Resource semantics: logic as a modelling technology

David Pym, UCL and The Alan Turing Institute, London

The Logic of Bunched Implications (BI) was introduced by O'Hearn and Pym. The original presentation of BI emphasised its role as a system for formal logic (broadly in the tradition of relevant logic) that has some interesting properties, combining a clean proof theory, including a categorical interpretation, with a simple truth-functional semantics. BI quickly found significant applications in program verification and program analysis, chiefly through a specific theory of BI that is commonly known as 'Separation Logic'. We survey the state of work in bunched logics — which, by now, is a quite large family of systems, including modal and epistemic logics and logics for layered graphs — in such a way as to organize the ideas into a coherent (semantic) picture with a strong interpretation in terms of resources. One such picture can be seen as deriving from an interpretation of BI's semantics in terms of *resources*, and this view provides a basis for a systematic interpretation of the family of bunched logics, including modal, epistemic, layered graph, and process-theoretic variants, in terms of resources. We explain the basic ideas of resource semantics, including comparisons with Linear Logic and ideas from economics and physics. We include discussions of BI's λ-calculus, of Separation Logic, and of an approach to distributed systems modelling based on resource semantics.

1. INTRODUCTION

The Logic of Bunched Implications (BI) was introduced by O'Hearn and Pym [89; 95]. The original presentation of BI emphasised its role as a system for formal logic (broadly in the tradition of relevant logic) that has some interesting properties, combining a clean proof theory, including a categorical interpretation, with a simple truth-functional semantics. BI has since found significant applications in program verification and program analysis, chiefly through a specific theory of BI that is commonly known as 'Separation Logic' [65; 99; 61].

The purpose of the article is to survey the state of work in bunched logics — which, by now, is a quite large family of systems — in such a way as to organize the ideas into a coherent (semantic) picture. One such picture can be seen as deriving from an interpretation of BI's semantics in terms of *resources*. How does this interpretation arise? We need to begin, to set the scene, with a few remarks on the logic itself. We'll just talk about conjunction for now.

As we'll see below, BI can be understood proof-theoretically: indeed, that is where its name comes from. Consider — eliding for now lots of details, such as the Exchange rule for swapping the order of formulae — the following two forms of conjunction, $\wedge_1$ and $\wedge_2$, given in terms of left- and right-rules of a single-conclusion sequent calculus: the 'additive' form,

$$\frac{\Gamma, \phi_1, \phi_2, \Delta \vdash \psi}{\Gamma, \phi_1 \wedge_1 \phi_2, \Delta \vdash \psi} \;\wedge_1 \mathrm{L} \qquad \frac{\Gamma \vdash \phi_1 \quad \Gamma \vdash \phi_2}{\Gamma \vdash \phi_1 \wedge_1 \phi_2} \;\wedge_1 \mathrm{R},$$

and the 'multiplicative' form,

$$\frac{\Gamma, \phi_1, \phi_2, \Delta \vdash \psi}{\Gamma, \phi_1 \wedge_2 \phi_2, \Delta \vdash \psi} \;\wedge_2 \mathrm{L} \qquad \frac{\Gamma_1 \vdash \phi_1 \quad \Gamma_2 \vdash \phi_2}{\Gamma_1, \Gamma_2 \vdash \phi_1 \wedge_2 \phi_2} \;\wedge_2 \mathrm{R}.$$

In the presence of the structural rules of Weakening (W) and Contraction (C),

$$\frac{\Gamma_1, \Gamma_2 \vdash \psi}{\Gamma_1, \phi, \Gamma_2 \vdash \psi} \;\mathrm{W} \qquad \text{and} \qquad \frac{\Gamma_1, \phi, \phi, \Gamma_2 \vdash \psi}{\Gamma_1, \phi, \Gamma_2 \vdash \psi} \;\mathrm{C},$$

the two forms, $\wedge_1$ and $\wedge_2$, are equivalent (proof-theoretically, they are interderivable). In the absence of these rules, the two conjunctions are distinct.

In BI, we choose to have both forms, and must set up a proof system to handle them simultaneously. To do this, we move from sequents in which the antecedents are finite lists of formulae to ones in which they are finite trees with internal vertices labelled with either a comma, ',', or a semi-colon, ';', and with leaves labelled with formulae. These antecedents — which are required to satisfy certain equivalences [89; 95], including congruence of substitution and the Exchange rule, are called *bunches* — and they have a history in relevant logic (see, for example, [4; 5; 98]).

The comma and the semicolon are both operations that build antecedents, just like the regular comma in the classical or intuitionistic sequent calculus, but we can now associate the structural rules of Weakening and Contraction with one of them (semicolon) and not the other:

$$\frac{\Gamma(\phi) \vdash \chi}{\Gamma(\phi\,;\,\psi) \vdash \chi}\ \mathrm{W} \quad \text{and} \quad \frac{\Gamma(\phi\,;\,\phi) \vdash \psi}{\Gamma(\phi) \vdash \psi}\ \mathrm{C},$$

but not for the comma.

Notice that we are now working with 'deep' rules, in which we look inside the structure of the tree, and we will write $\Gamma(\Delta)$ to denote that Δ is a sub-bunch of Γ. We will give more details of the structure of bunches below. For, it will suffice to observe that we can now write the two forms of conjunction, which we'll now call $\wedge$ and $*$, as follows:

$$\frac{\Gamma(\phi_1, \phi_2) \vdash \psi}{\Gamma(\phi_1 \wedge \phi_2) \vdash \psi}\ \wedge\mathbf{L} \qquad \frac{\Gamma_1 \vdash \phi_1 \quad \Gamma_2 \vdash \phi_2}{\Gamma_1\,;\,\Gamma_2 \vdash \phi_1 \wedge_1 \phi_2}\ \wedge_1\mathbf{R},$$

noting that if $\Gamma_1 = \Gamma_2$, then Contraction can be applied in the conclusion of the rule and so the additive form of the rule can be recovered, and

$$\frac{\Gamma(\phi_1, \phi_2) \vdash \psi}{\Gamma(\phi_1 * \phi_2) \vdash \psi}\ *\mathbf{L} \qquad \frac{\Gamma_1 \vdash \phi_1 \quad \Gamma_2 \vdash \phi_2}{\Gamma_1, \Gamma_2 \vdash \phi_1 \wedge_2 \phi_2}\ *\mathbf{R}.$$

noting that even if $\Gamma_1 = \Gamma_2$, the contraction rule is not applicable in the conclusion of the rule. Each of these conjunctions — the first one is intuitionistic; the second is linear — has an associated implication. The first one is intuitionistic implication, written $\rightarrow$; the second one is linear implication, written $-\!\!*$, also know as 'magic wand'.

From a semantic perspective, there are two ways to handle BI: by interpreting the proof theory categorically and by giving a truth-functional semantics, and they are intimately related. To set the scene, we'll consider an elementary, Kripke-style truth-functional semantics [73; 35; 46]. We will need two main ingredients to set this up. First, because we need to interpret the intuitionistic connectives, we need a set of worlds that is preordered (i.e., the order is reflexive and transitive). Second, because we need to interpret the linear connectives, we need a monoidal operation [7]. The basic solution is to work with a structure $\mathbf{M}$ consisting in a set M of worlds m that enjoys a preorder $\sqsubseteq$ and monoidal operation $\circ$, with unit e, subject to a coherence condition that we will mention below:

$$\mathbf{M} = (M, \sqsubseteq, \circ, e)$$

An example of such a structure is given by the natural numbers, including 0, ordered by less-than-or-equals and with monoidal composition given by addition: $(\mathbb{N}, \leq, +, 0)$. Later, we'll see the need for variations such as partial and non-commutative monoids.

With this set up, we can now interpret our two conjunctions and implications as follows:

$$\begin{array}{rl} m \models \phi_1 \wedge \phi_2 & \text{iff } m \models \phi_1 \text{ and } m \models \phi_2 \\ m \models \phi \rightarrow \psi & \text{iff for all } n \text{ s.t. } n \sqsubseteq m,\ n \models \psi \end{array}$$

and

$$\begin{array}{rl} m \models \phi_1 * \phi_2 & \text{iff there are } n_1 \text{ and } n_2 \text{ s.t. } m \sqsubseteq n_1 \circ n_2 \text{ and} \\ & n_1 \models \phi_1 \text{ and } n_2 \models \phi_2 \\ m \models \phi -\!\!* \psi & \text{iff for all } n \text{ s.t. } n \models \phi,\, m \circ n \models \psi \end{array}$$

It is this semantics that gives rise to use of BI and its associated systems as bases for modelling technologies, with Separation Logic and its applications being leading a leading example.

In Section 2, we outline the resource interpretation of BI's semantics, starting from some basic observations about the concept of resource. In Section 3, we consider some of the ideas in substructural logic that form the background to BI. As the rest of this article is mainly focussed on semantics, here we balance things with a more proof-theoretic perspective. In Section 4, we describe BI in more detail, summarizing its proof theory, its categorical semantics, and its truth-functional semantics. We also introduce the ideas of systems of *labelled tableaux*, which are useful tool in the metatheory of bunched logics. We also mention BI's lambda calculus, $\alpha\lambda$, and its interpretation in resource semantics. In Section 5, we provide a summary of Separation Logic from the perspective of resource semantics. In Section 6, we sketch modal epistemic extensions of BI with interpretations that are based on resource semantics. In Section 7, we explain how the family of bunched logics includes systems that have very weak structural properties — based on structures that are neither commutative nor associative — but which nevertheless are interesting both from the logical point view, where they help us to establish a general framework for bunched logics, and from a modelling perspective, using their semantics in layered graphs. In Section 8, we explain how the ideas in the previous sections can be brought together to provide a basis for a theory and, indeed, implementation of a framework for modelling distributed systems, based on ideas of location, resource, and process. Finally, in Section 9, we summarize the story of BI and its associated systems — including logics for layered graphs, modal variants, and process calculi — as a basis for a modelling technology. concluding with a brief summarizing discussion of logic as a modelling technology.

This article presumes the reader has some familiarity with a range of basic supporting ideas. These include the basics of logic, such as proof systems and truth-functional semantics, the basics of categorical logic, the basic idea of program verification, and the basic ideas of concurrent and distributed systems and approaches to modelling them. References are intended to be illustrative and helpful rather than comprehensive (which would require a vast number). The style of the article is *informal* throughout.

The content and organization of the article are based on several presentations given by the author: first, an invited lecture at the Second SYSMICS Workshop in Vienna, Austria, 26–28 February 2018; second, a tutorial of four lectures at the SYSMICS Summer School, Les Diablerets, Switzerland, 22–26 August 2018; third, an invited lecture an the Dagstuhl Seminar on Logics for Dependence and Independence, Schloss Dagstuhl, Germany, 14–18 January 2019. The author is grateful to the organizers of those meetings for their kind invitations to speak. This article draws directly and explicitly on papers by myself, my colleagues, and my PhD students about BI, its semantics, and its applications in program verification and systems. All such work is fully cited throughout the article.

2. RESOURCE SEMANTICS BASICS

We have begun by mentioning BI's logical motivation, which can be seen as lying within a combination of programmes of research in relevant logic [4; 5; 98], intuitionistic logic [35; 73; 10; 46], categorical logic [81; 102; 78], and program logic [63; 6; 80], which mathematical, philosophical, and computational logical methods all exercising some influence over its development.

In this section, we consider an alternative motivation — wholly consistent with the logical motivation — that is about *logical systems modelling*. For these purposes, we consider 'systems' in the sense of an abstraction of the concept of a 'distributed system' in the theory of computer systems, as described, for example, in the work of Coulouris, Dollimore, Kindberg, and Blair [31].

This view of the classical theory of distributed systems provides a rigorous conceptual basis for our modelling perspective, which can be conveniently abstracted to describe systems in terms of collections of the following:

- interconnected *locations*, at which are situated
- *resources*, relative to which
- *processes* execute — consuming, creating, moving, and otherwise manipulating resources as they evolve — and so deliver a system's services [28; 27].

Distributed systems, as described in this way, do not exist in isolation, but within *environments* with which they interact. A system's environment is both a source of events, that are incident upon the system, and the recipient of events caused by the execution of the system's processes. For the purposes of this article, we will be concerned with the structural aspects of systems models — locations, resources, and processes. In [27; 17], it is explained how environment is added this picture through the use of probability distributions to capture the incidence of events across the boundary of a system model.

In the course of this article, we will discuss all of location, resource, and process in some detail. For the moment, however, we concentrate on resources as a basis for BI's semantics.

Conceptually, *resource semantics* begins with a simple axiomatization of resource. Starting with a given homogeneous set of resource elements — for example, bags of fruit, units of currency, or computer memory – we expect the following properties:

- to be able to *combine* two units of the given type of resource to form a new unit of that type of resource;
- to be able to *compare* (using either a simple equality or an ordering) two units of a given type of resource;
- that combination and comparison should be appropriately compatible.

Mathematically, this basic set-up is captured by pre-ordered partial monoids of resources (PRMs), defined as follows [89; 52]:

$$R = (R, \circ, e, \sqsubseteq),$$

where R is a set of resource elements, $\sqsubseteq$ is a pre-order (we write $=$ for $\sqsubseteq \cap \sqsupseteq$) and $\circ$ is a monoidal composition with unit e, subject to the 'functoriality' coherence condition that, where defined,

$$\text{if } r \sqsubseteq s \text{ and } r' \sqsubseteq s' \text{ , then } r \circ r' \sqsubseteq s \circ s'$$

An example of such a structure is provided by the natural numbers, including zero (for which the monoid operation happens to be total):

$$(\mathbb{N}, +, 0, \leq)$$

with addition and less-than-or-equal.

Some basic examples of resources that fit into this framework include the following:

- Money and homogenous commodities — these are essentially the natural numbers, as above;

- Tuples of commodities, with pointwise orderings;
- Petri nets — see [97; 20], and
- Computer memory — the 'stack–heap' model of Separation Logic [65; 110].

The last of these, computer memory, makes essential use of partiality.

This basic axiomatization has proved remarkably robust. It supports Separation Logic — the basic ideas of which we shall describe in more detail later — and its developments [65; 99; 86; 61; 87] in a vast subsequent literature. It also supports resource interpretations of modal logics based on BI [19; 20; 50] modalities and illuminates connections with a range of other influential logical perspectives, including Dependence Logic [108; 1], and quantum information theory [24; 39]. Simon Docherty's recent PhD thesis [39] provides an excellent discussion of the scope of resource semantics in a generalized setting.

3. SUBSTRUCTURAL LOGIC

There is a long history of the study of systems of logic — substructural logics — which have weaker structural properties than logics based on classical or even intuitionistic logic (see, for example, [4; 5; 98; 54; 101; 77; 101], to name just a few sources). In this section, we explain BI in this context.

It is perhaps most convenient to characterize this work using proof-theoretic representations consequence. Relevant, or relevance, logic is an attempt to avoid the 'paradoxes' of material (and strict) implication, such as

$$\vdash \phi \to (\psi \to \phi) \qquad \text{and} \qquad \vdash (\phi \to \psi) \vee (\psi \to \chi)\,.$$

That these consequences are provable in classical logic follows from the structural rule of Weakening, which is expressed in the sequent calculus as as the pair of left and right rules

$$\frac{\Gamma \vdash \Delta}{\Gamma, \phi \vdash \Delta}\ \mathrm{WL} \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \phi, \Delta}\ \mathrm{WR}$$

that allow 'irrelevant' assumptions to be introduced to proofs.[1]

In proof-theoretic terms, relevant logic [11; 98] refers to systems that reject Weakening. The structural rule of Contraction, which allows the removal of duplication assumptions, is expressed in the sequent calculus

$$\frac{\Gamma, \phi, \phi \vdash \Delta}{\Gamma, \phi \vdash \Delta}\ \mathrm{CL} \qquad \text{and} \qquad \frac{\Gamma \vdash \phi, \phi, \Delta}{\Gamma \vdash \phi, \Delta}\ \mathrm{CR}\,.$$

This rule can also be dropped, so giving a 'purely relevant' system in which the number of uses of a formula is tracked exactly in provable consequences.

Girard's Linear Logic (LL) [54] is a substructural logic that represents Weakening and Contraction, within an otherwise purely relevant system, using S4-like modalities, or exponentials, ! and ?, with left and right rules of the form

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, !\phi \vdash \Delta}\ !\mathrm{L} \qquad \text{and} \qquad \frac{?\Gamma \vdash \phi, ?\Delta}{?\Gamma \vdash ?\phi, ?\Delta}\ ?\mathrm{L}$$

and

$$\frac{!\Gamma \vdash \phi, ?\Delta}{!\Gamma \vdash !\phi, ?\Delta}\ !\mathrm{R} \qquad \text{and} \qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash ?\phi, \Delta}\ ?\mathrm{R}$$

These modalities are dual via linear negation: $(!\phi)^{\perp} = ?\phi^{\perp}$ and $(?\phi)^{\perp} = !\phi^{\perp}$.

[1] For the purposes of this section, all calculi are assumed to admit the Exchange (permutation) rule.

Then the structural rules arise as

$$\frac{\Gamma \vdash \Delta}{\Gamma, !\phi \vdash \Delta} \ \mathrm{WL} \qquad \text{and} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?\phi, \Delta} \ \mathrm{WR}$$

and

$$\frac{\Gamma, !\phi, !\phi \vdash \Delta}{\Gamma, !\phi \vdash \Delta} \ \mathrm{CL} \qquad \text{and} \qquad \frac{\Gamma \vdash ?\phi, ?\phi, \Delta}{\Gamma \vdash ?\phi, \Delta} \ \mathrm{CR}\,.$$

Alongside this classical, multiple-conclusioned version of Linear Logic (CLL) comes an intuitionistic version. The single-conclusioned calculus for intuitionistic Linear Logic (ILL) employs (therefore) just the single modality !, with the sequent calculus rules corresponding to those above obtained by erasing all of the right-hand side formulae other than the leftmost.

With this control of the Weakening and Contraction, Linear Logic — like relevant logics [4; 5; 98] — is able to distinguish between *multiplicative* and *additve* connectives (here we use these terms just to denote the forms inference rules, not in Linear Logic's semantic sense). For example, in ILL, we have the multiplicative conjunction $\otimes$ and the additive conjunction $\&$ as right rules

$$\frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma, \Delta \vdash \phi \otimes \psi} \ \otimes\mathrm{R} \qquad \text{and} \qquad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \,\&\, \psi} \ \&\mathrm{R}$$

and left rules

$$\frac{\Gamma, \phi_1, \phi_2, \Gamma' \vdash \psi}{\Gamma, \phi_1 \otimes \phi_2, \Gamma' \vdash \psi} \ \otimes\mathrm{L} \qquad \text{and} \qquad \frac{\Gamma, \phi_i, \Gamma' \vdash \psi}{\Gamma, \phi_1 \& \phi_2, \Gamma' \vdash \psi} \ (i = 1, 2) \quad \&\mathrm{L}.$$

In the presence of unrestricted Weakening and Contraction, the two forms of conjunction are inter-derivable.

Along with the multiplicative conjunction comes ILL's implication, $\multimap$, with the following left and right rules:

$$\frac{\Gamma \vdash \phi \quad \psi, \Gamma' \vdash \chi}{\Gamma, \phi \multimap \psi, \Gamma' \vdash \chi} \ \multimap\mathrm{L} \qquad \text{and} \qquad \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \multimap \psi} \ \multimap\mathrm{R}$$

The key thing to notice here is that the only sensible way to write a different, additive, implication is to use the modality, !, as follows:

$$\frac{\Gamma, !\phi \vdash \psi}{\Gamma \vdash !\phi \multimap \psi}$$

and this, in fact, gives ILL's representation — often known as 'Girard's translation' — of intuitionistic implication:

$$\phi \to \psi \ = !\phi \multimap \psi.$$

4. THE LOGIC OF BUNCHED IMPLICATIONS

Linear Logic's use of the modalities, ! and ?, is not the only way to control the use of the structural rules. In this section, we will see how an alternative approach, which employs a richer structure for sequents, results in a very different logic. Starting from this proof-theoretic perspective, we will sketch a categorical semantics, and then summarize BI's truth-functional semantics.

Having summarized BI's basic proof-theoretic and semantic set-ups, in subsequent sections we can then consider how it — and, in particular, its semantics and its associated λ-calculus, $\alpha\lambda$ — can be understood as a basic theory of resource and, consequently, as a systems modelling tool.

4.1. Proof Theory

We have seen how Linear Logic controls the use of the structural rules of classical and intuitionistic logic using modalities. There is, as we have mentioned, a very different way of handling them. In terms of sequent calculus, in BI we employ a richer underlying sequential structure in which the antecedent Γ in a sequent $\Gamma \vdash \phi$ is structured not as finite list of formulae, but rather as a finite tree of formulae in which the leaves of the tree are labelled with formulae and the internal vertices of the tree are labelled with one of two combinators, ';' and ',', which construct antecedents. These structures are called *bunches*.

The semi-colon admits both Weakening and Contraction, and so corresponds to the list-constructor in intuitionistic sequents,

$$\frac{\Gamma(\phi) \vdash \chi}{\Gamma(\phi\,;\,\psi) \vdash \chi}\ \mathrm{W} \quad \text{and} \quad \frac{\Gamma(\phi\,;\,\phi) \vdash \psi}{\Gamma(\phi) \vdash \psi}\ \mathrm{C}$$

whereas the comma admits neither, and so corresponds to the list-constructor in ILL. The weakening and contracting can, of course, be generalized to be bunches themselves.

Bunches are required to satisify an equivalence ($\equiv$) that includes the commutative monoid equations for ';' and ',' and a congruence property that if $\Delta \equiv \Delta'$, then $\Gamma(\Delta) \equiv \Gamma(\Delta')$.

With this set-up, it is easy to define both additive connectives, corresponding to the intuitionistic connectives, and multiplicatives, corresponding to those of MILL. For example,

$$\frac{\Gamma(\phi_1\,,\,\phi_2) \vdash \psi}{\Gamma(\phi_1 * \phi_2) \vdash \psi}\ {*}\mathrm{L} \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma\,,\,\Delta \vdash \phi * \psi}\ {*}\mathrm{R}$$

$$\frac{\Gamma(\phi_1\,;\,\phi_2) \vdash \psi}{\Gamma(\phi_1 \wedge \phi_2) \vdash \psi}\ \wedge\mathrm{L} \qquad \frac{\Gamma \vdash \phi \quad \Delta \vdash \psi}{\Gamma\,;\,\Delta \vdash \phi \wedge \psi}\ \wedge\mathrm{R}$$

and

$$\frac{\Gamma \vdash \phi \quad \Delta(\psi) \vdash \chi}{\Delta(\Gamma, \phi \mathrel{-\!\!*} \psi) \vdash \chi}\ {-\!\!*}\,\mathrm{L} \qquad \frac{\Gamma\,,\,\phi \vdash \psi}{\Gamma \vdash \phi \mathrel{-\!\!*} \psi}\ {-\!\!*}\,\mathrm{R}$$

$$\frac{\Gamma \vdash \phi \quad \Delta(\psi) \vdash \chi}{\Delta(\Gamma; \phi \to \psi) \vdash \chi}\ {\to}\mathrm{L} \qquad \frac{\Gamma\,;\,\phi \vdash \psi}{\Gamma \vdash \phi \to \psi}\ {\to}\mathrm{R}$$

Additive disjunction, as well as all the logical units — $\top$, $\bot$, and I (for $*$) — can also be handled in this way, and many extensions (modal, action, epistemic, quantified) of this set-up are possible.

BI's sequent calculus satisfies *cut-elimination*: if $\Theta \vdash \phi$ is provable using the Cut rule,

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

then $\Theta \vdash \phi$ is provable without using the Cut rule.

Other proof systems are available for BI, including Prawitz-style natural deduction [94], Hilbert-type systems, Display Calculi [15], and Labelled Tableaux.

BI contains MILL and IL as sublogics; the connections can be stated precisely as follows:

- BI is conservative over IL: that is, $\phi_1; \ldots; \phi_n \vdash \phi$, where each ϕ_i and ϕ is a formula containing only additives, is provable in BI iff it is provable in IL;
- BI is conservative over MILL: that is, $\phi_1, \ldots, \phi_n \vdash \phi$, where each ϕ_i and ϕ is a formula containing only multiplicatives, is provable in BI iff $\phi_1^*, \ldots, \phi_n^* \vdash \phi^*$, where $(-)^*$ replaces each $*$ by $\otimes$ and each $-\!\!*$ by $\multimap$, is provable in MILL.

Note that conservativity does not extend to MAILL (MILL extended with LL's additive conjunction and disjunction). The reason is that BI, just as in IL, admits distribution of additive conjunction over additive disjunction — that is, $\phi \wedge (\psi \vee \chi) \vdash (\phi \wedge \psi) \vee (\phi \wedge \chi)$ — but MAILL does not (for $\&$ and $\oplus$).

4.2. Categorical Semantics

BI's proof theory can also be understood algebraically, in the tradition of categorical semantics [81; 102; 78]. BI's proofs are interpreted in *bicartesian doubly closed categories*, or DCCs, for short here [89; 95; 85]. A category is said to be *doubly closed* if it enjoys two symmetric monoidal closed structures [7]. A doubly closed category is cartesian if one of its closed structures is cartesian, and is bicartesian if it also has finite coproducts.

The cartesian structure is used to interpret the additive fragment of BI and the other symmetric monoidal closed structure is used to interpret the multiplicative fragment. To see this, consider that the two adjunctions, $[H * E, F] \cong [H, E -\!\!* F]$ and $[H \times E, F] \cong [H, E \to F]$, correspond to the rules for implication,

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi -\!\!* \psi} \quad \text{and} \quad \frac{\Gamma; \phi \vdash \psi}{\Gamma \vdash \phi \to \psi}$$

For a given DCC, assume an interpretation $[\![\mathrm{p}]\!]$ of BI's propositional letters, extended to formulae and contexts in the obvious way. Note the interpretation of $\vee$L (or $\vee$ elimination in a natural deduction presentation) needs distribution of both $\times$ and $*$ over $+$ — but both are left adjoints, and so preserve colimits. Soundness and completeness then arise as follows: $\Gamma \vdash \phi$ is provable in BI iff, for every bicartesian DCC $\mathcal{D}$ and every interpretation $[\![-]\!]$, the homset $\mathcal{D}[[\![\Gamma]\!], [\![\phi]\!]]$ is non-empty.

This abstract set-up is all very well, but one might ask whether there are natural examples of bicartesian DCCs. Happily, they are plentiful:

- $\mathbf{Cat}$, the category of small categories, via Gray's tensor product (see ncatlab.org);
- $\mathbf{Set} \times \mathbf{Set}$: SMC structure,with unit $I = (1, 0)$, given as
  - $I = (1, 0)$
  - $(E_0, E_1) \otimes (F_0, F_1) = ((E_0 \times F_0) + (E_1 \times F_1), (E_0 \times F_1) + (E_1 \times F_0))$
  - $(E_0, E_1) \multimap (F_0, F_1) = ((E_0 \to F_0) \times (E_1 \to F_1), (E_0 \to F_1) \times (E_1 \to F_0))$.
- More generally, presheaves $\mathbf{Set}^{\mathcal{C}^{op}}$ for monoidal $\mathcal{C}$ (if $\mathcal{C}$ is symmetric, so is $\mathbf{Set}^{\mathcal{C}^{op}}$): $\mathbf{Set}^{\mathcal{C}^{op}}$ is bicartesian closed, and Day's tensor product construction [37; 38] gives another SMC structure: the co-end gives the product

$$(E \otimes F)X = \int^{Y,Y'} EY \times FY' \times \mathcal{C}[X, Y \otimes Y']$$

  and its right adjoint, which is an end, gives the hom:

$$(E \multimap F)X = \int_Y \mathbf{Set}[EY, F(X \otimes Y)] \cong \mathbf{Set}^{\mathcal{C}^{op}}[E(-), F(X \otimes -)].$$

This semantics provides a simple and convenient way to see a key distinction between MILL and BI. $\mathbf{Set} \times \mathbf{Set}$, as mentioned above, helps us to understand what is going on:

$$
\begin{array}{rll}
r \models \mathrm{p} & \text{iff} & r \in \mathcal{V}(\mathrm{p}) \\
r \models \bot & \text{never} & \\
r \models \top & \text{always} & \\
r \models \phi \vee \psi & \text{iff} & r \models \phi \text{ or } r \models \psi \\
r \models \phi \wedge \psi & \text{iff} & r \models \phi \text{ and } r \models \psi \\
r \models \phi \to \psi & \text{iff} & \text{for all } s \sqsubseteq r,\ s \models \phi \text{ implies } s \models \psi \\
\\
r \models I & \text{iff} & r \sqsubseteq e \\
r \models \phi * \psi & \text{iff} & \text{there are worlds } s \text{ and } t \text{ such that} \\
 & & r \sqsubseteq (s \cdot t)\!\downarrow \text{ and } s \models \phi \text{ and } t \models \psi \\
r \models \phi \mathrel{-\!\!*} \psi & \text{iff} & \text{for all } s \text{ such that } (r \cdot s)\!\downarrow \text{ and } s \models \phi, \\
 & & r \cdot s \models \psi.
\end{array}
$$

Fig. 1. A simple truth-functional semantics for BI

- $\mathbf{Set} \times \mathbf{Set}$ is a non-degenerate model: I is not a terminal object and $*$ is not a cartesian product
- There are no maps in the model from 1 to I
- $(0,1) \to (1,0) = (1,0)$ and $(0,1) \mathrel{-\!\!*} (1,0) = (0,1)$ (just unpack the definition above). Hence $\mathrel{-\!\!*}$ and $\to$ are distinct.
- Now we can see a key distinction between MILL and BI:
  - There is no endofunctor $! : \mathbf{Set} \times \mathbf{Set} \to \mathbf{Set} \times \mathbf{Set}$ which admits an isomorphism $!E \multimap F \cong E \to F$, corresponding to Girard's translation of intuitionistic logic into intuitionistic linear logic, $\phi \to \psi =!\phi \multimap \psi$;
  - To see this, consider that $(1,0) \to (2,2) = (2,1)$, but, for any E, $E \multimap (2,2) = (X,Y)$, where the sets X and Y have the same cardinality.

4.3. Truth-functional Semantics

Logic, however, is about much more than proof theory. Critical — in my view at least — to understanding a system of logic is a truth-functional semantics, which — again, in my view at least — should be as directly motivated and as simply expressed as possible. Although the distinction between the proof theory of BI and that of ILL is quite clear and compelling, it is perhaps in the motivation — in terms of resources — and expression — as a very straightforward satisfaction relation — of BI's truth-functional semantics that BI's value can be most clearly seen.

BI's partially defined monoid (or PDM) semantics is defined as follows [89; 95; 52]: $(R, \cdot, e, \sqsubseteq)$ a partially ordered commutative partial monoid, $\mathcal{V}$ an interpretation of propositional letters in $\wp(R)$, and $r, s \in R$ such that the satisfaction relation given in Figure 1 holds. This semantics requires the persistence, or (Kripke) monotonicity, property: if $r \models \phi$ and $s \sqsubseteq r$, then $s \models \phi$.

The resource interpretation, also know as the *sharing interpretation*, of this semantics is now quite clear:

- the components of additive conjunction ($\wedge$), disjunction ($\vee$), and implication ($\to$) may share resources, whereas
- the components of multiplicative conjunction ($*$) and implication ($\mathrel{-\!\!*}$) *do not* share resources.

Boolean BI is the variant of this logic in which the additives are taken to be classical. In the semantics above, we replace the intuitionistic implication with the usual classical one,

$$r \models \phi \rightarrow \psi \;\text{ iff }\; r \models \phi \text{ implies } r \models \psi$$

and take the ordering $\sqsubseteq$ to be equality. Boolean BI is the basis of Separation Logic; this is discussed extensively below. The metatheory of Boolean BI has been studied by Larchey-Wendling [79].

A multiplicative disjunction and a multiplicative negation can also be defined. These ideas are discussed in, for example, the work of Brotherston and Villard [16], which considers the ideas of 'classical BI' and 'sub-classical Boolean BI' and develops their theory in some detail. For example, the algebra of worlds can be enriched with a notion of a 'dualizing' operation on worlds, $-\,r$, such that $-\,-\,r = r$, so that a multiplicative negation, $\sim \phi$, and a multiplicative disjunction, $\phi + \psi$, can be defined, essentially, as follows:

$$\begin{array}{rl} r \models \sim \phi & \text{iff } -r \not\models \phi \\ r \models \phi + \psi & \text{iff for all worlds } s \text{ and } t, \text{ if } r \sqsubseteq -(-s \circ t), \text{ then } s \models \phi \text{ or } t \models \psi. \end{array}$$

Connectives such as these can be interpreted in terms of the 'stack–heap' models of Separation Logic that we discuss in Section 5.

This analysis is further expanded to a comprehensive treatment of multiplicative variants of all of the standard propositional connectives, in both the intuitionistic and classical settings, to be found in the work of Docherty and Pym [41; 42].

Having set up the truth-functional semantics, it is natural to introduce a tableaux proof system — for, example, Smullyan [9; 104]. Tableaux for classical propositional logic can be set up as a very simple proof system, which can be seen in two intimately related ways. First, as representation of proof-search in the sequent calculus. Second, as a syntactic representation of truth-functional semantics.

A tableau checks whether or not a given set of formulae is satisfiable (for a given satisfaction relation). Checking an entailment $\phi_1, \ldots, \phi_m \models \psi$ amounts to checking the satisfiability of $\{\, \phi_1, \ldots, \phi_m, \neg\psi \,\}$. The construction of a tableau consists in decomposing the formulae, by the application of tableaux rules in order to identify the presence of complementary pairs of literals (i.e., of atoms and negated atoms).

The rules come in pairs for each connective, just as in the sequent calculus, one for a formula having the connective outermost and one for its negation. For example, one for $\phi \wedge \psi$ and one for $\neg(\phi \wedge \psi)$. Often, such pairs are written as $\mathbb{T}(\phi \wedge \psi)$ and $\mathbb{F}(\phi \wedge \psi)$, a representation that is convenient in non-classical logics.

Two forms of representation are commonplace. First, using a pictorial representation of trees and, second, using the the syntactic form of inference rules. For example, the two rules for $\wedge$ can be written as follows:

$$\begin{array}{c} \mathbb{T}(\phi \wedge \psi) \\ | \\ \mathbb{T}(\phi) \\ \\ \mathbb{T}(\psi) \end{array} \qquad\qquad \begin{array}{ccc} & \mathbb{F}(\phi \wedge \psi) & \\ \swarrow & & \searrow \\ \mathbb{F}(\phi) & & \mathbb{F}(\psi) \end{array}$$

Notice that the $\mathbb{F}$ rule introduces two branches to the tableau. These branches represent a disjunction or nondeterminism in the tableaux method: in order to check $\neg(\phi \wedge \psi)$ for satisfiability, it is sufficient falsify just one of ϕ and ψ. The representation of the two rules for $\vee$ is exactly dual to that of those for $\wedge$.

The second form of representation is more thoroughly syntactic, but is perhaps a more convenient approach for more complex systems of tableaux, such as those we consider in this article. In this representation, the rules for $\wedge$ have the following form:

$$\frac{\mathbb{T}(\phi \wedge \psi)}{\{\, \mathbb{T}(\phi)\, ,\, \mathbb{T}(\psi) \,\}} \qquad \frac{\mathbb{F}(\phi \vee \psi)}{\mathbb{F}(\phi) \mid \mathbb{F}(\psi)}$$

where the $|$ symbol encodes the formation of the branches of the tableau. Again, the form of the two rules for $\vee$ is exactly dual to that of those for $\wedge$.

If a pair of complementary literals is found on a branch of a tableau, then the tableau is said to be *closed* and the root formula is falsified.

For BI, and other non-classical logics, we work not with the basic tableaux of classical logic, but with *labelled tableaux* [52] — see Goré [55; 58] for general discussions. The general approach is quite straightforward, and can be read off from the form of the truth-functional semantics. We can think of a classical model as having a single world, where as a model of BI has an ordered partial monoid of worlds, the structure of which is used essentially in the semantics of $*$. This stands in contrast to the classical semantics of $\wedge$, which is given within the one-world of the classical model.

The tableaux rules for $*$ reflect this difference quite directly. To determine the satisfaction of $\phi * \psi$ it is necessary to assume a world and then use the algebra of the partially ordered partial monoid to determine worlds at which the satisfaction of the subformulae, ϕ and ψ, are satisfied. This algebraic relationship between these three worlds is a constraint upon the construction of tableaux and must be represented within their rules.

The solution is, in summary, as follows:

- associate with formulae labels denoting the worlds at which satisfaction is being considered;
- associate with the tableaux rules the necessary algebraic relationship between the worlds to which the rules refer using an algebra of labels;
- extend the closure conditions on tableaux to account for the presence of the labels.

The tableaux rules for BI, illustrated in Figure 2, employ a set $\mathcal{F}$ of *labels* for formulae and a set $\mathcal{C}$ of constraints on labels. The labels can be composed and constraints are orderings of labels. The set $\mathcal{C}$ comes along with a set of rules for closing constraints and the closure of $\mathcal{C}$, denoted $\overline{\mathcal{C}}$, is the least relation closed under these rules. A *tableau* for the formula ϕ is a *tableau* for $\langle \{\mathbb{F}\phi : c_0\}, \{c_0 \preccurlyeq c_0\}\rangle$. See [52; 20; 40; 43; 44] for more details of the general — and quite generic — formulation of this set-up.

Notice how the rules for each connective follow the pattern of the corresponding case of the satisfaction relation. For example, the rule for $*$ requires that the labels for the components of the conjunction when combined are below the label for their conjunction in the constraint ordering. For another example, note that the labelling and constraints play no role in the rules for $\wedge$ and $\vee$. For the intuitionistic implication, the constraints capture the requirement from Kripke semantics of satisfiability at future worlds, which can also be seen as the Gödel-Mckinsey-Tarski translation of intutionistic logic into S4 modal logic.

Soundness and completeness results for this semantics relative to BI's proof theory, expressed either as a Hilbert systems as in [89; 95] or as described above in terms of labelled tableaux, can be obtained in a variety of settings [95; 52].

The first step is an elementary semantics based simply on partially ordered monoids (not partial monoids). This semantics is very appealing, but is complete (for BI's various proof systems) only for BI without $\perp$. Incompleteness derives from the interaction between falsity and multiplicative implication. To see this, consider that in the elementary semantics, we have the following form of consistency for a formula ϕ: for any

$$\frac{\mathbb{T}\phi \wedge \psi : x \in \mathcal{F}}{\langle\{\mathbb{T}\phi : x, \mathbb{T}\psi : x\}, \emptyset\rangle} \langle\mathbb{T}\wedge\rangle \qquad \frac{\mathbb{F}\phi \wedge \psi : x \in \mathcal{F}}{\langle\{\mathbb{F}\phi : x\}, \emptyset\rangle \mid \langle\{\mathbb{F}\psi : x\}, \emptyset\rangle} \langle\mathbb{F}\wedge\rangle$$

$$\frac{\mathbb{T}\phi \vee \psi : x \in \mathcal{F}}{\langle\{\mathbb{T}\phi : x\}, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : x\}, \emptyset\rangle} \langle\mathbb{T}\vee\rangle \qquad \frac{\mathbb{F}\phi \vee \psi : x \in \mathcal{F}}{\langle\{\mathbb{F}\phi : x, \mathbb{F}\psi : x\}, \emptyset\rangle} \langle\mathbb{F}\vee\rangle$$

$$\frac{\mathbb{T}\phi \rightarrow \psi : x \in \mathcal{F} \text{ and } x \preccurlyeq y \in \overline{\mathcal{C}}}{\langle\{\mathbb{F}\phi : y\}, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : y\}, \emptyset\rangle} \langle\mathbb{T}\rightarrow\rangle \qquad \frac{\mathbb{F}\phi \rightarrow \psi : x \in \mathcal{F}}{\langle\{\mathbb{T}\phi : c_i, \mathbb{F}\psi : c_i\}, \{x \preccurlyeq c_i\}\rangle} \langle\mathbb{F}\rightarrow\rangle$$

$$\frac{\mathbb{T}\phi * \psi : x \in \mathcal{F}}{\langle\{\mathbb{T}\phi : c_i, \mathbb{T}\psi : c_j\}, \{c_i c_j \preccurlyeq x\}\rangle} \langle\mathbb{T}*\rangle \qquad \frac{\mathbb{F}\phi * \psi : x \in \mathcal{F} \text{ and } yz \preccurlyeq x \in \overline{\mathcal{C}}}{\langle\{\mathbb{F}\phi : y\}, \emptyset\rangle \mid \langle\{\mathbb{F}\psi : z\}, \emptyset\rangle} \langle\mathbb{F}*\rangle$$

$$\frac{\mathbb{T}\phi \mathrel{-\!\!*} \psi : x \in \mathcal{F} \text{ and } x \preccurlyeq y, yz \preccurlyeq yz \in \overline{\mathcal{C}}}{\langle\{\mathbb{F}\phi : z\}, \emptyset\rangle \mid \langle\{\mathbb{T}\psi : yz\}, \emptyset\rangle} \langle\mathbb{T}\mathrel{-\!\!*}\rangle \qquad \frac{\mathbb{F}\phi \mathrel{-\!\!*} \psi : x \in \mathcal{F}}{\langle\{\mathbb{T}\phi : c_j, \mathbb{F}\psi : c_i c_j\}, \{x \preccurlyeq c_i, c_i c_j \preccurlyeq c_i c_j\}\rangle} \langle\mathbb{F}\mathrel{-\!\!*}\rangle$$

with c_i and c_j being fresh atomic labels

Fig. 2. Some tableaux rules for BI [52]

m,

$$m \models (\phi \mathrel{-\!\!*} \bot) \rightarrow \bot \quad \text{iff} \quad \text{there is an } n \text{ such that } n \models \phi.$$

This can be established by unpacking the implications using the satisfaction relation. It follows, then, that

$$((\phi \mathrel{-\!\!*} \bot) \rightarrow \bot) \wedge ((\psi \mathrel{-\!\!*} \bot) \rightarrow \bot) \models ((\phi * \psi \mathrel{-\!\!*} \bot) \rightarrow \bot),$$

but

$$((\phi \mathrel{-\!\!*} \bot) \rightarrow \bot) \wedge ((\psi \mathrel{-\!\!*} \bot) \rightarrow \bot) \vdash ((\phi * \psi \mathrel{-\!\!*} \bot) \rightarrow \bot)$$

is not provable. To see this, consider that if $k \models \phi$ and $l \models \psi$, then $k \circ l \models \phi * \psi$.

Alternatively, we can see that $\phi, \phi \mathrel{-\!\!*} \bot \vdash \bot$ is provable, but, in the necessary term model construction the bunch $\phi, \phi \mathrel{-\!\!*} \bot$ is equivalent to $\bot$, so that a world representing $\bot$ would be needed.

The second step is to formulate a version of the elementary semantics in 'Grothendieck topological monoids' [95]. In this semantics, the topological structure admits an 'empty' world, which satisifies $\bot$, and completeness for the full logic is recovered. The third step is to formulate BI's semantics in the tradition of relevant logics, using ternary relations R satisfying a number of conditions on worlds [4; 5; 98]. In this semantics, for example, $m \models \phi * \psi$ iff there exist n and p such that $R(n, p, m)$ and $n \models \phi$ and $p \models \psi$, and there is a distinguished element that satisfies $\bot$. Completeness holds for this semantics. Finally, the PDM semantics, as given above in Figure 1, can be seen to arise as a special case of the ternary relation semantics [52].

The partiality of PDM semantics is, in practice, very useful. It is used in BI's 'Pointer Logic' [65], which is a theory of Boolean BI, to which we return below, and which provides the semantic basis for Separation Logic [99]. The completeness of labelled tableaux for the partial monoid semantics of Boolean BI has been given by Larchey-Wendling [79].

This analysis is comprehensively generalized by Docherty and Pym [44].

From the perspective of resource semantics, we might argue that, in practical applications of resource semantics, partiality of resource composition is a natural requirement. Though conceptually convincing quite directly, this naturality will become very clear when we consider the semantic basis of Separation Logic, below.

4.4. Linear Logic, BI, and Resources in Economics and Physics

Linear Logic (LL) famously has an interpretation in terms of resources, due to Lafont [75]. This interpretation differs from our resource semantics in that it resides in LL's proof systems. The classic example is perhaps that of the vending machine.

We denote having a chocolate bar by the atomic proposition $Choc$ and having one dollar by $\$1$. one can state that one dollar will buy one chocolate bar by an implication $\$1 \rightarrow Choc$, using material implication. But then we have $\$1 \rightarrow Choc \wedge Choc$, which asserts that one dollar will buy two chocolates. Linear Logic's proofs analyse this situation more carefully using linear implication and tensor product to write $\$1 \multimap Choc$ and $(\$1 \otimes \$1) \multimap (Choc \otimes Choc)$. From $\$1$ and $\$1 \multimap Choc$ we can conclude $Choc$ and from $\$2$ and $(\$1 \otimes \$1) \multimap (Choc \otimes Choc)$ we can conclude $(Choc \otimes Choc)$; that is, 2 chocolates. Thus linear implications $A \multimap B$ are interpreted as transforming resource A into resource B. In this interpretation, $A \otimes B$ is interpreted as simultaneous occurrence of resources A and B, $A \& B$ is interpreted as external (consumer) choice between resources A and B, and $A \oplus B$ is interpreted as internal (producer) choice between resources. LL's modalities, ! and ?, are interpreted as expressing infinite (stream of) resource(s), and the units denote things such as the absence of resource and waste baskets for resources.

In contrast, BI's resource semantics would treat the vending machine using a model that is essentially the ordered monoid of natural numbers, $(\mathbb{N}, \leq, +, 0)$, and then make assertions such as $2 \models Choc * Choc$, read as '2 dollars is enough money to buy two chocolates'.

In this vein, BI's multiplicative implicational formulae, $\phi \mathrel{-\!\!*} \psi$, can be interpreted as functions with a resource cost, and the cost of obtaining the output from the function is the cost of the input combined with the cost of the function. For example, consider making dinner, using the cooking formula $Ingredients \mathrel{-\!\!*} Dinner$. Suppose the cost of the ingredients is $\$m$ and the cost of cooking them is $\$n$, then the cost of making dinner, is $\$(m+n)$:

$$n \models Ingredients \mathrel{-\!\!*} Dinner \text{ iff for any } m \text{ s.t. } m \models Ingredients,\ n+m \models Dinner.$$

BI includes MILL, and so includes that fragment of LL's resource interpretation. Infinite resources can be represented using BI's additives. BI does not directly represent the internal–external distinction, but the concepts can readily be replicated within resource semantics: for example, see the producer–consumer models in [97; 27].

LL's (relatively recently settled) Kripke semantics [2; 18] does not have BI's direct resource interpretation. Understanding what resource interpretations might be available there would seem to be an interesting challenge. In contrast, as we discuss herein, Brotherston and Villard [16] and, more comprehensively, Docherty and Pym [41; 42] have provided comprehensive treatments of multiplicative variants of all the propositional connectives, including disjunction.

We conclude this section by remarking on the concept of resource in other fields. The main area of interest for this is perhaps economics, where two ideas are pertinent: *rivalrous goods* and *excludable goods* (though we have also remarked upon connections to quantum information theory [24; 39], and see below). Here, the term 'goods' corresponds to our notion of resources.

A good is rivalrous if, when the good is used by one actor, then it has been consumed and cannot be used by another actor. For example, if a person eats some food, then no-one else can eat the same food. Rivalrous resources are thus consumed in the sense of Linear Logic, and cannot be shared, in the sense of BI's resource semantics. A good is excludable if one actor can prevent another from using the good. For example, if one passenger buys a ticket for a reserved seat on a train, then other passengers cannot use that seat. This latter property of resources belongs more properly in our treatment of the manipulation of located resources by processes.

The concept of resource also appears in physics, as discussed in Docherty's PhD thesis [39]. In particular, recent work by Coecke et al. [24] and Fritz [48] proposes an algebraic framework for 'resource theories' that is inspired by quantum information theory. The basic idea closely resembles BI's semantics: a set of resources carries the structure of an ordered commutative monoid in which composition represents combination of resources, but the order $\succcurlyeq$ is generated by a background (e.g., physical) theory of the *conversion* of resources; for example, the conversions of molecules under chemical reactions.

Two interpretations are of interest, using Docherty's [39] terminology of 'invariance' and 'sufficiency'.

- In the *invariant interpretation*, $s \preccurlyeq r$ is read as 's converts to r'. This idea derives from the interpretation of persistence of formula satisfaction: if ϕ is a property of s and s converts to r, then ϕ is a property of r ; that is, ϕ is an invariant property of conversion). For example, in chemical reactions, the invariant is the number of basic chemical elements (conservation of mass).
- In the *sufficiency interpretation*, $r \succcurlyeq s$ is read as 'r converts to s'. This idea derives from the interpretation of persistence of formula satisfaction: if s is sufficient for task ϕ and r converts to s, then r is sufficient for ϕ. In the vending machine notion of BI, conversion is spending money elsewhere, so the 'conversion' order and the greater-than order coincide. For example, consider the ordered monoid of natural numbers, $(\mathbb{N}, \leq, +0)$, that is a model of BI. If $\$m$ is sufficient for a given purchase, then any amount $\$(m+n)$ is also sufficient for that purchase, and $\$(m+n) \geq \$m$ by spending $\$n$ elsewhere.

These ideas are discussed much more thoroughly in Docherty's PhD thesis [39].

4.5. Propositions-as-types

The usual close relationship between presheaf models and Kripke semantics — which we might think of as 'propositions-as-types' — can be seen to extend to the multiplicatives via Day's construction — notice the immediate correspondence between the categorical interpretation of (proofs of) $*$-formulae and the truth-functional satisfaction condition for $*$:

$$\begin{array}{ccccccc} (E \otimes F)X & = & \int^{Y,Y'} & EY & \times & FY' & \times \; \mathcal{C}[X, Y \otimes Y'] \\ X \models E \otimes F & \text{iff} & \exists Y, Y' \text{ s.t.} & Y \models E & \text{and} & Y' \models F & \text{and} \; X \sqsubseteq Y \otimes Y' \end{array}$$

The correspondence for $-\!\!*$ is similar.

The propositions-as-types correspondence is, of course, properly expressed as a correspondence between, on the one hand, the terms and types of a λ-calculus and, on the other, the (usually natural deduction) proofs and propositions of a logic. For propositional BI with intuitionistic additives, the propositions-as-types correspondence holds for the $\alpha\lambda$-calculus.

4.6. The $\alpha\lambda$-calculus and its resource-semantics interpretation

The $\alpha\lambda$-calculus is the λ-calculus that stands in propositions-as-types correspondence with BI (with intuitionistic additves). The $\alpha\lambda$-calculus has been used to give accounts of interference and non-interference in the programming languages SCI and Idealized Algol [85].

The role of resource semantics in the $\alpha\lambda$-calculus emphasises the interpretation of multiplicative implication, in contrast to its additive counterpart, in terms of the separation and sharing of resources in the arguments of functions.

The basic idea — in the context of functional programming, — is that a function f may have types $A \multimap B$ or $A \to B$:

- $f : A \multimap B$ functions f that *do not share* resources with their arguments — for example, the cooking procedure mentioned above.
- $f : A \to B$ functions f that *may share* resources with their arguments; that is, it is possible that f shares resources with its argument, but it does not necessarily do so.

These interpretations are supported directly by the typing rules associated with such functions, which stand in the usual proposition-as-types relation to the underlying propositional logic, as described by Howard [64], extended to include the multiplicative functions, as described above, and types $A * B$ and I . For example,

$$\frac{\Gamma\,,\, x\!:\!A \vdash M : B}{\Gamma \vdash \lambda x\!:\!A.M : A \multimap B}\ \multimap\mathbf{I} \qquad \frac{\Gamma\,;\, x\!:\!A \vdash M : B}{\Gamma \vdash \alpha x\!:\!A.M : A \to B}\ \to\mathbf{I}$$

and

$$\frac{\Gamma \vdash M : A \multimap B \quad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\ \multimap\mathbf{E} \qquad \frac{\Gamma \vdash M : A \to B \quad \Delta \vdash N : A}{\Gamma ; \Delta \vdash M @ N : B}\ \to\mathbf{E}.$$

The basic properties of $\alpha\lambda$ are as one would expect:

- Equations:

$$(\alpha x.M)N = M[N/x] \quad (\alpha x.Mx) = M\ (x \notin \mathbf{FV}(M))$$

$$(\lambda x.M)N = M[N/x] \quad (\lambda x.M@x) = M\ (x \notin \mathbf{FV}(M))$$

the usual additive projection–pairing equations

$$\mathbf{let}(x_1, x_2) = M_1 * M_2\, \mathbf{in}\, N = N[M_1/x_1, M_2/x_2]$$

$$\mathbf{let}(x, y) = M\, \mathbf{in}\, x * y = M$$

- Properties:
  - Cut (substitution) is admissible;
  - The usual additive forms of the typing rules are admissible;
  - β-reduction (equations read left to right) preserves typing.

The use of $\alpha\lambda$ and its models as a quite general framework for understanding syntactic control of interference is a substantial topic with a lot of technical content — we cannot possibly begin to do it justice here. See [85] for the full story.

5. THE RESOURCE SEMANTICS OF SEPARATION LOGIC

Separation Logic [86], introduced by Ishtiaq and O'Hearn [65], and Reynolds [99], is an extension of Hoare's program logic which addresses reasoning about programs that access and mutate data structures. Here we follow a discussion by Pym, Spring, and O'Hearn [96].

The usual presentation of Separation Logic is based on Hoare triples — for reasoning about the state of imperative programs — of the form $\{\,\phi\,\}\,C\,\{\,\psi\,\}$, where C is a program command, ϕ is pre-condition for C, and ψ is a post-condition for C. Reynolds' programming language is a simple language of commands with a Lisp-like set-up for creating and accessing cons cells: $C \ ::=\ x := E \mid x := E.i \mid E.i := E' \mid x := \mathit{cons}(E_1, E_2) \mid$

$\ldots$. Here the expressions E of the language are built up using booleans, variables, etc., $cons$ cells, and atomic expressions. Separation Logic thus facilitates verification procedures for programs that alter the heap.

A key feature of Separation Logic is the local reasoning provided by the so-called Frame Rule,

$$\frac{\{\phi\}\, C\, \{\psi\}}{\{\phi * \chi\}\, C\, \{\psi * \chi\}}\ \text{Modifies}(C) \cap \text{Free}(\chi) = \emptyset,$$

where the side-condition ensures that χ does not include any free variables modified by the program C; that is, that the state that is manipulated by C is separate from that which is described by the property χ.

The value of the Frame Rule lies in its support for *local reasoning* about state. This is best understood by reading the rule from conclusion to premisses, as in proof-search. Then, in seeking to establish the provability or truth of the conclusion, identification of a part of the model — here computer memory — that is characterized by χ allows that part of the model to be discarded, so enabling significant simplification of the computation.

Static analysis procedures based on the Frame Rule form the basis of Facebook's Infer tool (fbinfer.com) that is deployed in its code production. The decomposition of the analysis that is facilitated by the Frame Rule is critical to the practical deployability of Infer.

The resource semantics described above, somewhat richer than that which is available in Linear Logic [54], allows the construction of specific logical models for a characterization of computer memory. Characterizing memory addressed challenging problems in program verification. Over the following 15 years, Separation Logic has developed into a reasoning tool successfully deployed at large technology firms like Facebook and Spotify. In this section, we explain how the semantics of (Boolean) BI as described above forms the basis of separation logic.

Ishtiaq and O'Hearn [65] introduced 'BI Pointer Logic', based on a specific example of Boolean BI's resource semantics. Three points about BI Pointer Logic are key.

- First, its resource semantics is constructed using the stack, used for static, compile-time memory allocation, and the heap, used for dynamic, run-time memory allocation:
- Second, the semantics of the separating conjunction, $*$, splits the heap, but not the stack: the stack contains the allocations required to define the program, which are unchanged at run-time; the heap contains the allocations made during computation.
- Third, it employs a special class of atomic propositions constructed using the 'points to' relation, $\mapsto$: $E \mapsto E_1, E_2$ means that expression E points to a cons cell E_1 and E_2. (It also employs a class of atomic propositions which assert the equality of program expressions, but this is a standard formulation.)

These factors combine to give an expressive and convenient tool for making statements about the contexts of heap (cons) cells. For example, the separating conjunction

$$(x \mapsto 2, y) * (y \mapsto 3, x)$$

says that x and y denote distinct locations. Further, x is a structured variable with two data types; the first, an integer, is 2, and the second is a pointer to y. The variable y denotes a location with a similar two-part structure in which the first part, also called the car, contains 3 and the second part, sometimes called the cdr ('could-er'), contains a pointer back to x [65]. Note that the pointers identify the whole two-part variable, not just the car. Figure 3 displays this linked list relationship in pictures.

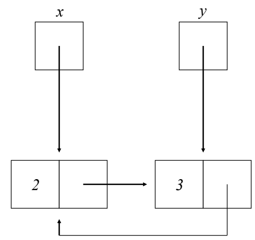

Fig. 3. Variable names are placed above their boxes, and contents of the variable are inside the box. Overall, the diagram represents the logical assertion $(x \mapsto 2, y) * (y \mapsto 3, x)$.

Separation Logic can usefully and safely be seen (see [65; 110] for the details) as a presentation of BI Pointer Logic — indeed, this is the semantics of Separation Logic, as described by Reynolds [99].

BI Pointer Logic, a *theory* of (first-order) Boolean BI (BBI), is an instance of BBI's resource semantics in which the monoid of resources is constructed from the program's heap; that is, we employ a model in which 'resource' corresponds to 'portion of computer memory'. (Note: a version of Separation Logic based on BI, not Boolean BI, is also possible [65].)

In detail, this model has two components, the store and the heap. The store is a partial function mapping from variables to values, $a \in$ Val, such as integers, and the heap is a partial function from natural numbers to values. In logic, the store is often called the valuation, and the heap is a possible world. In programming languages, the store is sometimes called the environment. Within this set-up, the atomic formulae of BI Pointer Logic include equality between expressions, $E = E'$, and, crucially, the points-to relation, $E \mapsto F$. To set all this up, we need some additional notation: $dom(h)$ denotes the domain of definition of a heap h and $dom(s)$ is the domain of a store s; $h\#h'$ denotes that $dom(h) \cap dom(h') = \emptyset$; $h \cdot h'$ denotes the union of functions with disjoint domains, which is undefined if the domains overlap; $[f \mid v \mapsto a]$ is the partial function that is equal to f except that v maps to a; expressions E are built up from variables and constants, and so determine denotations $[\![E]\!]s \in \mathrm{Val}$. With this basic data, the satisfaction relation for BI Pointer Logic is defined as in Figure 4.

The judgement, $s, h \models \phi$, says that the assertion ϕ holds for a given store and heap, assuming that the free variables of ϕ are contained in the domain of s. Note the case for $*$,

$$s, h \models \phi * \psi \quad \text{iff} \quad \begin{array}{l}\text{there are } h_0,\, h_1 \text{ s.t. } h_0 \# h_1,\, h_0 \cdot h_1 = h, \\ s, h_0 \models \phi \text{ and } s, h_1 \models \psi,\end{array}$$

includes the condition $h_0 \# h_1$, that the heaps h_0 and h_1 are disjoint — we consider only those compositions of heaps that satisfy this constraint and consider $\circ$ to be undefined if this constraint is not satisfied.

The remaining classical connectives are defined in the usual way: $\neg\phi = \phi \to \bot$; $\top = \neg\bot$; $\phi \vee \psi = (\neg\phi) \to \psi$; $\phi \wedge \psi = \neg(\neg\phi \vee \neg\psi)$; and $\forall x \,.\, \phi = \neg\exists x \,.\, \neg\phi$.

The multiplicative negation and disjunction, discussed in Section 4.3, can be interpreted in terms of intersections of heaps [16] in the resource semantics of Separation Logic. This provides, in the presence of appropriate structure on worlds, an example of the richness of resource semantics; for example, intersections support a rich language for expressing properties of memory.

A systematic theory of labelled tableaux proof systems for separation theories, supporting systematic soundness and completeness theorems has been presented by

$s, h \models E = E'$	iff	$[\![E]\!]s = [\![E']\!]s$
$s, h \models E \mapsto (E_1, E_2)$	iff	$[\![E]\!]s = \mathrm{dom}(h)$ and $h([\![E]\!]s) = \langle [\![E_1]\!]s, [\![E_2]\!]s \rangle$
$s, h \models \mathtt{emp}$	iff	$h = [\,]$ (the empty heap)
$s, h \models \phi * \psi$	iff	there are h_0, h_1 s.t. $h_0 \# h_1$, $h_0 \cdot h_1 = h$, $s, h_0 \models \phi$ and $s, h_1 \models \psi$
$s, h \models \phi \mathrel{-\!\!*} \psi$	iff	for all h', if $h' \# h$ and $s, h' \models \phi$, then $s, h \cdot h' \models \psi$
$s, h \models \bot$	never	
$s, h \models \phi \to \psi$	iff	$s, h \models \phi$ implies $s, h \models \psi$
$s, h \models \exists x \,.\, \phi$	iff	for some $v \in \mathrm{Val}$, $[s \mid x \mapsto v], h \models \phi$

Fig. 4. The satisfaction relation for BI Pointer Logic [65].

Docherty and Pym [44]. The tableaux systems employed follow the pattern of those we have described for above BI and of those that we describe below for a modal variation of BI and for layered graph logics.

As we have seen, BI Pointer Logic, with its truth-functional semantics — a specific example of (B)BI's resource semantics —- of the form

$$s, h \models \phi$$

provides an elegant semantics for reasoning about the correctness of programs that manipulate computer memory. However, as we have seen, for reasoning directly about the behaviour of programs, Hoare logic, based on triples $\{\phi\}\, C\, \{\psi\}$, is both natural and convenient.

The main reason why Hoare triples are so convenient is that they include directly code, C, whereas BI Pointer Logic is formulated wholly in terms of properties of the contents of memory. We connect these two points of view by providing a semantics of Hoare triples in terms of BI Pointer Logic [65; 110]. There are essentially two ways of going about this, depending on the strength of requirements on the behaviour of the code. The behaviour of code is expressed in terms of the evaluation of a program C — using stack s and heap h — with respect to sequences of steps defined by its operational semantics, $\rightsquigarrow$, and essentially denoted by $C, s, h \rightsquigarrow^* s', h'$, read as 'the program C transforms the memory configuration s, h into the memory configuration s', h''. There is a special configuration, *fault*, indicating a memory fault or abnormality.

The first semantics for Hoare triples, called partial correctness, relies on the notion of safety,

$$C, s, h \text{ is safe if } C, s, h \not\rightsquigarrow^* \textit{fault}$$

and is the 'fault-avoiding' interpretation, as explained in [110]:

Partial correctness semantics: $\{\phi\}\, C\, \{\psi\}$ is true in a model of Pointer Logic if, for all s, h, $s, h \models \phi$ implies
- C, s, h is safe, and
- if $C, s, h \rightsquigarrow^* s', h'$, then $s', h' \models \psi$.

The second, called total correctness [110], does not require the safety condition because it requires the 'stronger' property of 'normal' termination; that is, the program returns a value that lies within its intended range of outputs:

Total correctness semantics: $\{\phi\}\, C\, \{\psi\}$ is true in a model of Pointer Logic if, for all s, h, $s, h \models \phi$ implies

- C, s, h must terminate normally, and
- if $C, s, h \rightsquigarrow^* s', h'$, then $s', h' \models \psi$.

With these definitions, and some non-trivial technical development, soundness (that the rule transforms true properties into true properties) and completeness (that the rule derives one specification statement from another just when this inference holds semantically) theorems for the Frame Rule,

$$\frac{\{\phi\}\, C\, \{\psi\}}{\{\phi * \chi\}\, C\, \{\psi * \chi\}}\ \text{Modifies}(C) \cap \text{Free}(\chi) = \emptyset,$$

can be established [110]. These theorems give precise mathematical expression to the coincidence of the logical and engineering models of computer memory allocation.

In this section we have provided some detail on the novel aspects of Separation Logic's semantics, and how they support reasoning about computer memory as a resource. At heart, the atoms of the logic are composable in a way that mirrors the way that the physical substrate is composable. The physical transistors come apart, and one can make meaningful claims about affixing or pulling apart bits of silicon that have reliable impacts on the changes to the electrical and computational properties of the physical system. The structure of the logical model using partial commutative monoids and $*$ that we have introduced allows for logical claims to naturally mirror this physical fact.

The following section details the cluster of properties surrounding the proof theory of Separation Logic that make it a successful engineering tool. Part of these also relate to the composability of $*$ through the Frame Rule, as it is leveraged for efficient computation of results. Equally important to the deployability of the proof theory is the automation of bi-abduction for generating hypothetical pre- and post-conditions to drive proof solutions. The abductive rules we use are essentially encodings of engineer's heuristics when reasoning about computer memory usage, further demonstrating the deep ways in which the logical and engineering aspects of the task merge in Separation Logic.

Docherty and Pym [41; 42] have developed Stone-type duality theorems for Separation Logic — which requires the idea of 'BI hyperdoctrines'; q.v. [12] — and have placed these results in the broader context of the family of bunched logics [43]. These results establish a systematic analysis of the relationship between (Kripke-style) resource semantics and algebraic characterizations of bunched logics.

Moving beyond basic Separation Logic, there has been a great deal of work in the last nearly two decades. Much of that work is based, directly or indirectly, in the stack-heap semantics of Pointer Logic as presented in [65]. More recently, however, we have seen work in Separation Logic that exploits more seriously the strength of resource semantics. Leading examples of this direction can be see in the work Birkedal and Dreyer and their colleagues; see, for example, [69; 70]. These ideas are discussed in the setting of resource semantics in [88; 13].

Finally, we note that all existing algebraic approaches to Separation Logic are included within Docherty and Pym's duality framework for relating algebraic models and relational models of (B)BI, and this allows us to prove them sound with respect to the standard store-heap semantics [41; 42].

6. MODAL AND EPISTEMIC BUNCHED LOGICS

Modal logics, including epistemic logics, extend propositional and predicate logics with concepts such as 'necessitation' and 'possibility', and allow the logical study of concepts such as knowledge, belief, obligation, and time. These concepts essentially derive their meaning from their interpretation in 'possible worlds', often as formulated in terms of Kripke models.

The techniques for defining the semantics and proof theory of modal logics that are based on classical and intuitionistic systems translate well to defining modal logics that are based on substructural systems.

In this section, we consider a modal logic, LSM, based on BI and its resource semantics, that is a conservative extension of S4 [20]. We explain the basic set-up of the logic (truth-functional semantics, a tableaux system) and consider some systems examples — taken from [20] and which build on [97; 28; 29; 27] — that illustrate resource semantics in this setting: mutual exclusion and producer–consumer; in [20], timed Petri nets are also considered. We conclude by mentioning briefly an epistemic modal logic that is based on BI and its resource semantics.

6.1. LSM

We can set up a conservative extension (a 'Logic of Separating Modalities' or LSM [20]) of the modal logic S4 which adds *multiplicative modalities*; that is, modalities that are parametrized on local resources. These modalities are defined relative to two-dimensional worlds, one of which captures the S4 accessibility relation and one of which supports the resource parametrization.

Roughly speaking, an LSM model is a 4-tuple $(W, \mathcal{M}, \mathbf{R}, \mathcal{V})$, where W is a set of worlds, $\mathcal{M}$ is a partial monoid of 'resources', $(Res, \bullet, e)$, $\bullet \subseteq (W \times Res) \times (W \times Res)$ is a relexive and transitive relation, and V is an interpretation of propositional letters in $\wp(W \times Res)$. Then we have

$$\begin{array}{lll} w, r \models_{\mathcal{M}} \Diamond_s \phi & \text{iff} & \text{there exist } w' \in W \text{ and } r' \in Res \text{ such that } r \bullet s \downarrow, \\ & & (w, r \bullet s)\mathbf{R}(w', r') \text{ and } w', r' \models_{\mathcal{M}} \phi \end{array}$$

$$\begin{array}{lll} w, r \models_{\mathcal{M}} \Box_s \phi & \text{iff} & \text{for all } w' \in W \text{ and all } r' \in Res, \text{ if } r \bullet s \downarrow \text{ and} \\ & & (w, r \bullet s)\mathbf{R}(w', r'), \text{ then } w', r' \models_{\mathcal{M}} \phi. \end{array}$$

Here, s is the local resource, associated with the modality, and r, in the model, is the ambient resource. The modalities are read as asserting that ϕ is possibly (respectively, necessarily) true at the world (w, r) subject to the availability of additional resource s.

Note that two other pairs of modalities are derivable from these:

- The basic additive modalities:

$$\begin{array}{lll} w, r \models_{\mathcal{M}} \Diamond \phi & \text{iff} & \text{there exist } w' \in W \text{ and } r' \in Res \text{ such that } (w, r)\mathbf{R}(w', r') \\ & & \text{and } w', r' \models_{\mathcal{M}} \phi \\ w, r \models_{\mathcal{M}} \Box \phi & \text{iff} & \text{for all } w' \in W \text{ and all } r' \in Res, \text{ if } (w, r)\mathbf{R}(w', r') \text{ then} \\ & & w', r' \models_{\mathcal{M}} \phi. \end{array}$$

- Multiplicative modalities with undetermined additional resource parameters:

$$\begin{array}{lll} w, r \models_{\mathcal{M}} \Diamond_\bullet \phi & \text{iff} & \text{there exist } w' \in W \text{ and } s, r' \in Res \text{ such that } r \bullet s \downarrow, \\ & & (w, r \bullet s)\mathbf{R}(w', r'), \text{ and } w', r' \models_{\mathcal{M}} \phi \\ w, r \models_{\mathcal{M}} \Box_\bullet \phi & \text{iff} & \text{for all } w' \in W \text{ and all } s, r' \in Res, \text{ if } (r \bullet s \downarrow \text{ and} \\ & & (w, r \bullet s)\mathbf{R}(w', r')) \text{ then } w', r' \models_{\mathcal{M}} \phi. \end{array}$$

Full details, including conservativity of LSM over S4, may be found in [20].

6.2. Examples of LSM's resource semantics

Mutual exclusion. This example is quoted more-or-less directly from [20]. We consider two processes (E_1 and E_2) that are in mutual exclusion. The automaton that describes the behaviour of the processes is given in Figure 5.

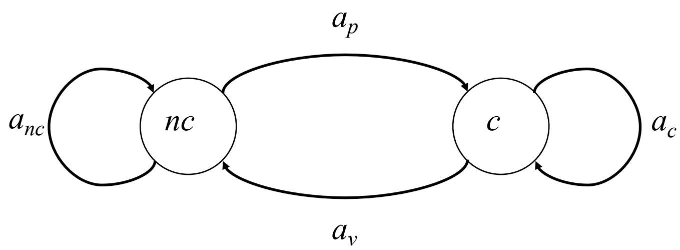

Fig. 5. Example of processes in mutual exclusion.

The processes have two states: nc, meaning that the process is in the non-critical section; and c, meaning that it is in the critical section. We denote by $\mathcal{S} = \{nc, c\}$ the state set of the processes.

In order to enter into the critical section, a process must hold a token, denoted J, and it releases the token when it leaves the critical section. The processes can perform four actions: a_{nc} a non-critical action, a_c a critical action, a_p the action that consists in taking a token and a_v the action that consists in releasing a token. We denote by $\mathcal{A} = \{a_{nc}, a_c, a_p, a_v\}$ the action set that can be performed by the processes.

We represent the resources (the token J) with $\mathcal{M} = (\{J^n \mid n \in \mathbb{N}\}, +, J^0)$, where $J^m + J^n = J^{m+n}$. In other words, J^n represents n tokens that are available for the system (the processes E_1 and E_2). We remark that $\mathcal{M}$ is obviously a partial monoid of resources. Now, we need a function that captures resource consumption and production when an action is performed. Following the approach taken in [97; 28; 27], we define a partial function $\mu : \mathcal{A} \times \{J^n \mid n \in \mathbb{N}\} \rightharpoonup \{J^n \mid n \in \mathbb{N}\}$ such that

$$\mu(a, J^n) = \begin{cases} J^n & \text{if } a \in \{a_{nc}, a_c\} \\ J^{n+1} & \text{if } a = a_v \\ J^{n-1} & \text{if } a = a_p \text{ and } n \geqslant 1 \\ \uparrow & \text{if } a = a_p \text{ and } n = 0 \end{cases}$$

where $\uparrow$ means 'undefined' and $\downarrow$ means 'defined'. We remark that performing a critical or a non-critical action (a_c and a_{nc}) consumes and produces no token, releasing a token (a_v) produces a token (J^{n+1}) and taking a token (a_p) consumes a token (J^{n-1}). Of course, $\mu(a_p, J^n)$ is defined if and only if there is at least one available token ($n \geqslant 1$). We introduce a relation that captures the transitions of a process and their effects on the available resources: $s, J^n \xrightarrow{a} s', J^m$ iff $s \xrightarrow{a} s'$ is a transition of Figure 5, $\mu(a, J^n) \downarrow$ and $\mu(a, J^n) = J^m$. For instance, we have $nc, J^1 \xrightarrow{a_p} c, J^0$, but $nc, J^1 \xrightarrow{a_v} c, J^0$ does not hold (because there is no transition $nc \xrightarrow{a_v} c$ in the automaton of Figure 5). This relation is very close in spirit to the judgements introduced in the SCRP calculus [97; 28; 27], which are of the form $R, E \xrightarrow{a} R', E'$, meaning that a process E performs an action a on a resource R and then provides the resource R' and the process E': this idea is discussed further in Section 8.

In order to deal with concurrent transitions, we need to define a set of concurrent states $W = \{s_1 \# s_2 \mid s_1, s_2 \in \mathcal{S}\}$ (where s_i is the state of the process E_i), a set of concurrent actions $\mathcal{A}^{\#} = \{a_1 \# a_2 \mid a_1, a_2 \in \mathcal{A}\}$ (where a_i is the action performed by the process E_i) and the following relation: $s_1 \# s_2, J^{n_1} + J^{n_2} \xRightarrow{a_1 \# a_2} s'_1 \# s'_2, J^{m_1} + J^{m_2}$ iff $s_1, J^{n_1} \xrightarrow{a_1} s'_1, J^{m_1}$ and $s_2, J^{n_2} \xrightarrow{a_2} s'_2, J^{m_2}$.

For example, the concurrent state $nc\#c$ is a state that captures E_1 in state nc and E_2 in state c. Moreover, the concurrent action $a_c\#a_p$ represents E_1 performing the action a_c and E_2 performing the action a_p. Concerning the relation $\Longrightarrow$, as $nc, J^1 \stackrel{a_p}{\to} c, J^0$ and $nc, J^0 \stackrel{a_{nc}}{\to} nc, J^0$ hold, then we have $nc\#nc, J^1 + J^0 \stackrel{a_p\#a_{nc}}{\Longrightarrow} c\#nc, J^0 + J^0$. Thus $nc\#nc, J^1 \stackrel{a_p\#a_{nc}}{\Longrightarrow} c\#nc, J^0$.

We are able to model the behaviour of the processes E_1 and E_2 and the token manipulation using the following LSM model $\mathcal{M} = (W, \mathcal{M}, \mathbf{R}, \mathcal{V})$, where

- $W = \{s_1\#s_2 \mid s_1, s_2 \in \mathcal{S}\}$,
- $\mathcal{M} = (\{J^n \mid n \in \mathbb{N}\}, +, J^0)$,
- $\mathbf{R}$ is the reflexive and transitive closure of $\Longrightarrow$, and
- $\mathcal{V}$ is defined by

p	$(w, r) \in \mathcal{V}(\mathrm{p})$ iff
J	$r = J^1$
nc_1	$w = nc\#nc$ or $w = nc\#c$
nc_2	$w = nc\#nc$ or $w = c\#nc$
c_1	$w = c\#nc$ or $w = c\#c$
c_2	$w = nc\#c$ or $w = c\#c$

We illustrate $\mathbf{R}$. As $c\#nc, J^0 \stackrel{a_v\#a_{nc}}{\Longrightarrow} nc\#nc, J^1$ and $nc\#nc, J^1 \stackrel{a_{nc}\#a_p}{\Longrightarrow} nc\#c, J^0$ hold, then $(c\#nc, J^0)\mathbf{R}(nc\#nc, J^1)$ and $(nc\#nc, J^1)\mathbf{R}(nc\#c, J^0)$. By transitive closure, we have $(c\#nc, J^0)\mathbf{R}(nc\#c, J^0)$. Concerning the valuation $\mathcal{V}$, J is the proposition meaning that there is one and only one available token, c_i is the proposition meaning that the process E_i is in critical section and nc_i is the proposition meaning that the process E_i is not in critical section.

We consider that the initial state of the system is $nc\#nc$ (each process is in non-critical section) and there is only one available token (J). We can obviously express that, in this initial state, each process is in non-critical section and there is only one available token as follows: $nc\#nc, J \models_{\mathcal{M}} nc_1 \wedge nc_2 \wedge J$.

The first important point is that LSM is a modal logic and it is possible to express properties on reachable states and available tokens. For example, we can express that it is impossible that the processes will be together in the critical section: $nc\#nc, J \models_{\mathcal{M}} \neg\Diamond(c_1 \wedge c_2)$ and also that it is always possible that each process can enter the critical section: $nc\#nc, J \models_{\mathcal{M}} \Box\Diamond c_1 \wedge \Box\Diamond c_2$.

The second important point is that LSM is a modal logic extended with the resource composition (denoted $\bullet$) that allows us to express properties of resources on the tokens that are produced and consumed. In particular, we can express that, in any reachable state, it is impossible that there can be more than one available token: $nc\#nc, J \models_{\mathcal{M}} \Box\neg(J * J * \top)$. It is also possible to express that if one process is in a non-critical section, then there is no available token $nc\#nc, J \models_{\mathcal{M}} \Box((c_1 \vee c_2) \to I)$. Indeed, only the unit resource satisfies I and, in our example, this unit resource is J^0 which encodes no available token.

Notice that the formula $\neg\Diamond(c_1 \wedge c_2)$, with the S4-like modality, fails to capture a vulnerability in the system. This security breach is highlighted by the new modalities: $nc\#nc, J \not\models_{\mathcal{M}} \neg\Diamond_{\bullet}(c_1 \wedge c_2)$. Indeed, if we assume that an intruder introduces one token into our system, then both processes can enter the critical section, because of the presence of a second token: $nc\#nc, J^1 + J^1 \stackrel{a_p\#a_p}{\Longrightarrow} c\#c, J^0$.

It follows that we can identify a new solution for the mutual exclusion problem such that $nc\#nc, J \models_{\mathcal{M}} \neg\Diamond_\bullet(c_1 \wedge c_2)$; that is, such that the processes cannot both enter into the critical section, whatever number of tokens is added.

Producer–consumer. This example also is quoted more-or-less directly from [20]. We consider an example based on the producer–consumer problem, but with a different approach: one in which the set of worlds W encodes the actions that the processes are performing and does not encode the current state of the processes. In this example, we consider two processes: a producer P_p and a consumer P_c that manipulate resources represented with $\mathcal{M} = (\{R^n \mid n \in \mathbb{N}\}, +, R^0)$, just as in the previous example.

The producer can perform just two actions: p (it is producing a new resource) and np (it is not producing). The consumer can also perform only two actions, which are c (it is consuming a resource) and nc (it is not consuming). Thus $W = \{p\#c, np\#c, p\#nc, np\#nc\}$ is the set of all concurrent actions that can be performed by the processes.

For instance, $p\#nc$ means that P_p is producing (p) and P_c is not consuming (nc). Clearly, only the following transitions hold, for all $w \in W$:

1. $np\#nc, R^n \Longrightarrow w, R^n$;
2. $p\#c, R^n \Longrightarrow w, R^n$;
3. $np\#c, R^n \Longrightarrow w, R^{n-1}$ only if $n \geqslant 1$; and
4. $p\#nc, R^n \Longrightarrow w, R^{n+1}$.

We remark that $np\#c, R^n \Longrightarrow w, R^{n-1}$ holds only if $n \geqslant 1$. Indeed, if there is no resource (R^0) and if P_p does not produce a new resource (np), then P_c cannot consume a resource (c).

Concerning the relation $\mathbf{R}$, we consider the reflexive and transitive closure of $\Longrightarrow$. Like in the previous example, we are able to propose a model for this system, that is $\mathcal{M} = (W, \mathcal{M}, \mathbf{R}, \mathcal{V})$ such that $\mathcal{V}$ is defined by

p	$(w, r) \in \mathcal{V}(\mathrm{p})$ iff
R	$r = R^1$
np	$w = np\#nc$ or $w = np\#c$
p	$w = p\#nc$ or $w = p\#c$
nc	$w = np\#nc$ or $w = p\#nc$
c	$w = np\#c$ or $w = p\#c$

In this model, by definition of $\mathbf{R}$ and reflexivity, $(np\#c, R^0)\mathbf{R}(w, R^n)$ only if $w = np\#c$ and $n = 0$, and we can express that if there is no resource (R^0), if P_p is not producing a new resource, and if P_c is consuming a resource, then the system is blocked (it never changes its state). In LSM, we can express this property as follows, for any $w \in W$ and any $n \in \mathbb{N}$: $w, R^n \models_{\mathcal{M}} \Box((I \wedge np \wedge c) \rightarrow \Box(I \wedge np \wedge c))$. It means that, for all reachable states (pairs of world/resource) and starting from any state, if there is no resource (I) and if P_p is not producing a new resource (np) and if P_c is consuming a resource (c) then the system always remains in this state ($\Box(I \wedge np \wedge c)$). Now, using multiplicative modalities, we can express that it is possible to unblock the system by adding a resource, as follows: $w, R^n \models_{\mathcal{M}} \Box((I \wedge np \wedge c) \rightarrow \Diamond_\bullet\neg(I \wedge np \wedge c))$.

6.3. Tableaux for LSM

Figure 6 gives the labelled tableaux rules for the modalities of LSM, presented in detail in [20]. The S4-like additive modalities are included for comparison with the multiplicative ones. Notice that there are a few differences between these rules and the previous ones for basic BI. These differences, in the handling of the labelling algebra, derive from the nature of the modalities.

$$\frac{\mathbb{T}\Diamond_y\phi : ux \in \mathcal{F}}{\langle\{\mathbb{T}\phi : s_ic_i\}, \{(u, x \circ \|y\|) \rightsquigarrow (s_i, c_i)\}\rangle}\ \langle\mathbb{T}\Diamond_y\rangle \qquad \frac{\mathbb{F}\Diamond_y\phi : ux \in \mathcal{F} \text{ and } (u, x \circ \|y\|) \rightsquigarrow (v, z) \in \overline{\mathcal{C}}}{\langle\{\mathbb{F}\phi : vz\}, \emptyset\rangle}\ \langle\mathbb{F}\Diamond_y\rangle$$

$$\frac{\mathbb{T}\Box_y\phi : ux \in \mathcal{F} \text{ and } (u, x \circ \|y\|) \rightsquigarrow (v, z) \in \overline{\mathcal{C}}}{\langle\{\mathbb{T}\phi : vz\}, \emptyset\rangle}\ \langle\mathbb{T}\Box_y\rangle \qquad \frac{\mathbb{F}\Box_y\phi : ux \in \mathcal{F}}{\langle\{\mathbb{F}\phi : s_ic_i\}, \{(u, x \circ \|y\|) \rightsquigarrow (s_i, c_i)\}\rangle}\ \langle\mathbb{F}\Box_y\rangle$$

$$\frac{\mathbb{T}\Diamond\phi : ux \in \mathcal{F}}{\langle\{\mathbb{T}\phi : s_ic_i\}, \{(u, x) \rightsquigarrow (s_i, c_i)\}\rangle}\ \langle\mathbb{T}\Diamond\rangle \qquad \frac{\mathbb{F}\Diamond\phi : ux \in \mathcal{F} \text{ and } (u, x) \rightsquigarrow (v, y) \in \overline{\mathcal{C}}}{\langle\{\mathbb{F}\phi : vy\}, \emptyset\rangle}\ \langle\mathbb{F}\Diamond\rangle$$

$$\frac{\mathbb{T}\Box\phi : ux \in \mathcal{F} \text{ and } (u, x) \rightsquigarrow (v, y) \in \overline{\mathcal{C}}}{\langle\{\mathbb{T}\phi : vy\}, \emptyset\rangle}\ \langle\mathbb{T}\Box\rangle \qquad \frac{\mathbb{F}\Box\phi : ux \in \mathcal{F}}{\langle\{\mathbb{F}\phi : s_ic_i\}, \{(u, x) \rightsquigarrow (s_i, c_i)\}\rangle}\ \langle\mathbb{F}\Box\rangle$$

with s_i, c_i and c_j being new label constants and $\|r\| = 1$ if $r = e$, otherwise r.

Fig. 6. Some tableaux modal rules for LSM.

Again, full details are to be found in [20], but, briefly:

- The starting point is the tableaux system for BI given in Figure 2, which employs an algebra of labels corresponding to the algebra employed in BI's satisfaction relation;
- LSM's satisfaction relation employs an algebra of labels for the basic connectives (not the modalities) that is essentially the same as that employed by BI (since LSM has classical additives, an ordering is not required);
- However, LSM employs a relational structure on worlds and resources for its semantics of the modalities;
- This reintroduces a need for a conversion operation, denoted by $\circ$ in the rules of Figure 6, to capture transitions between related world–resource pairs;
- Finally, note that since the language of the multiplicative modalities refers explicitly to resources — that is, semantic entities — we need a conversion function $\| - \|$ between resources and labels.

The main point to take away from these rules is that they follow the pattern of the basic set-up for BI, with variations that track the different semantics. This generic picture is explored more fully in [44].

Soundness and completeness results for LSM are provided in [20].

6.4. Epistemic modalities in resource semantics

Other modal extensions of BI have been proposed and explored in [50]. Epistemic modal logics employ modalities of the form K_a that are parametrized on an agent, a. The agent comes equipped with an equivalence relation $\sim_a$ on worlds, so that $w \models K_a\phi$ iff $v \models \phi$, where $v \sim_a w$. In separating epistemic logics, in which worlds are read as resources, we are able to parametrize the equivalence relation on additional local resource, as follows:

- $\mathbf{L}_a^s\phi$: required resource for outcome is equivalent to ambient resource + agent's resource: expresses that the agent, a, can establish the truth of ϕ using a given resource whenever the ambient resource, r, can be combined with the agent's local resource, s, to yield a resource that a judges to be equivalent to that given resource:

$$r \models \mathbf{L}_a^s\phi \text{ iff for all } r' \text{ such that } r' \sim_a r \bullet s,\ r' \models \phi;$$

- $\mathbf{M}_a^s\phi$: required resource for initiation is equivalent to ambient resource + agent's resource: expresses that the agent, a, can establish the truth of ϕ using a resource that is the combination of its local resource, s, with any resource such that a judges the combined resource to be equivalent to the ambient resource, r:

$$r \models \mathrm{M}_a^s \phi \text{ iff for all } r' \text{ such that } r' \bullet s \sim_a r,\, r' \bullet s \models \phi.$$

The epistemic system can be used to express access control policies and their violations [50]. Other applications of logics such as these remain to be explored.

7. WEAK BUNCHED LOGICS

So far our discussion has focussed on the logic BI, its modal extensions, their resource semantics, and their application to program verification through Separation Logic.

In this section, beginning again with BI's logical motivation, we broaden our discussion to a much broader family of logics, characterized by weakening the properties of BI's multiplicatives to be neither commutative nor associative. From a logical point of view, this yields a general framework for understanding the theory of the family of bunched logics and provides a setting in which a theory of *layered graphs* provide models.

This concept of layering also contributes to resource semantics. We have mentioned the idea of *location* as a key concept in distributed systems modelling. Typically, in modelling contexts, locations are captured using *directed graphs* or similar topological structures. Sometimes, even simpler structures will suffice: the stack-heap model in Separation Logic may be seen as consisting of locations (memory cells) with which are associated resources (values). In complex systems modelling, however, such as is very commonly encountered in physics and economics [74], the idea of layering is very widespread. Usually, it amounts to a layering of graphs of some kind.

Here we follow the development of Docherty and Pym [40], which follows on from [25; 26]. We first give a graph-theoretic account of the notion of layering that captures the concept as used in complex systems. Informally, two layers in a directed graph are connected by a specified set of edges, each element of which starts in the upper layer and ends in the lower layer. Our definition of layering contrasts with prior accounts in which the layering structure is left implicit [34; 90] and generalizes others which consider only a restricted class of layered graphs [91].

We begin by fixing notation and terminology. Given a directed graph, $\mathcal{G}$, we refer to its *vertex set* by $V(\mathcal{G})$. Its *edge set* is given by a subset $E(\mathcal{G}) \subseteq V(\mathcal{G}) \times V(\mathcal{G})$. H is a subgraph of $\mathcal{G}$ ($H \subseteq \mathcal{G}$) iff $V(H) \subseteq V(\mathcal{G})$ and $E(H) \subseteq E(\mathcal{G})$. The set of subgraphs of $\mathcal{G}$ is denoted $Sg(\mathcal{G})$.

To introduce layers, we identify a *distinguished set of edges* $\mathcal{E} \subseteq E(\mathcal{G})$. The *reachability relation* $\rightsquigarrow_{\mathcal{E}}$ on subgraphs of $\mathcal{G}$ is then defined $H \rightsquigarrow_{\mathcal{E}} K$ iff there exists $u \in V(H)$ and $v \in V(K)$ such that $(u, v) \in \mathcal{E}$. This generates a partial composition $@_{\mathcal{E}}$ on subgraphs of $\mathcal{G}$. Let $\downarrow$ denote definedness. For subgraphs H and K, $H\, @_{\mathcal{E}}\, K \downarrow$ iff $V(H) \cap V(K) = \emptyset, H \rightsquigarrow_{\mathcal{E}} K$ and $K \not\rightsquigarrow_{\mathcal{E}} H$, with output given by the graph union of the two subgraphs and the $\mathcal{E}$-edges between them. This composition is neither commutative nor — because grouping can determine definedness — associative.

Figure 7 shows subgraphs H and K for which $H\, @_{\mathcal{E}}\, K$ is defined, as well as the resulting composition. We say G is a *layered graph* (with respect to $\mathcal{E}$) if there exist H, K such that $H\, @_{\mathcal{E}}\, K \downarrow$ and $G = H\, @_{\mathcal{E}}\, K$. If this holds, we say H is *layered over* K and K is *layered under* H.

ILGL — intuitionistic layered graph logic, a variant of BI with intuitionistic additives in which the multiplicative conjunction is neither commutative nor associative — is interpreted on directed graphs that have been separated into ordered layers. Formally, an *ordered scaffold* is a structure $\mathcal{X} = (\mathcal{G}, \mathcal{E}, X, \preccurlyeq)$ such that

- $\mathcal{G}$ is a directed graph;

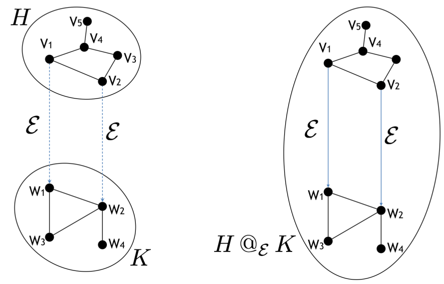

Fig. 7. The graph composition $H \mathbin{@_{\mathcal{E}}} K$

- $\mathcal{E}$ is a distinguished set of edges;
- X is a subset of $Sg(\mathcal{G})$ satisfying: if $G = H \mathbin{@_{\mathcal{E}}} K$ then $G \in X$ iff $H, K \in X$;
- $\preccurlyeq$ is an order on X that is reflexive and transitive.

We consider structures that are ordered so we can extend Kripke's ordered possible world semantics of intuitionistic propositional logic [73]. In Kripke's semantics, truth is *persistent* with respect to the order on possible worlds: if ϕ is true at a possible world x and $x \preccurlyeq y$, then ϕ is true at the world y. One can thus think of the intuitionistically valid propositions as those whose truth persists with the introduction of any new fact. In our setting, this means ILGL is suitable for reasoning about properties of graphs that are, for example, inherited from subgraphs, as well as modelling situations in which the components of the system carry a natural order.

A *layered graph model* $\mathcal{M} = (\mathcal{X}, \mathcal{V})$ is given by an ordered scaffold $\mathcal{X}$ and a valuation $\mathcal{V} : \mathrm{Prop} \to \mathcal{P}(X)$ satisfying $G \in \mathcal{V}(\mathrm{p})$ and $G \preccurlyeq H$ implies $H \in \mathcal{V}(\mathrm{p})$. For a layered graph model $\mathcal{M}$, the satisfaction relation $\models_{\mathcal{M}} \subseteq X \times \mathrm{Form}$ is inductively defined in Fig 8. ϕ is *valid for a layered graph model* $\mathcal{M}$ if, for all $G \in X$, $G \models_{\mathcal{M}} \phi$. ϕ is *valid* if it is valid for all layered graph models $\mathcal{M}$.

$$
\begin{array}{rll}
G \models_{\mathcal{M}} \top & & \text{always} \\
G \models_{\mathcal{M}} \bot & & \text{never} \\
G \models_{\mathcal{M}} \mathrm{p} & \text{iff} & G \in \mathcal{V}(\mathrm{p}) \\
G \models_{\mathcal{M}} \phi \wedge \psi & \text{iff} & G \models_{\mathcal{M}} \phi \text{ and } G \models_{\mathcal{M}} \psi \\
G \models_{\mathcal{M}} \phi \vee \psi & \text{iff} & G \models_{\mathcal{M}} \phi \text{ or } G \models_{\mathcal{M}} \psi \\
G \models_{\mathcal{M}} \phi \to \psi & \text{iff} & \text{for all } G \preccurlyeq H, H \models_{\mathcal{M}} \phi \text{ implies } H \models_{\mathcal{M}} \psi \\
G \models_{\mathcal{M}} \phi \blacktriangleright \psi & \text{iff} & \text{there exist } H, K \text{ s.t. } H \mathbin{@_{\mathcal{E}}} K \preccurlyeq G \\
 & & \text{and } H \models_{\mathcal{M}} \phi \text{ and } K \models_{\mathcal{M}} \psi \\
G \models_{\mathcal{M}} \phi \mathrel{-\!\blacktriangleright} \psi & \text{iff} & \text{for all } H, K, G \preccurlyeq H,\ H \mathbin{@_{\mathcal{E}}} K{\downarrow} \text{ and} \\
 & & H \models_{\mathcal{M}} \phi \text{ implies } H \mathbin{@_{\mathcal{E}}} K \models_{\mathcal{M}} \psi \\
G \models_{\mathcal{M}} \phi \mathrel{\blacktriangleright\!-} \psi & \text{iff} & \text{for all } H, K, G \preccurlyeq H,\ K \mathbin{@_{\mathcal{E}}} H{\downarrow} \text{ and} \\
 & & H \models_{\mathcal{M}} \phi \text{ implies } K \mathbin{@_{\mathcal{E}}} H \models_{\mathcal{M}} \psi
\end{array}
$$

Fig. 8. Satisfaction on layered graph models for ILGL

Consider the order given by $G \preccurlyeq G'$ iff $G' \subseteq G$. This has a spatial interpretation: the further up the order, the more specific the location. With this order, we can understand the semantic clause for $\phi \blacktriangleright \psi$ as 'G is contained in a layered graph $H \mathbin{@_{\mathcal{E}}} K$ such that H satisfies ϕ and K satisfies ψ'. Similarly, the clause for $\phi \mathrel{-\!\blacktriangleright} \psi$ states that 'for all subgraphs H of G, if K satisfies ϕ and is layered under H then the layered graph

$H \mathbin{@}_{\mathcal{E}} K$ satisfies ψ'. Finally, $\phi \blacktriangleright\!\!- \psi$ is the dual of the case for $-\!\!\blacktriangleright$, with K instead layered *over* H.

Proof systems for the layered graph logic ILGL can be given as systems of labelled tableaux [40; 43] in the same form as such systems can be given for BI and its modal variants [20] — indeed, the tableaux system for ILGL differs from that for BI only in having an algebra of labels that is neither commutative nor associative (cf. the properties of the labelled-graph constructor, @ and its consequences).

Soundness and completeness with respect to the layered graph semantics is established in [40; 43], with more general approaches based on duality theory being available [41; 42]. Weaker results are available for the classical layered graph logic, LGL [25], for which soundness and completeness results are obtained using rather weak algebraic structures called *magmas* [25].

$$
\begin{array}{rcl}
G[R] \models_{\mathcal{M}} \top & \text{always} & \\
G[R] \models_{\mathcal{M}} \bot & \text{never} & \\
G[R] \models_{\mathcal{M}} \mathrm{p} & \text{iff} & G[R] \in \mathcal{V}(\mathrm{p}) \\
G[R] \models_{\mathcal{M}} \phi \wedge \psi & \text{iff} & G[R] \models_{\mathcal{M}} \phi \text{ and } G[R] \models_{\mathcal{M}} \psi \\
G[R] \models_{\mathcal{M}} \phi \vee \psi & \text{iff} & G[R] \models_{\mathcal{M}} \phi \text{ or } G[R] \models_{\mathcal{M}} \psi \\
G[R] \models_{\mathcal{M}} \phi \rightarrow \psi & \text{iff} & \text{for all } G'[R'] \text{ such that } G[R] \preccurlyeq G'[R'], \\
& & G'[R'] \models_{\mathcal{M}} \phi \text{ implies } G'[R'] \models_{\mathcal{M}} \psi
\end{array}
$$

$$
\begin{array}{rcl}
G[R] \models_{\mathcal{M}} \phi_1 \blacktriangleright \phi_2 & \text{iff} & \text{for some } G_1[R_1], G_2[R_2] \text{ such that } G_1[R_1] \bullet_{\mathcal{E}} G_2[R_2] \preccurlyeq G[R], \\
& & G_1[R_1] \models_{\mathcal{M}} \phi_1 \text{ and } G_2[R_2] \models_{\mathcal{M}} \phi_2 \\
G[R] \models_{\mathcal{M}} \phi -\!\!\blacktriangleright \psi & \text{iff} & \text{for all } G[R] \preccurlyeq H[S] \text{ and all } K[T] \text{ such that } H[S] \bullet_{\mathcal{E}} K[T]\downarrow, \\
& & K[T] \models_{\mathcal{M}} \phi \text{ implies } (H[S] \bullet_{\mathcal{E}} K[T]) \models_{\mathcal{M}} \psi \\
G[R] \models_{\mathcal{M}} \phi \blacktriangleright\!\!- \psi & \text{iff} & \text{for all } G[R] \preccurlyeq H[S] \text{ and all } K[T] \text{ with } K[T] \bullet_{\mathcal{E}} H[S]\downarrow, \\
& & K[T] \models_{\mathcal{M}} \phi \text{ implies } (K[T] \bullet_{\mathcal{E}} H[S]) \models_{\mathcal{M}} \psi
\end{array}
$$

$$
\begin{array}{rcl}
G[R] \models_{\mathcal{M}} \langle a \rangle \phi & \text{iff} & \text{for some well-formed } G[R'] \text{ such that } G[R] \xrightarrow{a} G[R'], G[R'] \models_{\mathcal{M}} \phi \\
G[R] \models_{\mathcal{M}} [a] \phi & \text{iff} & \text{for all well-formed } G[R'] \text{ such that } G[R] \xrightarrow{a} G[R'], G[R'] \models_{\mathcal{M}} \phi \quad \Box
\end{array}
$$

Fig. 9. ILGL with resources and actions

We extend layered graph models to graphs labelled with resources and extend the interpretation of formulae to the action modalities (cf. Stirling's intuitionistic Hennessy–Milner logic [105; 106; 62]) that express resource manipulations. This extension, which quite closely resembles the modal logic LSM described above, introduces a degree of dynamics and statefulness to ILGL — and so enables more direct representations of examples that are about the behaviour of systems — without changing the underlying semantics. Such an extension — which can be interpreted as adding the notions of resource and action to a model based on a notion of location — also provides an explicit connection between the basic logical work and the application of resource semantics to an approach to modelling concurrent and distributed systems that we introduce in Section 8.

For a resource monoid $\mathcal{R}$, a countable set of actions, Act, and a layered graph model $\mathcal{M} = (\mathcal{X}, \mathcal{V})$ over labelled graphs, with the containment ordering on labelled graphs, we generate the satisfaction relation $\models_{\mathcal{M}} \subseteq X[R] \times \mathrm{Form}$ as in Figure 9, in which, having added resources to our models, we can complete an instantiation of our systems

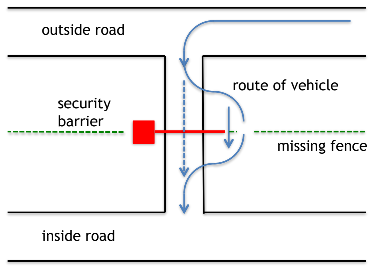

Fig. 10. The security barrier and side channel

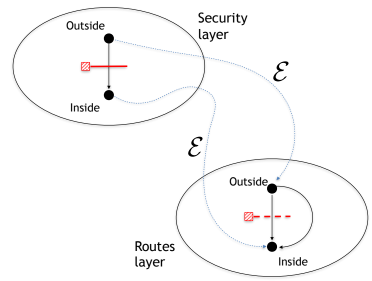

Fig. 11. The layered graph model

modelling approach by adding *action modalities* for possibility and necessity, $\langle a \rangle$ and $[a]$, respectively.

We can use the dynamic and stateful properties of this extension of ILGL to give some examples that will prefigure the application of resource semantics to an approach to the modelling of concurrent and distributed systems that we introduce in Section 8.

The first example (see Figure 10) is a situation highlighted by Schneier [100], wherein a security system is ineffective because of the existence of a side-channel that allows a control to be circumvented. The security policy, as expressed in the security layer, with graph G_1, requires that a token be possessed in order to pass from the outside to the inside; that is, $\langle \text{pass} \rangle (\phi_{\text{inside}} \rightarrow \phi_{\text{token}})$. However, in the routes layer, with graph G_2, it is possible to perform an action $\langle \text{swerve} \rangle$ to drive around the gate, as shown in the Figure 11; that is,

$$G_1 \mathbin{@_{\mathcal{E}}} G_2 \models_{\mathcal{M}} (\langle \text{pass} \rangle (\phi_{\text{inside}} \rightarrow \phi_{\text{token}}) \blacktriangleright \langle \text{swerve} \rangle (\phi_{\text{inside}} \wedge \neg \phi_{\text{token}}))$$

Thus we can express the mismatch between the security policy and architecture to which it is intended to apply.

Our second example concerns an organization which internally has high- and low-security parts of its network. It also operates mobile devices that are outside of its internal network but able to connect to it. Figure 12 illustrates our layered graph model of this set-up. We can give a characterization in ILGL of a side channel that

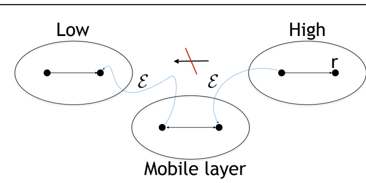

Fig. 12. Organizational Security Architecture

allows a resource from the high-security part of the internal network to transfer to the low-security part via the external mobile connection. Associated with the mobile layer are actions that allow the transference. We have two local compliance properties, in the high- and low-security parts of the network, respectively: $\chi_{\text{high}}(r)$ describes compliance with a policy allowing resource in the high-security network and $\chi_{\text{sec}}(r)$ is a correctness condition that if a resource r is not permitted in the low-security network, then it is not in it. We take actions $\mathrm{copy}, \mathrm{download}, \mathrm{upload}$ associated with the mobile layer G_2, allowing data to be copied to another location as well as moved down and up $\mathcal{E}$-edges respectively, with $\theta(r)$ a compliance property such that $G_2[R] \models_{\mathcal{M}} \langle \text{copy} \rangle \theta(r)$ in order to copy data r. Now we have that

$$G_2[R] \models_{\mathcal{M}} \langle \text{download} \rangle ((\chi_{\text{high}}(r) \blacktriangleright \theta(r)) \wedge \langle \text{copy} \rangle \langle \text{upload} \rangle (\theta(r) \blacktriangleright \neg \chi_{\text{sec}}(r)))$$

showing that the mobile layer is a side channel that can undermine the policy χ_{sec}.

A range of examples of the use of LGL, ILGL's classical variant, can be found in [25; 26].

The logical metatheory of the family of bunched logics, as well as the family of separation logics, has been developed by Docherty and Pym in [40; 41; 42; 43; 44], with fuller elaboration in Docherty's PhD thesis [39]. This work develops, in particular, the theory of Stone-type dualities for the family of bunched logics, connecting their Kripke semantics and algebraic characterizations. As such, it provides a systematic treatment of what we might call the logics of resource semantics. The present article should provide an introduction for readers wishing to explore this theory.

8. BI, PROCESS ALGEBRA, AND CONCURRENCY

Resource semantics has been deployed by O'Hearn and Brookes [87] in the development of Concurrent Separation Logic (CSL) and, for example, this is turn has been deployed by Dreyer and colleagues in reasoning about the safety properties of the programming language Rust; see, for example, [36].

We have previously mentioned our inspiration from modelling the behaviour and properties of distributed systems. We develop an approach to modelling distributed systems that brings together, in a generalized form, all of the components of resource semantics that we have considered so far and which, we conjecture, encompasses CSL and its applications. (Resolving this conjecture would entail interpreting CSL in the logic MBI we describe below in the sense in which Separation Logic is interpreted in Pointer Logic [110].)

The full story of this work is beyond the scope of this article, but can summarize the situation:

- we have considered resource-indexed multiplicative modalities, defined relative to resource-world pairs, which can be seen as resource-labelled worlds;
- graph models, which can be understood as models of location; and
- graph models enriched with resources labelling vertices and action modalities.

The actions employed in action modalities give us an elementary representation of processes. We can, however, adopt a more general approach in which we model distributed systems directly using concepts of

- *location*, for now treated mathematically as directed graphs (though more abstract axiomatizations are possible),
- *resource*, modelled as in BI as, say, PDMs [52], and
- *process*, modelled for now as, essentially, SCCS terms [83; 28; 27; 3].

SCCS is a convenient basis for modelling processes because of its simplicity and generality [83]. Specifically, Robert de Simone's theorem [103] implies that it is able to represent a wide class of forms of concurrent behaviour, including asynchrony.

Given these concepts we can set up a process algebra of models, as described and developed extensively in [28; 27; 3]. The basic idea is that we set up a calculus of locations, resources, and process that coevolve according to an operational semantics:

$$L, R, E \xrightarrow{a} L', R', E'.$$

The basic rule (in the style of Structural Operational Semantics [92; 93]) is for action-prefix in the process terms, $a : E$:

$$\frac{\mu(a, L, R) = L', R'}{L, R, a : E \xrightarrow{a} L', R', E}$$

Notice that this rule is parametrized on a 'modification function' μ that determines the effect of the action a at location L with resources R, returning a new location and new resources. The set of such functions specified in giving a model should be seen as a signature for a model.

Without much loss of generality, we can drop location from our formal set up and so reduce our notational overhead somewhat. The operational semantics rule for concurrent product is

$$\frac{R, E \xrightarrow{a} R', E' \quad S, F \xrightarrow{b} S', F'}{R \otimes S, E \times F \xrightarrow{ab} R' \otimes S', E' \times F'},$$

where $\otimes$ is a monoidal operation on resources, and the rule for sum is

$$\frac{R_i, E_i \xrightarrow{a} R'_i, E'_i}{R_1 \oplus R_2, E_1 + E_2 \xrightarrow{a} R'_i, E'_i}\ i = 1, 2$$

where $\oplus$ is a monoidal operation on resources. Other rules for *hiding*, which associates resource locally with a process, and *recursion* are also required [28; 27; 3].

That the two rules above employ two combinators, $\otimes$ and $\oplus$, on resources derives from our desire to obtain, as described below, a completeness theorem in the sense of van Benthem, Hennessy, and Milner (vBHM) (see, for example, [62; 83; 105; 106; 8]), to the effect that equivalence in a logic of state coincides with bisimulation equivalence of processes. If we work with the form of resource semantics taken in the previous sections, it turns out that vBHM completeness can be obtained only for a fragment of the natural logic of state. Completeness requires sufficient structure on resources to track both concurrent product and choice.

This semantics for distributed systems modelling, together with the treatment of environment mentioned in Section 2 has been implemented both in a bespoke language called Gnosis [27] and in Julia [68], the latter providing the basis for ongoing work with packages available at https://github.com/tristanc/SysModels.

How do models become live? Where do actions come from? The answer really lies in the conceptual approach to distributed systems modelling with which we began. The key component here is environment. Models become live when actions are incident upon their boundaries, either inbound or outbound. Note that bits of a system 'within' a model may also amount to environment; for example, some black-box component.

As an example, we consider Figure 13, which is a picture of the kind of model we might construct.

Figure 13 is a picture of a system model of information-flow security in an office. To the left, we consider the routes that an office worker might take from home to the office building. In the middle, we consider how people access the office through a lobby in which access controls are implemented. To the right, we have the office itself. In each of the three components of the models, there are security vulnerabilities: for example, devices might be lost on the train, unauthorized personnel might circumvent the access control in the lobby, by perhaps tailgating legitimate staff, and, having gained access to the office, they might steal information stored on devices or written on paper, or might shoulder-surf to obtain computer passwords.

The location-resource-process model can then be used to explore, using Monte Carlo simulation, the security consequences of different policies in this set-up. For example,

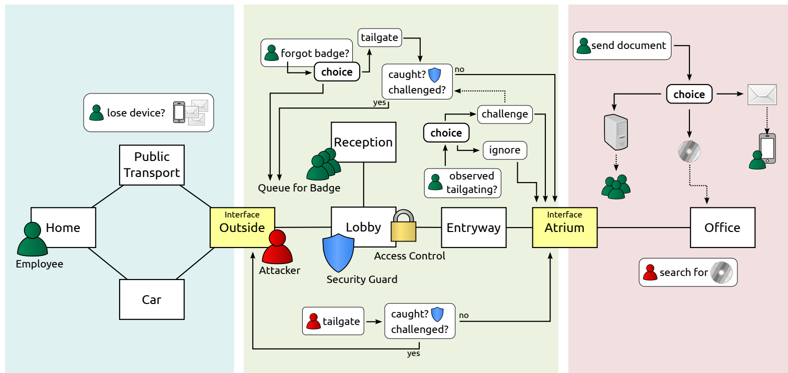

Fig. 13. A system model (for the theory of interfaces, see Caulfield and Pym, Simutools 2015 and IEEE S&P 2015). There are Julia packages for all this stuff:

what are the right levels of staffing of the reception desk and the right numbers of security guards to ensure that neither legitimate staff (who may have forgotten their credentials and need to visit reception) or intruders (who wish to remain unnoticed and uncaptured) are incentivized to try to tailgate through the access control barriers?

Of course, Monte Carlo simulations are not the only way to reason about models. We may also wish to establish logical properties. These may, for example, be assertions about termination, security, or resource consumption, or measures of the utility of policies.

There is a well-established theory of process logics, developed by van Benthem, Hennessy, Milner, Stirling, and many others. See [105; 106; 28; 27; 3] for many references.

In our setting, the basic idea is to set up logics to reason about the resource-semantics models. That is, a logic which is defined by a satisfaction relation of the form

$$L, R, E \models \phi.$$

What is this logic? The details are developed fully in [28; 27; 3], but we can summarize the situation quite efficiently.

- We consider a modal logic of state for this transition system (as before, we drop location for now).
- We will describe a classical version (an intuitionistic version is also available).
- Here is the propositional language:

$$\begin{array}{rll} \phi \ ::= & \mathbf{p} \mid \bot \mid \top \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \rightarrow \phi & \text{classical propositional additives} \\ & \mid \langle a \rangle \phi \mid [a] \phi & \text{classical additive modalities} \\ & \mid I \mid \phi * \phi \mid \phi -\!\!* \phi & \text{propositional multiplicatives} \\ & \mid \langle a \rangle_\nu \phi \mid [a]_\nu \phi & \text{multiplicative modalities} \end{array}$$

Here, the multiplicative modalities are similar to $\langle a \rangle_\bullet$ and $[a]_\bullet$ in LSM [20], described in Section 6, and permit the action to employ unspecified additional resource. The same variations are available here as in LSM;

- In a given model $\mathcal{M}$, a truth judgement, $R, E \models_{\mathcal{M}} \phi$:

$$\begin{array}{rl} R, E \models_{\mathcal{M}} \langle a \rangle \phi & \text{iff some } R, E \xrightarrow{a} R', E', R', E' \models_{\mathcal{M}} \phi \\ R, E \models_{\mathcal{M}} \phi_1 * \phi_2 & \text{iff some } R = R_1 \otimes R_2 \; E_1 \times E_2 = E, \\ & \quad R_1, E_1 \models_{\mathcal{M}} \phi_1 \text{ and } R_2, E_2 \models_{\mathcal{M}} \phi_2 \\ R, E \models_{\mathcal{M}} \langle a \rangle_{\nu} \phi & \text{iff some } S, S' \text{ s.t. } R \otimes S, E \xrightarrow{a} R' \otimes S', E', \\ & \quad R' \otimes S', E' \models_{\mathcal{M}} \phi. \end{array}$$

- Some choices for the last one.

We can also set up both the usual (additive) quantifiers and, perhaps more surprisingly, multiplicative quantifiers.

- For example, the multiplicative existential makes use of the hiding combinator for process terms mentioned above, $\nu S.F$, which associates the resource S locally with process term F, and goes like this:

$$\begin{array}{rl} R, E \models \exists_{\nu} x.\phi & \text{iff there exist } S, F \text{ and } a \text{ s.t. } R, E \sim R, \nu S.F \\ & \text{and } R \circ S, F \models \phi[a/x]. \end{array}$$

We conclude this discussion of distibuted systems modelling by remarking that the desired coincidence between operational equivalence (bisimulation, $R, E \sim S, F$, defined in the evident way [3]) and logical equivalence — in the sense of van Benthem, Hennessy, and Milner — does indeed hold [3]: let E and F be image-finite processes. Then, for any resources R and S,

$$R, E \sim S, F \text{ iff for all } \phi, R, E \models \phi \text{ iff } S, F \models \phi.$$

9. DISCUSSION: LOGIC AS A MODELLING TECHNOLOGY

We have explained the logical theory of the logic BI, the logic of bunched implications [89; 95; 52], and its associated systems. These include the $\alpha\lambda$-calculus [85], modal and epistemic systems, [19; 20; 50] and the layered graph logics [40; 43], which point the way to the general theory of bunched logics [41; 42; 43; 44]. We have also explained how Separation Logic [99] is a model of a specific theory of BI, through BI's 'pointer logic' [65; 110].

We have also explained how BI and its associated logics can be motivated as a basis for logical systems modelling, showing how notions of location, resource, and process arise throughout.

How are models actually built? The answer is that we deploy the classical methods of mathematical modelling, which can be summarized by the picture in Figure 14.

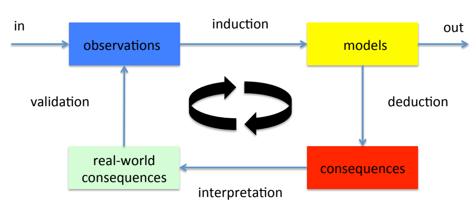

Fig. 14. The classical mathematical modelling cycle

A few remarks on this approach to modelling are perhaps worthwhile.

- Our approach is essentially scale-free: locations, resources, and processes as described build in no commitment to any particular scale.
- So, the abstraction level therefore chosen to fit the problem at hand: models should be as simple as possible, and no simpler. Recall Einstein's Principle: *A scientific theory should be as simple as possible, but no simpler.*
- Predictions about properties of models and the systems they describe can explored using simulations.
- Model checking, using the logics we have described, is also possible (though much less developed at this point).
- The map is not the territory (Alfred Korzybski [72]): models always exclude things that present in the system being modelled.
- Time-value of models: in practice, a less good model obtained quickly can often be more useful than a better model that is only obtainable much later.

Our reflections on the nature of modelling that is the based of Separation Logic have been elaborated in [96]. Briefly, we ask why it is that Separation Logic has been such an effective tool in the development of tools such as INFER [61], now deployed widely at scale in many large companies. Briefly, we argue in [96] that the coincidence of 'engineering' and 'logical' models, allowing formal reasoning techniques, giving precise statements of correctness requirements, to apply directly.

The definition of truth for BI Pointer Logic — that is, its satisfaction relation — provides a first clear illustration of an argument concerning the merging of logic-models and engineering-models. The stack and the heap and the ways in which they are manipulated by programs are considered directly by working programmers: indeed, memory management at this level of abstraction is a key aspect of the C programming language.

Additionally, we have

- that the decomposition of models, via the Frame Rule, manages scale, and
- the coincidence and convenience of the logical and pragmatic value of partiality in the semantics.

These factors lead to an implementable and deployable proof theory, via bi-abduction applied to the Frame Rule. We would suggest that these reasons for the effectiveness of Separation Logic may be reflected in other settings in which logic is deployed as a modelling technology.

The ideas described herein are providing a basis for a substantial research project in systems modelling and verification that is emphasising the concept of *interfaces* — as suggested in Figure 15 — between models as a basis for a compositional theory.

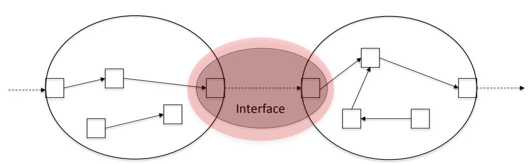

Fig. 15. An interface between models

There are several ways to approach a theory of interfaces in this kind of setting, including that sketched in [17] and approaches based on layered graphs. In all cases, it seems important to establish concepts of local reasoning, supported by forms of Frame Rules [86], about interfaces.

Acknowledgements

I am grateful to Johan van Benthem, James Brotherston, Tristan Caulfield, Jean-René Courtault, Simon Docherty, Kevin Fong, Didier Galmiche, Peter Jipsen, Pierre Kimmel, Alexander Kurz, Daniel Méry, Peter O'Hearn, Jonathan Spring, Hongseok Yang, and Mike Mislove variously for their interest in, comments upon, and contributions to the ideas summarized herein, their encouragement of my writing this article, and for their comments on its contents.

REFERENCES

1. S. Abramsky and J. Väänänen. From IF to BI: a tale of dependence and separation. *Synthese*,167(2):207–230, 2009.
2. G. Allwein and M. Dunn. Kripke Models for Linear Logic. *Journal of Symbolic Logic* 58(2):514–545, 1993.
3. G. Anderson and D. Pym. A calculus and logic of bunched resources and processes. *Theoretical Computer Science* 614:63-96, 2016.
4. R. Anderson and N. Belnap. *Entailment: Logic of Relevance and Necessity, Volume 1*. Princeton University Press, 1992.
5. R. Anderson and N. Belnap. *Entailment: Logic of Relevance and Necessity, Volume 2*. Princeton University Press, 1975.
6. K. Apt. Ten Years of Hoare's Logic: A Survey — Part 1. *ACMTransactions on Programming Languages and Systems* 3(4):431–483, 1981.
7. M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1998. Available at http://www.math.mcgill.ca/triples/Barr-Wells-ctcs.pdf. Accessed 10 March 2019.
8. J. van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2014.
9. E. Beth. Semantic entailment and formal derivability. Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N.R. Vol 18, no 13, 1955, 309–342.
10. N. Bezhanishvili and D. de Jongh. Intuitionistic logic. Technical Report PP-2006-25, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2006.
11. K. Bimbó and J. M. Dunn. *Generalized Galois Logics: Relational Semantics of Non-classical Calculi*. CSLI Publications, 2008.
12. B.Biering, L. Birkedal, and N. Torp-Smith. BI hyperdoctrines and higher-order separation logic. In *Proc. 14th ESOP*, 233–247, Springer-Verlag, 2005.
13. S. Brookes and P. O'Hearn. Concurrent Separation Logic. *ACM SIGLOG News* 3(3), 47–65, 2016.
14. P. Bródka, K. Skibicki, P. Kazienko, and K. Musiał. A degree centrality in multi-layered social network. In *Proc. CASoN '11*, 237–242, 2011.
15. J. Brotherston. Bunched Logics Displayed. *Studia Logica* 100(6), 1223–1254, 2012.
16. J. Brotherston and J. Villard. Sub-Classical Boolean Bunched Logics and the Meaning of Par. *Proceedings of CSL-24*, LIPlcs, Dagstuhl, 325–342, 2015.
17. T. Caulfield and D. Pym. Modelling and Simulating Systems Security Policy. In *Proc. SIMUTools 2015*, ACM Digital Library, ACM Digital Library, 2015.
18. D. Coumans, M. Gehrke, and L. van Rooijen. Relational semantics for full linear logic. *Journal of Applied logic* 12(1):50-66, 2014. doi.org/10.1016/j.jal.2013.07.005
19. J.-R. Courtault and D. Galmiche. A Modal BI Logic for Dynamic Resource Properties. In *Proc. LFCS 2013*, Springer Berlin Heidelberg, 134–138, 2013.
20. J.-R. Courtault, D. Galmiche, and D. Pym. A Logic of Separating Modalities. *Theoret. Comput. Sci.*, 637:30–58, 2016.
21. C. Calcagno, D. Distefano, P. O'Hearn, and H. Yang. Compositional shape analysis by means of bi-abduction. *J. ACM*, 58(6), 2011.
22. L. Cardelli, P. Gardner, G. Ghelli. A spatial logic for querying graphs. In *Proc ICALP '02*, LNCS 2380, 597–610, 2002.
23. D. D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. SIGCOMM '88*, Computer Communication Review, 18(4): 106–114, 1988.
24. B. Coecke, T. Fritz, and R. Spekkens. A mathematical theory of resources. Information and Computation 250:59–86, 2016.
25. M. Collinson, K. McDonald, and D. Pym. A substructural logic for layered graphs. *J. Log. Comp.*, 24(4):953–988, 2014.

26. M. Collinson, K. McDonald, and D. Pym. Layered graph logic as an assertion language for access control policy models. *J. Log. Comp.*, 27(1):41–80 2017.
27. M. Collinson, B. Monahan, and D. Pym. *A Discipline of Mathematical Systems Modelling*. College Publications, 2012.
28. M. Collinson and D. Pym. Algebra and logic for resource-based systems modelling. *Math. Struc. Comp. Sci.*, 19(5):959–1027, 2009.
29. M. Collinson and D. Pym. Algebra and logic for access control. *Formal Aspects of Computing* 22(2): 83–104, 2010. Erratum: *Formal Aspects of Computing* 22(3):483–484, 2010.
30. G. Conforti, D. Macedonio, and V. Sassone. Spatial logics for bigraphs. In *Proc. ICAP '05*, LNCS 3580, 766–778, 2005.
31. G. Coulouris, J. Dollimore, T. Kindberg, and G, Blair. *Distributed Systems: Concepts and Design*. Pearson, 2011.
32. D. Coumans. Generalising canonical extension to the categorical setting. *Ann. Pure. Appl. Log.*, 163(12):1940–1961, 2012.
33. J.-R. Courtault and D. Galmiche. A modal separation logic for resource dynamics. *Journal of Logic and Computation*, doi:10.1093/logcom/exv031, 2015.
34. A. Fiat, D. Foster, H. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan. Competitive algorithms for layered graph traversal. *SIAM Journal on Computing*, 28(2):447–462, 1998.
35. D. van Dalen. *Logic and Structure*. 4th Edition. Universitext, Springer, 2008.
36. H.-H. Dang, J.-H. Jourdan, J.-O. Kaiser, and D. Dreyer. RustBelt Relaxed. Submitted for publication, November 2018.
37. B. Day. On closed categories of functors. In: S. Mac Lane, editor, *Reports of the Midwest Category Theory Seminar*. Lecture Notes in Mathematics 137:1–38, 1971.
38. B. Day. An embedding theorem for closed categories. In: A. Dold and B. Eckmann, editors, *Proceedings of the Sydney Category Seminar* 1972/73. Lecture Notes in Mathematics 420:55-65, 1973.
39. S.Docherty. *Bunched Logics: A Uniform Approach.* PhD thesis, University College London, 2019.
40. S. Docherty and D. Pym. Intuitionistic layered graph logic. emphProc.IJCAR 2016. LNAI 9706:469–486, 2016.
41. S. Docherty and D. Pym. A Stone-type duality theorem for Separation Logic via its underlying bunched logics *Electronic Notes in Theoretical Computer Science* 336 (2018) 101–118.
42. S. Docherty and D. Pym. A Stone-type duality theorem for Separation Logic via its underlying bunched logics Logical Methods in Computer Science 15(1) (March 14, 2019), 27:1–27:51. https://lmcs.episciences.org/5284/pdf.
43. S. Docherty and D. Pym. Intuitionistic Layered Graph Logic: Semantics and Proof Theory *Logical Methods in Computer Science* 14(4) (October 31, 2018), 1–36. https://lmcs.episciences.org/4942/pdf.
44. S. Docherty and D. Pym. Modular Tableaux Calculi for Separation Theories In: Baier C., Dal Lago U. (eds) *Foundations of Software Science and Computation Structures*. FoSSaCS 2018. LNCS 10803:441–458. Springer. doi.org/10.1007/978-3-319-89366-2_24.
45. J. M. Dunn and G. Hardegree. *Algebraic Methods In Philosophical Logic*. OUP, 2001.
46. L. Esakia. Topological Kripke models. *Soviet Math. Dokl.* 15, 147–15, 1974.
47. M. Fitting. Tableau methods of proof for modal logics. *Notre Dame J. Fom. Log.*, 13(2):237–247, 1972.
48. Tobias Fritz. Resource convertibility and ordered commutative monoids. *Mathematical Structures in Computer Science* 27(6):850?938, 2017.
49. N. Galatos and P. Jipsen. Distributive residuated frames and generalized bunched implication algebras. *Algebra Univers.*, 78(3):303–336, 2017.
50. D. Galmiche, P. Kimmel, and D. Pym. A Substructural Epistemic Resource Logic. In *Proc. ICLA 2017*, LNCS 10119:106–122, 2017.
51. D. Galmiche and D. Méry. Tableaux and resource graphs for separation logic. *J. Log. Comp.*, 20(1): 189-231, 2010.
52. D. Galmiche, D. Méry, and D. Pym. The semantics of BI and resource tableaux. *Math. Str. Comp. Sci.*, 15(06):1033–1088, 2005.
53. D. Galmiche and D. Méry. Tableaux and resource graphs for separation logic. J. Logic Comput., 20(1):189–231, 2007.
54. J.-Y. Girard. Linear logic. *Theor. Comp. Sci.*, 50(1): 1–101, 1987.
55. R. Goré. Tableau Methods for Modal and Temporal Logics In: D?Agostino M., Gabbay D.M., Hhnle R., Posegga J. (eds) *Handbook of Tableau Methods*, 297–396. Springer, Dordrecht. doi.org/10.1007/978-94-017-1754-0_6.

56. L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Math. Prog.*, 128(1): 123–148, 2011.
57. D. Grohmann and M. Miculan. Directed bigraphs. In *Proc. MFPS XXIII*, ENTCS 173, 121–137, 2007.
58. Reiner Hähnle. Tableaux and Related Methods. In: Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, Springer, 2001, 101–178.
59. Z. Haniková and R. Horčík. The finite embeddability property for residuated groupoids *Algebra Univers.*, 72(1):1–13, 2014.
60. J. Harland and D. Pym. Resource-distribution via Boolean constraints. *ACM ToCL*, 4(1):56–90, 2003.
61. Facebook. Infer. https://fbinfer.com: accessed 10 March 2019. https://code.fb.com/developer-tools/open-sourcing-facebook-infer-identify-bugs-before-you-ship/: accessed 10 March 2019.
62. M. Hennessy and G. Plotkin. On observing nondeterminsm and concurrency. *Proc. 7th ICALP*. LNCS 85:299–309, 1980.
63. C. Hoare. Proof of correctness of data representations. *Acta Informatica* 1:271–281, 1971.
64. W. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. HindIey, editors, *To H. B. Curry: Essays on Combnatory Logic, Lambda-Calculus, and Formalism*, 479–490. Academic Press, 1980.
65. S. S. Ishtiaq and P. O'Hearn. BI as an assertion language for mutable data structures. In *Proc. Principles of Programming Languages '01*, ACM Sigplan Notices 36(3):14–26, 2001.
66. P. Jipsen and C. Tsinakis. A survey of residuated lattices. In *Ordered Algebraic Structures*, Developments in Mathematics 7:19-56, 2002.
67. P. T. Johnstone *Stone Spaces*. Cambridge Studies In Advanced Mathematics 3, CUP, 1982.
68. The Julia Programming Language. https://julialang.org. Accessed 10 March 2019.
69. R. Jung, D. Swasey, F. Sieczkowski, K. Svendsen, A. Turon, L. Birkedal, and D. Dreyer. Iris: Monoids and Invariants as an Orthogonal Basis for Concurrent Reasoning *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 637–650. ACM 2015. 10. 1145/2676726.2676980.
70. R. Jung, R. Krebbers, J.-H. Jourdan, L. Birkedal, and D. Dreyer. Iris from the ground up. Submitted, 2018. Manuscript: https://people.mpi-sws.org/~dreyer/papers/iris-ground-up/paper.pdf. Accessed 10 March 2019.
71. M. Kivelä, A. Arenas, M. Barthelemy, J. Gleeson, Y. Moreno and M. A. Porter. Multilayer networks. *J. Comp. Net.*, 2(3): 203–271, 2014.
72. A. Korzybski. Non-Aristotelian System and its Necessity for Rigour in Mathematics and Physics. Presented to the American Association for the Advancement of Science, New Orleans, Louisiana, 28 December 1931. Reprinted in *Science and Sanity*, 1933, 747–761.
73. S. Kripke. A semantical analysis of intuitionistic logic I. In *Formal Systems and Recursive Functions*, Studies In Logic and the Foundations of Mathematics 40:92–130, 1965.
74. M. Kurant and P. Thiran. Layered complex networks. *Phys. Rev. Lett.*, 96:138701, 2006.
75. Y. Lafont. *Introduction to Linear Logic*. Lecture notes from TEMPUS Summer School on Algebraic and Categorical Methods in Computer Science, Brno, Czech Republic, 1993.
76. J. Lambek. On the calculus of syntactic types. In *Studies of Language and its Mathematical Aspects*, 166–178, 1961.
77. J. Lambek. From categorical grammar to bilinear logic. In P. Schroeder-Heister and K. Došen, editors, *Substructural Logics*, 207–237
78. J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic.* Cambridge University Press, 1986.
79. D. Larchey-Wendling. The formal proof of the strong completeness of partial monoidal Boolean BI. *J. Log. Comp.*, 26(2):605–640, 2016.
80. J. Loeckx and K. Sieber. *Foundations of Program Verification*. 2nd Edition. John Wiley & Sons, 1987.
81. M. Makkei and G. Reyes. *First Order Categorical Logic: Model-Theoretical Methods in the Theory of Topoi and Related Categories.* Lecture Notes in Mathematics 611, 1977.
82. C. Maus, S. Rybacki, and A. M. Uhrmacher. Rule-based multi-level modeling of cell biological systems *BMC Sys. Bio.*, 5(166), doi:10.1186/1752-0509-5-166, 2011.
83. R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science* 25(3):267–310, 1983.
84. R. Milner. *The Space and Motion of Communicating Agents*. CUP, 2009.
85. P. O'Hearn. On Bunched Typing. *Journal of Functional Programming* 13(4), 747–796, 2003.

86. P. O'Hearn. A Primer on Separation Logic. *Software Safety and Security; Tools for Analysis and Verification. NATO Science for Peace and Security Series* 33:286–318, 2012.
87. P. O'Hearn. Resources, concurrency, and local reasoning. *Theoretical Computer Science* 375 (1–3), 2007, 271–307.
88. P. O'Hearn. Separation Logic. *Communications of the ACM* 62(2), February 2019, 86–95. 10.1145/3211968.
89. P. O'Hearn and D. Pym. The logic of bunched implications. *Bull. Symb. Log.*, 5(2):215–244, 1999.
90. C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.
91. A. Paz. A theory of decomposition into prime factors of layered interconnection networks. *Discrete Applied Mathematics*, 159(7):628–646, 2011.
92. G. Plotkin A structural approach to operational semantics, DAIMIFN-19, Computer Science Department, Aarhus University, 1981.
93. G. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming* 60–61, 2004, 3–15.
94. D. Prawitz. *Natural Deduction*. Almqvist and Wiksell, 1965.
95. D. Pym, P. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of BI. *Theor. Comp. Sci.*, 315(1):257–305, 2004. (Erratum: p. 285, l. -12: ', for some $P', Q \equiv P; P'$' should be '$P \vdash Q$'.)
96. D. Pym, J. Spring, and P. O'Hearn. Why Separation Logic Works. *Philosophy and Technology* (2018). https://doi.org/10.1007/s13347-018-0312-8.
97. D. Pym and C. Tofts. A Calculus and Logic of Resources and Processes. Formal Aspects of Computing 18(4):495–517, 2006.
98. S. Read. *Relevant Logic*. Blackwell, 1988.
99. J. Reynolds. Separation Logic: a logic for shared mutable data structures. In *Proc LICS '02*, IEEE Comp. Soc. Press, 55–74 2002.
100. B. Schneier. The weakest link. https://www.schneier.com/blog/archives/2005/02/the_weakest_lin.html. Schneier on Security, https://www.schneier.com, 2005. Accessed 10 March 2019.
101. P. Schroeder-Heister and K. Došen, editors, Substructural Logics. Oxford University Press, 1993.
102. R. Seely. Hyperdoctrines, Natural Deduction and the Beck Condition. *Mathematical Logic Quarterly* 29(10):505-542, 1983.
103. R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoretical Computer Science* 37:245–267, 1985.
104. R. Smullyan. *First-order Logic*. Dover, 1995.
105. C. Stirling. Modal logics for communication systems. *Theoretical Computer Science*, 49:311–347, 1987.
106. C. Stirling. *Modal and Temporal Properties of Processes*. Springer, 2001.
107. M. Stone. The theory of representations of Boolean algebras. *Trans. AMS* 40: 37–111, 1936.
108. J. Väänänen. *Dependence Logic*. Cambridge University Press, 2007.
109. H. Wang, J. Wang and P. De Wilde. Topological analysis of a two coupled evolving networks model for business systems. *Expert Syst. Appl.*, 36(5):9548–9556, 2009.
110. H. Yang and P. O'Hearn. A Semantic Basis for Local Reasoning. In *Proc. FOSSACS 2002*, LNCS 2303:402–416, 2002.