDROV, *Uniform proofs as a foundation for logic programming*, ***Annals of Pure and Applied Logic***, vol. 51 (1991), pp. 125–157.

[28] P. W. O'HEARN, *Resource interpretations, bunched implications and the $\alpha\lambda$-calculus*, in preparation, preliminary version to appear in ***Typed λ-calculus and applications***, Lecture Notes in Computer Science, 1999.

[29] P. W. O'HEARN, A. J. POWER, M. TAKEYAMA, and R. D. TENNENT, *Syntactic control of interference revisited*, to appear in ***Theoretical Computer Science***, preliminary version in [9] and in [30].

[30] P. W. O'Hearn and R. D. Tennent (editors), ***Algol-like languages***, Birkhäuser, Boston, 1997, two volumes.

[31] G. D. PLOTKIN, *Type theory and recursion*, slides for Scottfest talk, 1993.

[32] D. J. PYM, *Logic programming with bunched implications*, ***Electronic Notes in Theoretical Computer Science***, vol. 17 (1998), extended abstract, 24 pp.

[33] ———, *The semantics and proof theory of the logic of bunched implications, II*: *predicate* **BI**, available on the web at `http://www.dcs.qmw.ac.uk/~pym`, 1998.

[34] ———, *The semantics and proof theory of the logic of bunched implications, I*: *Propositional* **BI**, available on the web at `http://www.dcs.qmw.ac.uk/~pym`, 1998.

[35] D. J. PYM and J. A. HARLAND, *A uniform proof-theoretic investigation of linear logic programming*, ***Journal of Logic and Computation***, vol. 4 (1994), pp. 175–207.

[36] S. READ, ***Relevant logic: a philosophical examination of inference***, Basil Blackwell, 1987.

[37] J. C. REYNOLDS, *Syntactic control of interference*, ***Conference record of the fifth annual acm symposium on principles of programming languages*** (New York), ACM, January 1978, also in [30], pp. 39–46.

[38] ———, *The essence of Algol*, ***Algorithmic languages: Proceedings of the international symposium on algorithmic languages*** (Amsterdam) (J. W. de Bakker and J. C. van Vliet, editors), North-Holland, October 1981, also in [30], pp. 345–372.

[39] P. RUET and F. FAGES, *Concurrent constraint programming and mixed non-commutative linear logic*, ***Proceedings of CSL '97*** (Berlin), Lecture Notes in Computer Science, no. 1414, Springer-Verlag, 1997.

[40] E. D. SACERDOTI, ***A structure for plans and behaviour***, Elsevier North Holland, 1977.

[41] P. SCHROEDER-HEISTER, *Structural frameworks, substructural logics and the role of elimination inferences*, ***Logical frameworks*** (G. Huet and G. Plotkin, editors), Cambridge University Press, 1991, pp. 385–403.

[42] R. A. G. SEELY, *Linear logic, $*$-autonomous categories and cofree coalgebras*, ***Categories in computer science and logic*** (Providence, Rhode Island) (J. W. Gray and A. Scedrov, editors), Contemporary Mathematics, no. 92, American Mathematical Society, 1989,

pp. 371–382.

[43] A. S. Troelstra, *Tutorial on linear logic*, ***Substructural logics*** (K. Došen and P. Schroeder-Heister, editors), Oxford University Press, 1993, pp. 327–355.

[44] A. Urquhart, *Semantics for relevant logics*, this Journal, vol. 49 (1972), pp. 1059–1073.

DEPARTMENT OF COMPUTER SCIENCE
QUEEN MARY & WESTFIELD COLLEGE
UNIVERSITY OF LONDON
LONDON E1 4NS, UK
E-mail: ohearn@dcs.qmw.ac.uk

DEPARTMENT OF COMPUTER SCIENCE
QUEEN MARY & WESTFIELD COLLEGE
UNIVERSITY OF LONDON
LONDON E1 4NS, UK
E-mail: pym@dcs.qmw.ac.uk