
# Ribbon Proofs - A Proof System for the Logic of Bunched Implications

Bean, Julian Michael Lewis

ISSN 1470-5559

# Ribbon Proofs - A Proof System for the Logic of Bunched Implications

**Julian Michael Lewis Bean**

Queen Mary
University of London
Department of Computer Science

**RR-06-01** January 2006

# Ribbon Proofs – A Proof System for the Logic of Bunched Implications

**Julian Michael Lewis Bean**

Submitted for the degree of Doctor of Philosophy

Queen Mary, University of London

2006

# Ribbon Proofs – A Proof System for the Logic of Bunched Implications

**Julian Michael Lewis Bean**

## Abstract

In this thesis we present ribbon proofs, a proof system for Pym and O'Hearn's Logic of Bunched Implications (**BI**). We describe two key motivations for the system. Firstly, the existing proof theory for **BI** is sequentialized in the style of Gentzen's LJ; there is no existing proof system which works on the level of individual formulae like Gentzen's NJ. Secondly, we believe that proofs in **BI**'s existing proof systems do not do justice to the strong semantic notions of spatiality and resource which are such exciting motiviations for the study of the logic itself.

We present ribbon proofs first informally as a graphical system. We then go on to formalize the system precisely, the main result of the thesis being the proof of its soundness and completeness relative to existing notions of proof for **BI**. We discuss some properties of our formalization and its relation to **BI**'s model theory, and make formal a few geometric intuitions arising from the system. We present an extension of the system used to prove some real-world results from a paper in program logic, and finally a skeletal implementation of the system in ML which was instrumental in the development of the formalization.

Submitted for the degree of Doctor of Philosophy

Queen Mary, University of London

2006

# Contents

# List of Figures

# Acknowledgements

I thank my supervisors, David Pym and Edmund Robinson, for their support and encouragement through all stages of this thesis. I would also like to thank Richard Bornat, for giving me considerable insights into the usefulness of logical reasoning and formal proof, Peter O'Hearn, Cristiano Calcagno and Josh Berdine for sharing with me their enthusiasm and exciting ideas on program verification, Pablo Armelin and Mike Samuels, who have been a sounding board for ideas since the very beginning, and finally my wife Elizabeth, without whose support this thesis would never have been completed.

In this corrected version of the thesis, I also wish to thank my examiners, Gianluigi Bellin and Peter Schroeder-Heister, for their kind and encouraging comments which have been incorporated into this version.

# Chapter 1

# Introduction

---

In this thesis we introduce a novel form of proof for the Logic of Bunched Implications (**BI**)[32]. The structure of these proofs extends the notion of box proofs (or 'Fitch proofs'[10]), but to deal with **BI**'s substructural nature an additional notion of 'ribbon' is used. This technique could be applied in other substructural logics, although we will not attempt that in the present work. Ribbon proofs give us the closest thing available to an analogue of Gentzen's NJ which operates on the level of single formulae.

The Logic of Bunched Implications (**BI**) was introduced by Pym and O'Hearn in [30, 31] as a logic which is proof-theoretically natural, but also has applications to the study of resource usage. The most complete reference for it is Pym's monograph[32]. It has two conjuctions and correspondingly two implications, one pair being intuitionistic and the other being substructural. It has two proof systems, one with natural deduction style introduction and elimination rules for each connective, and one with left and right rules for each connective, but both of these are presented as inference systems for sequents. It has a sound and complete model theory in terms of partially ordered partial commutative monoids[14]. It has a cut elimination theorem, and a type calculus extending the $\lambda$-calculus. However, it has no proof system which can be presented, in the style on Gentzen's NJ, on the level of propositions.

When looking at Gentzen's LJ, which is an inference system for sequents, it is common to consider that a underlying a sequent $\Gamma \vdash P$ there is some object $\Phi : \Gamma \vdash P$ such that $\Phi$ is the actual proof. Commonly these proof objects $\Phi$ are considered to be NJ proofs. The proof theory in [32] does not present any candidate object to represent such $\Phi$.

**BI** is presented in [32] and elsewhere as a logic for reasoning about *resource*. With its two conjunctions it can contrast the notion that $A$ and $B$ share resources ($A \wedge B$) and the notion that they have disjoint resources ($A * B$). We are particularly interested in the work of Ishtiaq, O'Hearn, Reynolds, Yang and others ([24, 29, 40]) in which the resources concerned are computational, such as memory cells; this provides a powerful practical motivation for studying the logic. However the two inference systems in [32] do not directly reflect these semantic intuitions about **BI**, and proofs using them do not follow the lines that semantic intuitions suggest.

The core of this thesis is a proof system — ribbon proofs — for **BI** which does work at the level of propositions. It generalizes box proofs (as in Fitch[10]), which are essentially one-dimensional, into two dimensions. The horizontal structure of the proof is used to model the resource-sensitive part of the logic. We will develop this system informally as an attractive graphical notation, and we will claim that it reflects the spatial intuitions fostered by the model theory and applications of **BI**. It also provides a possible candidate to objectify proofs $\Phi : \Gamma \vdash P$.

We go on to give a complete formalization for the system, which we will use to prove in detail that this system is a full (sound and complete) proof system for **BI**. We prove this relative to Pym's system LBI[32]; however the depth of the proof indicates that ribbon proofs are in fact slightly more than merely a re-presentation of LBI proofs. We discuss formal properties like normalization and substitution, although they are not the focus of our work.

We will see that the structure of ribbon proofs intimately involves partial commutative monoids, and we will investigate the extent to which we can use this to build models of **BI** from proofs.

Finally we investigate to what extent the graphical or geometrical nature of ribbon proofs is a notational trick, and we attempt to give some real geometric meaning to them with a redefinition of them explicitly embedded in the plane $\mathbf{R}^2$, in which setting we can restate some simple proof-theoretic results geometrically.

As an application of ribbon proofs, we take up some ideas from a paper by Yang[40] which works in the program logics of O'Hearn et al.[29], an extension of a particular model of **BI**. Yang's paper is a proof of correctness for the Schorr-Waite graph-marking algorithm, working in this model. His proof is worked entirely within the semantic system, and relies on some lemmas which Yang asserts can be easily shown to hold in all models. We present a slightly informal extension of ribbon proofs in which we can in principle prove these lemmas syntactically, and we demonstrate two sizeable ribbon proofs for two of those lemmas.

## 1.1 Proof Systems

The definining characteristic of a formal (that is, syntactic) proof system is that it should be possible by a mere syntactic analysis to classify a candidate proof as being valid, that is, if a proof of a formula (or sequent) exists, the formula is a theorem of the logic under consideration. However, for practical use of proof systems, we often want to search for a secondary property: we want a formal proof system in which proofs as nearly as possible mirror the informal proofs we are accustomed to reading in mathematical works.

Syntactic proof systems for formal logics fall into some broad categories. Perhaps the simplest in presentation are the Hilbert-style systems[21]. A Hilbert system consists of a number of logical formulae accepted to be fundamentally true (axioms) and a number of rules allowing deductions of new formulae from old. Hilbert systems tend to be characterized by very small numbers of rules; for example intuitionistic propositional logic can be presented with *modus ponens* as the only rule of inference. A proof in a Hilbert system is generally presented as a sequence of formulae with each being either a (substitution instance of an) axiom, or being deduced from some earlier formulae by a rule of inference. Hilbert systems are frequently hard to use in practice, and they rarely resemble in any way the common informal methods of proof.

An alternative approach which reduces the emphasis on large sets of known theorems to use as axioms is to instead focus on the connectives of the logic. For each connective # we consider the two questions 'What does it mean to prove $A\#B$?', and 'What can we deduce from $A\#B$?'. In this way we characterize for each connective introduction and elimination rules. This approach yields natural deduction systems, which are most commonly presented as tree-shaped proofs. They normally have no axioms, and instead produce proofs based on certain hypotheses.

The definitive example of a natural deduction system is Gentzen's NJ[16], a natural deduction system for intuitionistic logic. The most important, and most difficult to handle, connective of NJ is the implication $\rightarrow$. The elimination rule causes no problem; *modus ponens* is well understood. The introduction rule, however, introduces the delicate concept of discharging a hypothesis. To prove $A \rightarrow B$ we attempt to prove $B$ with an additional, but temporary, hypothesis $A$. Once $B$ has been proven, the temporary hypothesis $A$ is discharged, we conclude $A \rightarrow B$, and we may not use $A$ elsewhere in the proof. The formalization of this is difficult: there may be multiple occurences of $A$ as a hypothesis, discharged in some cases but not others.

The central role of hypotheses and their discharge leads one to consider *sequents* which con-

cisely indicate which hypotheses have been used to prove a given formula. A sequent $A_0, A_1, \ldots, A_n \vdash B$ means that $B$ has been proved from the hypotheses $A_i$. It is possible to present the rules of natural deduction in sequent form. However, considering the structure of sequents themselves there is another natural system; one with 'left' and 'right' rules for each connective, such as Gentzen's LJ. This approach has the merit that, reading a proof upwards, only one rule – Cut – introduces formulae that are not subformulae of some part of the original sequent. Gentzen's Haupsatz[16] says that Cut can be eliminated without affecting the strength of the system. This means that any formulae occuring anywhere in a proof without Cut must be subformulae of some part of the conclusion, which gives strong intuitions about the complexity of proof-search and indeed suggests algorithms for proof search in some cases. The sequential system brings to the foreground the 'structural rules', which are implicit in natural deduction. Weakening and contraction allow hypotheses to be used more than once or not at all; exchange makes the order of hypotheses irrelevant. Finally the rule of 'cut' witnesses the way natural deduction proofs can be composed to form larger proofs.

Given that, as Gentzen showed, LJ and NJ both describe the same logic, there is a second way of interpreting proof rules in LJ. If a sequent holds in LJ, there must be a proof of that sequent in NJ. The LJ rules can now be read as stating the existence of recipes for creating new NJ proofs from old.

For **BI**, Pym describes two proof systems **NBI** and **LBI**, named by analogy with NJ and LJ. However, the nature of the logic and the more complex handling of hypotheses into bunches mean that both these systems are necessarily presented in sequential form. It is hard to imagine a direct analogue of NJ for **BI**, since the very nature of NJ is to internalize weakening and contraction, which in **BI** we need to control more carefully. The system of ribbon proofs which we present here is a natural deduction system working on the level of propositions rather than sequents, like NJ, and it uses the notion of ribbon to represent the bunch structures and control the structural rules.

## 1.2 Substructural Logics

Substructural logics[36] are those logics which restrict or remove entirely some of the structural rules; that is, weakening, contraction, exchange and cut.

Behind much of the research into these logics lie philosophical objections to these rules. The

rule of weakening permits the proof of the theorem $Q \to (P \to P)$, but in what sense does $Q$ really imply $P \to P$? The truth or otherwise of $Q$ seems irrelevant to the truth of $P \to P$. In this tradition lies the family of 'relevant'[2] logics. It is in the domain of relevant logic that the notion of bunches, so central to **BI**, arose — as in Dunn[9], to whom credit is also given for the use of commas and semicolons as distinct separators, as in [1], and developed by Read[34] and Slaney. In that context the problem was to formulate a relevant logic in which the distribution law (of $\wedge$ and $\vee$) was valid, since its standard proof uses weakening in an essential fashion.

Another class of logics which fail to admit the rule of weakening are the non-monotonic logics (see for example [12]). In non-monotonic logics, propositions can be proved 'defeasibly' — that is, provisionally true subject to possibly being later discarded. Such logics are are often discussed in the field of artificial intelligence, where complex computer programs attempt to make inferences based on incomplete or imperfect information.

In AI a particular problem is the 'frame problem'[20, 35]. The frame problem is generally illustrated with a computer program attempting to reason about the changing state of the world, as in the pervasive notion of a hypothetical robot arm moving blocks around. Picture a situation with three blocks available for the arm to manipulate, red, green and blue, and suppose that the red block currently sits on top of the blue block. A formalization of this system into a first-order logic (along the lines of the situation calculus[35]) might include predicates $\mathsf{Above}(r,b), \mathsf{Below}(b,r)$ which hold in this situation, whilst the predicate $\mathsf{Above}(r,g)$ does not hold. The problem occurs when attempting to encode the effect of actions. Consider an action which moves the red block onto the green block. We can see that it will make some predicates hold (e.g. $\mathsf{Above}(r,g)$) and it will make some other predicates hold no longer (e.g. $\mathsf{Above}(r,b)$). But how does it affect the valuation of $\mathsf{Above}(g,b)$?

The typical feature of such systems is that a very large class of facts remain unaltered by any given action; in any sufficiently expressive system, an infinite class. It becomes very difficult to formulate the system so that standard first-order reasoning can be used to make all the valid deductions. One approach to his problem is to use (defeasible) axiom schemes which assume that actions do not change state, except where this is contradicted by the action's own axioms, using some non-monotonic logic. Such approaches are highly problematic; problems include non-local properties such as Fodor's fridgeon[11] property, which holds on any arbitrary object just in that case that Fodor's fridge is on; then a local change (the unplugging of Fodor's fridge)

changes the state of every object in the universe.

We will discuss below how a different kind of substructural logic (based on **BI**) can solve some instances of the frame problem.

The rule of contraction, on the other hand, concerns multiple instances of identical formulae, as in the theorem $X, X \vdash X$. Logics which restrict this rule require, under some circumstances, that all premisses be 'used up'; Girard's Linear Logic[17, 19, 18] lies in this camp. Linear Logic outlaws both weakening and contraction, giving rise to a system in which each premiss must be used exactly once – at least, in a minimal subset of the logic. The logic in fact contains modalities which allow indirect access to the structural rules under very limited conditions. Denying both weakening and contraction has elegant implications in the categorical model of the logic. Intuitionistic logic is well known to have a categorical model[28] in which premiss combination (equivalently, $\wedge$) is interpreted as a cartesian product. In fact, the existence of the projection maps $A_1 \times A_2 \to A_i$ embodies the validity of weakening, and that of the diagonal map $A \to A \times A$ embodies contraction, and we obtain a model of the $\otimes, \multimap$ fragment of Linear Logic simply by using general monoidal categories.

Restricting exchange leads to the consideration of logics where the order of premisses matters. Such logics have been proposed to model notions of language, as in the work of Lambek[27].

**BI** fits into this picture as a logic which permits the rules of weakening and contraction only under some circumstances. It is instructive to decompose **BI** into two sublogics, the $\top, \wedge, \to, \bot$-fragment, and the $I, *, -\!\!*$ fragment. The $\top, \wedge, \to, \bot$-fragment of **BI** is isomorphic to intuitionistic logic, while the $I, *, -\!\!*$ fragment is isomorphic to 'multiplicative intuitionistic linear logic' (i.e. the $I, \otimes, \multimap$-fragment of linear logic). This decomposition is natural in the important sense that the whole logic **BI** is the fibring (in the sense of Gabbay[13]) of these two logics. Each then brings with it its own notion of a premiss combination; we use ';' for the intuitionistic and ',' for the linear. The premisses in **BI** sequents are bunches using these two punctuation marks; and **BI** permits the rules of weakening and contraction for ';' but not for ','.

## 1.3 Logics for resources

One of the intriguing ideas when Girard first described linear logic was that it could be given a semantics in terms of resources which are finite, and which are consumed by their use. This generally illustrated with the notion of money; a system might contain the axioms $\mathsf{euro} \multimap \mathsf{choc}$

('with a Euro I can buy a chocolate') and $\mathsf{euro} \multimap \mathsf{lemonade}$ ('with a Euro I can buy a lemonade'). Then within linear logic you can deduce $(\mathsf{euro} \otimes \mathsf{euro}) \multimap (\mathsf{choc} \otimes \mathsf{lemonade})$ but not $\mathsf{euro} \multimap (\mathsf{choc} \otimes \mathsf{lemonade})$ or $(\mathsf{euro} \otimes \mathsf{euro}) \multimap \mathsf{choc}$.

The semantics of **BI**, on the other hand, is better understood in terms of the notions of sharing and independence[33]. The proposition $P * Q$ should be understood to denote that there is some way of dividing all those resources available into two piles, such that $P$ holds of one pile and $Q$ of the other; in contrast, $P \wedge Q$ should be understood as indicating that $P$ holds with *all* the resources available and $Q$ also holds with all the resources available. In otherwords, in $P * Q$, $P$ and $Q$ are independent in the sense of sharing no resources, whilst in $P \wedge Q$, $P$ and $Q$ share all their resources[1]. Similarly we have the sharing interpretation of $P \to Q$, 'given that $P$ holds of (all) these resources, so does $Q$', and $P \mathrel{-\!\!*} Q$, 'given some (new) resources of which $P$ holds, $Q$ holds of our current resources combined with the new resources'.

It is intriguing to note that Cardelli and Gordon in their work[8] on ambient logics independently developed a semantic system which combined additive and multiplicative connectives equivalent to $\wedge$ and $*$ where the resource concerned is related to location. Further work by Cardelli, Gardner, and Ghelli[7] uses a closely related system to reason about graphs.

A particularly interesting notion of resource returns us to the frame problem. There is an instance of (a restricted form) of the frame problem in computer programs which reason about pointers. Consider a Hoare-style[22] logic in which we maintain assertions describing the state of a computer system. Suppose that one of our assertions denotes '$y$ points to the beginning of a linked list'. Now, if we change the contents of the memory at address $x$, will this alter the validity of the assertion? More generally, can we describe which assertions will be affected by changing the memory at address $x$? Reynolds[37] tackles this problem by the notion of 'separation', and Ishtiaq and O'Hearn[24, 29] pose a rule for Hoare triples called 'Frame Introduction':

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}} \textit{ Frame Introduction}$$

with some side conditions which do not concern us here. The essential content of this rule is based on the idea that the triple $\{P\}C\{Q\}$ guarantees that the program $C$ will only access memory locations described (or guaranteed to exist) by $P$. Under this interpretation it is sound to postulate an arbitrary additional set of memory locations, described in $R$, and conclude that any

[1] Possibly all. We can construct sensible systems in which 'all' is interpreted strictly, that $P$ must use all resources, as well as systems in which $P$ simply uses at most all resources

assertion true of these locations before the execution of $C$ remains true after the execution of $C$.

Our application of ribbon proofs in Chapter 7 works in this setting.

## 1.4 Overview of the thesis

In Chapter 2, we introduce the system of box proofs, a presentation of natural deduction of which ribbon proofs form an extension. We give a formalization of box proofs in the same spirit as our later formalization of ribbon proofs, and make precise the relationship between box proofs and NJ in terms of translations between them. We discuss the problems inherent in expression of normalization in the box proof setting.

In Chapter 3, we introduce the logic **BI**, with informal motivation and then a formal presentation of its grammar, its proof theory, its model theory, and statements of the main theorems about it.

In Chapter 4, we describe the system of ribbon proofs with examples, and then give a formalization of the system. We prove that the system is equivalent in proof-theoretic strength to the conventional proof systems for **BI**.

In Chapter 5, we discuss some properties of ribbon proofs, including how they represent substitution, how they relate to normalization, and how they relate to the partial monoid models of **BI**.

In Chapter 6 we discuss the extent to which the apparently geometric nature of displayed ribbon proofs can be formalized, by giving a formal geometric model for them in $R^2$, and informally 'proving' some simple proof-theoretic results geometrically.

In Chapter 7 we give some examples of a slight extension of ribbon proofs used to prove some lemmas from a published paper in a system derived from BI, handling informally some issues about substitution and quantification.

In Chapter 8 we give a brief overview of a partial implementation of the formalization of ribbon proofs, the source code of which is reproduced in the appendix.

# Chapter 2

# Box Proofs

---

In this chapter we describe a system of ‘box proofs’; a presentation of natural deduction whose roots lie as far back as natural deduction itself. We outline our particular choice of notation, and contrast the system to the more common proof notions for natural deduction. We then describe the precise relationships between the proof forms, and use these relationships to discuss normalization in the context of box proofs.

The mappings developed between box proofs and natural deduction proofs in this chapter will in later chapters be developed into mappings between ribbon proofs and the conventional proof theory of BI.

## 2.1 Introduction

Box proofs are a presentation of natural deduction widely used for teaching intuitionistic logics and proofs[4, 6, 38, 3, 23]. Natural deduction, as most logicians use the term, was formalized by Gentzen, who called the system NJ[16]. The system distinguished itself from earlier systems with use of introduction and elimination rules for each connective, as opposed to a having just a small number of rules coupled with a set of axioms, as was the norm for Hilbert-style systems.

In the same paper, Gentzen also formalized LJ, a system which used the novel notion of sequents to manage formulæ, with left and right rules for each connective.

NJ and LJ, like any inference system which contains binary rules, generate tree-shaped proofs. Box proofs, which are a close relative of NJ and share its rules, are a linearization of this tree structure. It can be argued that they more closely mirror the common form of informal math-

ematical proof, which consists of a sequence of sentences in a natural language such as English, each derivable from some of the earlier sentences. Gentzen was certainly aware that his system could be represented this way, and he had various reasons for his choice of a tree-shaped presentation. Fitch's proofs ([10], also used extensively by Thomason[39]) and Jaskowski's 'method of suppositions'[25] are both very close to our style of proof.

$$\dfrac{\dfrac{\dfrac{A \quad A\to B\wedge C}{B\wedge C}\to\text{-}elim}{B}\wedge\text{-}elim \qquad \dfrac{\dfrac{A \quad A\to B\wedge C}{B\wedge C}\to\text{-}elim}{C}\wedge\text{-}elim}{\dfrac{B\wedge A}{B\wedge A\wedge C}\wedge\text{-}intro}\wedge\text{-}intro$$

| | | |
|---|---|---|
| 1. | $A$ | hypothesis |
| 2. | $A\to B\wedge C$ | hypothesis |
| 3. | $B\wedge C$ | $\to$-*elim* 1,2 |
| 4. | $B$ | $\wedge$-*elim* 3 |
| 5. | $B\wedge A$ | $\wedge$-*intro* 4,1 |
| 6. | $C$ | $\wedge$-*elim* 3 |
| 7. | $B\wedge A\wedge C$ | $\wedge$-*intro* 5,6 |

Figure 2.1: NJ and Box proofs contrasted

Compare the two presentations of the proof in Fig 2.1. Both are proofs (the same proof, in an important sense) of $A, A\to B\wedge C \vdash B\wedge A\wedge C$. Note that the NJ proof mentions the hypothesis $A$ three times, whilst the box proof mentions it only once. Furthermore, the NJ proof proves the formula $B\wedge C$ twice, using exactly the same proof both times, whilst the box proof proves it only once. The dependency information given by the line numbers in the justifications shows us how the box proof can be 'unpacked' into the corresponding tree proof, where this information is represented geometrically, at the cost of repeating twice the proof of $B\wedge C$.

The feature of box proofs which gives them their name is the treatment of assumptions, or *discharged hypotheses*. NJ requires, in some of its rules, the use of a hypothesis which is later discharged. The canonical example is $\to$-*intro*. Box proofs use rectangular nested boxes to indicate the scopes of these discharged hypotheses which, in the context of box proofs, are called assumptions. Examine the box proof in Figure 2.2. The conclusion in the final line is to be proved using the rule $\to$-*intro*. The standard treatment of this is to add to the proof as a hypothesis the antecedent of the $\to$ — in this case, $A\wedge(B\wedge C)$; but to 'mark' this hypothesis as being discharged by the $\to$-*intro* use. This marking is the most technically inelegant (and

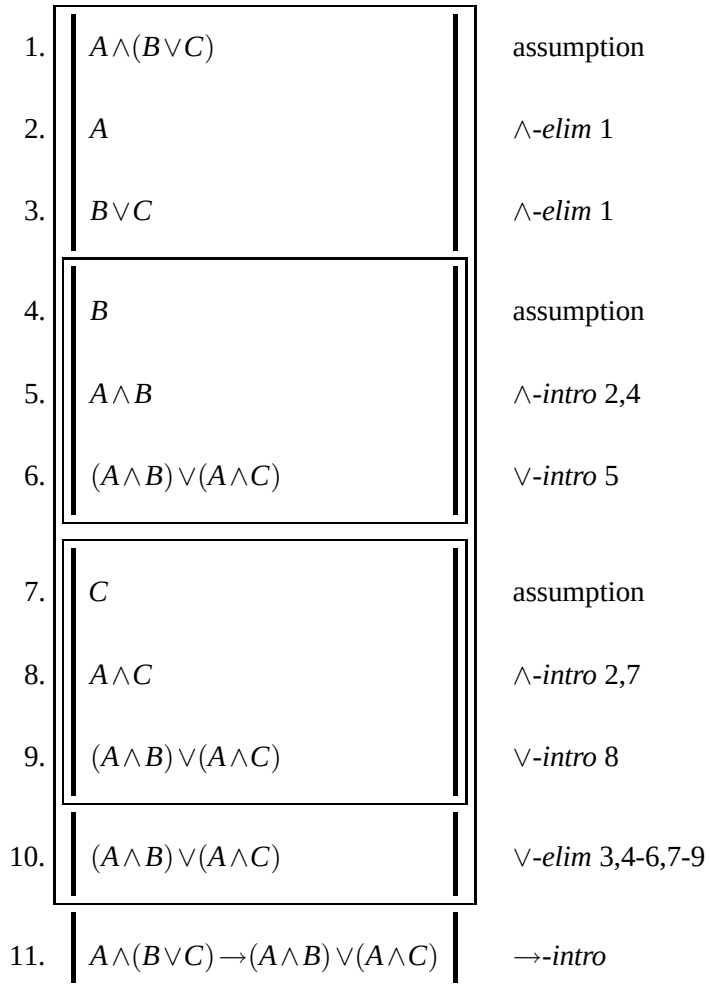



Figure 2.2: A box proof using nested boxes

$$\dfrac{\dfrac{[A\wedge(B\vee C)]}{B\vee C}\wedge\text{-}elim \qquad \dfrac{\dfrac{\dfrac{[A\wedge(B\vee C)]}{A}\wedge\text{-}elim \qquad [B]}{A\wedge B}\wedge\text{-}intro}{(A\wedge B)\vee(A\wedge C)}\vee\text{-}intro \qquad \dfrac{\dfrac{\dfrac{[A\wedge(B\vee C)]}{A}\wedge\text{-}elim \qquad [C]}{A\wedge C}\wedge\text{-}intro}{(A\wedge B)\vee(A\wedge C)}\vee\text{-}intro}{(A\wedge B)\vee(A\wedge C)}\vee\text{-}elim}{A\wedge(B\vee C)\rightarrow(A\wedge B)\vee(A\wedge C)}\rightarrow\text{-}intro$$

Figure 2.3: The proof of Figure 2.2 in NJ form

often obscurely presented) part of the NJ system. It was perhaps never truly handled properly until term-labelling systems. In a box proof, we draw a box and keep the hypothesis in the box. The box is then a representation of the scope of the particular discharged hypotheses to which it relates. In Figure 2.3, we show one presentation of the same proof in NJ; we have marked discharged hypotheses with brackets. Note that there is no visible link between point of discharge and the hypotheses (some authors suggest a numbering scheme for this); in this particular example there is no great ambiguity but in a proof of $A \to A \to A \wedge A$, some additional notation is necessary to indicate which $A$ is used when.

The boxes make the Deduction Theorem appealingly obvious; the proof of $\Gamma \vdash P \to Q$ can be 'derived' graphically from the proof of $\Gamma, P \vdash Q$ simply by drawing a box around the main part of the proof and adding the $\to$-*intro* step. Box proofs, being a form of natural deduction, don't have a Cut rule. However, the admissibility of the Cut rule in sequent calculus is witnessed by a simple vertical composition of proofs: given proofs of $\Gamma \vdash P$ and $P \vdash Q$ you can form a proof of $\Gamma \vdash Q$ by adjoining the proofs vertically (overlapping on the $P$).

The box proof system pays a price for the notational convenience it offers: it is not easy to formulate a notion of normal proof. However, from the provability perspective, they are equivalent to natural deduction:

**Proposition 1** (Relative soundness and completeness for box proofs)**.** *Box proofs give rise to precisely the same of theorems as natural deduction.*

We give one possible proof of this proposition below. We also observe that both NJ and box proofs admit a simple-minded truth-valued semantics. The appearance of a formula $P$ in a proof denotes '$P$ is true, and we have proved it from the true formulæ above it'. Or, in the presence of discharged hypotheses/assumptions, '$P$ is true assuming $A, B, C, \ldots$, and we have proved it from the true formulæ above it', where $A, B, C, \ldots$ are any assumptions applicable at $P$.

## 2.2 Formally relating box proofs to NJ

We give formal mappings between box proofs and natural deduction, and use them to prove that box proofs do indeed represent the same system (prove exactly the same set of theorems).

Firstly we need a formal notion of box proof. We work with an auxiliary notion — a 'box structure' — of which box proofs will form a special case.

**Definition 1.** *Define the following:*

- *A* box structure *consists of a single* box*, which we will refer to as the* outermost *box of the structure;*
- *A* box *is sequence of* lines *and* boxes*;*
- *A* line *is a pair* $\langle f, j \rangle$ *of a* formula *and a* justification*;*
- *A* formula *is a formula of intuitionistic logic; in this thesis we are only concerned with the propositional fragment;*
- *A* justification *is either the name of natural deduction rule (such as* $\rightarrow$*-intro) or one of the special justifications* assumption *or* from line*, along with some* references*;*
- *The* references *indicate which lines and boxes were used as premisses for the rule.*

*We assume some sensible line- and box-numbering system for the references. The number of* references *is the number of premisses the natural deduction rule takes; the special justification* from line *is unary, and* assumption *is nullary. Note that* $\rightarrow$*-intro, for example, is a unary rule, its single premiss being a box (the subproof).*

Box proofs will be represented by box structures obeying certain well-formedness conditions.

**Definition 2.** *The* scope *of a line l in a box b in a box structures contains all lines in b after l, and all lines in boxes within b which themselves occur after l, and recursively all lines within boxes within those boxes. Conversely, a line is said to be* visible *from those lines in its scope.*

**Definition 3.** *A line in a box structure is* well-justified *if it is the conclusion of the natural deduction rule named as its justification, with premisses given by the formulæ in the lines referred to by the references, and the references are visible from the line. Where a premiss in the natural deduction rule is a subproof (as in the case of* $\rightarrow$*-intro), the corresponding reference will refer to a box which contains as its only assumption the discharged hypothesis, and as its final line the conclusion of the subproof. A* from line *is well-justified if it contains precisely the formula in the line referred to by the reference.* Assumptions *are well-justified if they occur in a group at the beginning of a box (that is to say, preceded by no non-assumption line).*

**Definition 4.** *A box in a box structure is* well-placed *if it is used as a premiss by exactly one line.*

The definitions above are intended to ensure that the proof has indeed been correctly derived by the rules of natural deduction. They also prevent spurious boxes; this is not vital but harmless. Now we can define a box proof:

**Definition 5.** *A* box proof *is a box structure in which every line is well-justified, and every box except the outermost is well-placed.*

There are a few points to note respecting the correspondence between this formalization and our informal notation for box proofs. We never actually draw the outer box, and we call the assumptions in this outer box hypotheses. The referencing scheme we use in our figures is linear numbering and ignores the nested box structure; the formalization would suggest a nested numbering scheme (and indeed some authors use such). All that matters for the theory is that some unique identifier be attached to each line and box.

Finally, we note the convention that the final line of a box proof is its conclusion:

**Definition 6.** *A box proof is said to be the proof of the sequent $\Gamma \vdash P$, where $\Gamma$ is the list of assumptions in the outermost box (the 'hypotheses'), and P is the formula in the final line.*

Now we wish to formalize the sense in which these proofs are indeed natural deduction proofs. We exhibit maps between box proofs and natural deduction proofs.

**Definition 7.** *Let the set of all box proofs be **BP**. Let the set of all NJ proofs be denoted **NJ**.*

**Definition 8.** *We define a map $\psi : \mathbf{BP} \longrightarrow \mathbf{NJ}$. Fix a proof $p \in \mathbf{BP}$. We firstly define an auxiliary map $\hat{\psi}_p$ which assigns to each line of p, and each box of p, a proof in **NJ**. We work by induction over the lines and boxes in the proof.*

*For a line $l = \langle P, j \rangle$ in a proof, we assign a proof $\hat{\psi}_p(l) \in \mathbf{NJ}$ as follows:*

- *If j is* assumption*, then the proof $\hat{\psi}_p(l)$ is the axiom proof of P using hypothesis P;*
- *If j is* from line *referencing $l'$, then the proof $\hat{\psi}_p(l) = \hat{\psi}_p(l')$, already defined by induction;*
- *If j is a rule of natural deduction referencing premisses $\{x_i\}$, then the proof $\hat{\psi}_p(l)$ is constructed by taking the proofs $\{\hat{\psi}_p(x_i)\}$, already defined by induction, and combining them using the rule j. That this indeed makes a wellformed NJ proof is guaranteed by the notion of well-justified lines. If $x_i$ is a line, then the premiss $x_i$ is attached to the natural deduction rule using the proof $\hat{\psi}_p(x_I)$. If $x_i$ is a box, then we attach the proof formed from $\hat{\psi}_p(x_i)$ by discharging all instances of its assumption to the natural deduction rule.*

*For a box b, we assign proofs to all lines and boxes inside it, and then set* $\hat{\psi}_p(b) = \hat{\psi}_p(l)$*, the proof of its final line.*

*Now we define* $\psi(p) = \hat{\psi}_p(l)$ *where l is the last line of the proof.*

We remark that the construction of $\psi$ will ignore lines which are not used in the deduction of the conclusion.

**Proposition 2** (Relative Soundness)**.** *Given a box proof p of* $\Gamma \vdash P$*, there is a correct natural deduction proof of* $\Gamma \vdash P$*.*

*Proof.* By induction over the number of lines in the proof *p*, $\psi(p)$ is such a correct natural deduction proof of $\Gamma \vdash P$. □

We call this proposition the 'relative soundness' of box proofs: box proofs are sound relative to natural deduction in the sense that, given a box proof of a theorem, we can construct a natural deduction proof of that theorem. Since we wish to show that box proofs are exactly the same strength as natural deduction, we now need to show a relative completeness result.

**Definition 9.** *We define a map* $\bar{\psi} : \textbf{NJ} \longrightarrow \textbf{BP}$*, by induction on the structure of* ***NJ*** *proofs. We view the* ***NJ*** *rules as the constructors for an inductive notion of proof, taking the sequent view in which each rule constructs a proof of a sequent that is its conclusion from (proofs of) sequents that are its premisses. For each of the rules we will show how to construct a* ***BP*** *proof of the conclusion, using* ***BP*** *proofs of the premisses. Note that the* ***BP*** *proof of a premiss will necessarily have the active formula in the premiss itself occurring as the final line.*

- *For the trivial* ***NJ*** *proof of the axiom* $P \vdash P$*, we use the one-line* ***BP*** *proof with hypotheses P.*

- *We treat the following rules together:* $\wedge$*-intro,* $\wedge$*-elim,* $\vee$*-intro,* $\rightarrow$*-elim. We begin with the* ***BP*** *proof(s) of the premiss(es). In the case that there are two premisses, we place one proof after the other, but we move the hypotheses of the second proof to be just after the hypotheses of the first; and we coalesce identical hypotheses. Naturally references are renumbered as appropriate. Now the conclusion is added on a new line, and it follows from the* ***BP*** *rules with the same name as the* ***NJ*** *rule used, from the formulæ occuring in the last lines of the premiss proof(s).*

- *The rule $\rightarrow$-intro has as its premiss a subproof. That subproof may contain two kinds of hypotheses: discharged, and non-discharged. We place the all of the subproof except the non-discharged hypotheses in a box, with the non-discharged hypotheses outside the box. $\vee$-elim is treated analogously.*

Now, from this definition we obtain relative completeness:

**Proposition 3.** *Given a **NJ** proof of $\Gamma \vdash P$, we can construct a box proof of $\Gamma \vdash P$.*

*Proof.* For **NJ** proof $p$, by structural induction over the rules used in $p$, $\bar{\psi}(p)$ is a box proof, and it certainly has conclusion $P$. Note that, following the construction above, each hypothesis in $p$ certainly generates a hypothesis in $\bar{\psi}(p)$, and these are the only hypotheses. Therefore, $\bar{\psi}(p)$ is a proof of $\Gamma \vdash P$. □

The translations $\psi$ and $\bar{\psi}$ are not inverse; there is no bijection between box proofs and natural deduction proofs. There is a one-sided inverse: $\psi \circ \bar{\psi}$ is an identity on **NJ**. The other composition $\bar{\psi} \circ \psi$ is therefore idempotent on **BP**, so it identifies particular subset of box proofs which are in bijection with natural deduction proofs.

With these maps in mind, we are in a position to consider what normalization means for box proofs.

## 2.3 Normalization for Box Proofs

We will consider four reductions on **NJ**: the $\beta$ and $\eta$ rules for $\wedge$ and $\rightarrow$. Let us consider in detail $\beta$ for $\wedge$ first of all.

$$\dfrac{\dfrac{\dfrac{\vdots\, p_A}{A} \quad \dfrac{\vdots\, p_B}{B}}{A \wedge B}\,\wedge\text{-}intro}{A}\,\wedge\text{-}elim \quad \longrightarrow \quad \dfrac{\vdots\, p_A}{A}$$

Figure 2.4: $\beta$-reduction for $\wedge$ in **NJ**

The $\beta$-reduction for $\wedge$ is shown in Figure 2.4. $p_A$ and $p_B$ stand for the proofs of $A$ and $B$ respectively. Note how the reduction eliminates not only the formula $B$, but also its entire proof $p_B$.

In the equivalent box-proof situation, with a $\wedge$-*elim* rule operating on a premiss which was itself proved by $\wedge$-*intro*, there can be serious consequences of the removal of the formula $B$. In a

| | | |
|---|---|---|
| 1. | $A$ | hypothesis |
| 2. | $B$ | hypothesis |
| 3. | $A \wedge B$ | $\wedge$-*intro* 1,2 |
| 4. | $B$ | $\wedge$-*elim* 3 |
| 5. | $A \vee C$ | $\vee$-*intro* 1 |

Figure 2.5: $\beta$-reduction issues in box proofs

box proof, a single formula can be used as the premiss of a rule more than once; this means that if $B$ was used as a premiss for some other rule, removing it will destroy the proof. Furthermore, any of the formulæ in the proof $p_B$ might have been used as a premiss for some rule not itself part of $p_B$, and therefore $p_B$ can't just be removed. For example, consider Figure 2.5, which is a particular case. Simply removing the formula $A$ from line 1 would invalidate the proof as it was used not only as a premiss for the $\wedge$-*intro* in line 3 but also the $\vee$-*intro* in line 5.

There are some proofs in which $B$ and the contents of $p_B$ are not used in this way. In particular, all proofs lying in the image of $\bar{\psi}$ have this property; however, those are not the only such. We can clearly '$\beta$-reduce' a box proof by applying the composition $\bar{\psi} \circ \beta \circ \psi$; but in general this will not be a 'local' change to the proof.

$$\dfrac{\dfrac{\dfrac{\vdots\, p_{A\wedge B}}{A\wedge B}}{A}\wedge\text{-}elim \quad \dfrac{\dfrac{\vdots\, p_{A\wedge B}}{A\wedge B}}{B}\wedge\text{-}elim}{A\wedge B}\wedge\text{-}intro \quad\longrightarrow\quad \dfrac{\vdots\, p_{A\wedge B}}{A\wedge B}$$

(a) $\eta$-reduction for $\wedge$

$$\dfrac{\dfrac{[A]}{\vdots\, p_B}}{\dfrac{B}{A\to B}\to\text{-}intro \quad \dfrac{\vdots\, p_A}{A}}{B}\to\text{-}elim \quad\longrightarrow\quad \dfrac{\vdots\, p_A}{\dfrac{A}{\vdots\, p_B}{B}}$$

(b) $\beta$-reduction for $\to$

$$\dfrac{\dfrac{\vdots\, p_{A\to B}}{A\to B} \quad [A]}{\dfrac{B}{A\to B}\to\text{-}intro}\wedge\text{-}elim \quad\longrightarrow\quad \dfrac{\vdots\, p_{A\to B}}{A\to B}$$

(c) $\eta$-reduction for $\to$

Figure 2.6: Reductions in **NJ**

The remaining reductions are shown in Figure 2.6. The same issues apply with all of them; when formulæ or entire proofs are excised by a reduction, the corresponding operation on box proofs might fail to produce a valid proof.

So, what can we say about normalization for box proofs? I suggest the following:

**Definition 10.** *A **BP** proof $p$ is normal if it is of the form $\bar{\psi}(q)$ for some normal proof $q \in$ **NJ**. The normal form of a proof $p \in$ **BP** is the proof $\bar{\psi}(q)$, where $q$ is the normal form of $\psi(p)$.*

This is feasible because of the way in which we have defined our translation from **NJ** to **BP**, and gives us unique normal forms, at least. An alternative approach would be to define $p$ to be normal if $\psi(p)$ is normal; this would give $\bar{\psi} \circ \psi$-equivalence classes instead of unique normal forms.

# Chapter 3

# The Logic of Bunched Implications(BI)

---

In this chapter, we describe briefly the Logic of Bunched Implications. We introduce it with some suggestions on how the logic can be thought of at an informal, intuitive level. We then describe the logic formally, with a grammar, a proof theory, and a formal semantics.

## 3.1 Introduction

### 3.1.1 Splitting conjunction

The Logic of Bunched Implications, **BI**[32], is a logic in which two different senses of the word 'and' coexist on equal footing. Compare the following two statements:

- Spurs is a great team *and* enjoyable to watch;
- In this season's squad, Spurs had a world-class goalkeeper *and* an excellent striker.

The first example is characteristic of the kind of conjunction well modelled by traditional formal approaches such as classical and intuitionistic logic. The second example, however, carries a slightly different sense. It strongly implies, at least, that the goalkeeper and the striker concerned are *distinct* individuals — they cannot be the same person. This sense would be lost in a standard translation into intuitionistic logic where 'and' is simply interpreted by $\wedge$, as in

$$(\exists x.\mathit{PlaysFor}(\mathit{Spurs},x) \wedge \mathit{WorldClassGoalkeeper}(x)) \wedge$$
$$(\exists y \mathit{PlaysFor}(\mathit{Spurs},y) \wedge \mathit{ExcellentStriker}(y))$$

The sense of conjunction we describe, which is equally natural if slightly less common in English, we shall term (following [29],...) a *spatial* conjunction. This is true *over here* and that is true *over there*. We use the standard symbol $\wedge$ for the non-spatial conjunction (because it behaves exactly like the standard conjunction), and for the spatial conjunction we use the symbol $*$. It has considerable importance in careful reasoning about resources: Spurs may just possibly have a player who can play very well in both of the positions mentioned, but he certainly cannot play both positions at the same time. This occurence; of resources which can play more than one role but only one at a time is endemic in the study of resource sensitive systems.

We will formalize this idea with a Kripke-style possible worlds semantics, as used for intuitionistic logic and some modal logics. We define a forcing relation on worlds $w$. The standard definition for the $\wedge$ connective is $w \models P \wedge Q$ iff $w \models P$ and $w \models Q$. For $w \models P * Q$, we want to capture the notion that some part of $w$ is sufficient to force $P$, and the remaining part will force $Q$. We add to the set of possible worlds a combining (monoidal) operation $\cdot$, and then we say $w \models P * Q$ if and only if $w$ is of the form $u \cdot v$ where $u \models P$ and $v \models Q$.

### 3.1.2 Splitting Implication

The second novel feature of **BI** is the presence of two *implications* on an equal basis. Just as to $\wedge$ there corresponds the implication $\rightarrow$, to $*$ there corresponds an implication, written $-\!\!*$. The implications are defined by following *adjointness*: The sequent $A \vdash B \rightarrow C$ holds if and only if $A \wedge B \vdash C$ holds, and similarly $A \vdash B -\!\!* C$ holds if and only if $A * B \vdash C$ holds.

In terms of the intuition given above for $*$ and $\wedge$, $\rightarrow$ is the ordinary logical implication, as in, 'If the weather is good tomorrow, we should win the match'. The symbol $-\!\!*$ (pronounced 'magic wand') talks of the (hypothetical) introduction of something new, as in 'If we could sign a good defender, we would do much better'.

### 3.1.3 Restricting the structural rules

**BI** captures the difference between these two senses of conjunction by controlling two of the so-called *structural* rules in its proof theory. Recall that intuitionistic logic[1] has the rules of Weakening and Contraction. Weakening says that, if we proved $P$ from hypotheses $A_0, A_1, \ldots$, then there is also a proof from $B, A_0, A_1, \ldots$; more precisely, it says that there is a canonical such

[1] In fact, the comments in this section about conjunction and implication are equally applicable to classical logic, since we are not concerned here with the behaviour of negation. We continue to refer to intuitionistic logic since that will be the variant we focus on for most of the thesis

proof, namely the one in which (this occurence of) $B$ is not used. Contraction says that given a proof of $P$ from $A_0, A_0, A_1, \ldots$, there is a proof from $A_0, A_1, \ldots$.

Logics which omit or control these rules have been widely studied, most famously in Girard's Linear Logic[17], which contains a conjunctive connective $\otimes$ which is multiplicative. Indeed, like **BI**, Linear Logic contains two conjunctions, one additive and one multiplicative. **BI**'s innovation, then, is the equal footing on which it places the two conjunctions. Throughout this thesis we follow Pym in referring to $*, -\!\!*$ as multiplicative and $\wedge, \rightarrow$ as additive. These terms more precisely refer to the presentation of the proof rules (as in linear logic) and we abuse terminology slightly to continue using them to distinguish between the connectives although our proof system actually chooses to present both connectives multiplicatively. A more exact terminology might be "intensional" and "extensional", and the two strongest parallels do not lie with linear logic but rather with other bunch logics carrying two families of connectives.

Intuitionistic proof theory can be presented in the form of sequents[16], where the informal interpretation of $A_1, A_2, \ldots, A_n \vdash C$ is that from the hypotheses $A_i$, we can prove the conclusion $C$. The comma used to separate the hypotheses $A_i$ has the sense of *and*: from $A_1$ and $A_2$ and $\ldots$ we can prove $C$. Indeed, the sequent $A_1, A_2, \ldots, A_n \vdash C$ is valid if and only if the sequent $A_1 \wedge A_2 \wedge \ldots \wedge A_n \vdash C$ is valid. In Girard's Linear Logic, on the other hand, the commas used in the similar sequents represent the linear conjunction $\otimes$, in the sense that the sequent $A_1, A_2, \ldots, A_n \vdash C$ is equivalent to the sequent $A_1 \otimes A_2 \otimes \cdots \otimes A_n \vdash C$. This feature in which the logic has a 'conjunction theorem' which says that the ',' in a sequent is represented by a connective is not uniform across substructural logics. There is a detailed classification of this wider family in Restall[36], who would say that $\otimes$ was a *fusion* for ','.

In **BI** both conjunctions can be internalized into the hypotheses, as it contains two distinct punctuation marks: the comma, and the semicolon. The phrases constructed out of these, such as $(A, B); (A, C); (B, (A; C; D), A)$ are known as *bunches*, and their structure can be nested arbitrarily deeply. The structural rules are then permitted for ';' which corresponds to $\wedge$, and not for ',' which corresponds to $*$. This generalization is the key to giving elegant introduction rules for $\rightarrow$ and $-\!\!*$.

## 3.2 Formal Definitions

We now formally define the logic **BI**.

**Definition 11.** *The* formulæ *of **BI** are formally defined by this grammar*

$$\begin{array}{rcl} \mathsf{formula} & := & \mathsf{atomic} \\ & | & \top \\ & | & I \\ & | & \mathsf{formula} * \mathsf{formula} \\ & | & \mathsf{formula} \wedge \mathsf{formula} \\ & | & \mathsf{formula} \rightarrow \mathsf{formula} \\ & | & \mathsf{formula} \mathrel{-\!\!*} \mathsf{formula} \end{array}$$

*We adopt the convention that atomic formulæ are denoted by the letters $A, B, C, \ldots$ whilst general formulæ are $P, Q, R, \ldots$. Acknowledging that both $\wedge$ and $*$ are commutative and associative, we usually omit unnecessary brackets, thus deliberately confusing certain equivalent formulæ.*

Note that $\top$ is the unit for $\wedge$ and $I$ is the unit for $*$.

**Definition 12.** *A* prebunch *of **BI** formulæ is defined by the following grammar:*

$$\begin{array}{rcl} \mathsf{prebunch} & := & \mathsf{formula} \\ & | & \emptyset_a \\ & | & \emptyset_m \\ & | & \mathsf{prebunch}, \mathsf{prebunch} \\ & | & \mathsf{prebunch}; \mathsf{prebunch} \end{array}$$

*A* bunch *is a member of the class formed from prebunches by reducing modulo the smallest equivalence relation which respects commutativity and (separate) associativity of ';' and ',', the unit laws for $\emptyset_a$ w.r.t ';' and $\emptyset_m$ w.r.t ',', and the substitution property that if $\Delta \equiv \Delta'$ then $\Gamma(\Delta) \equiv \Gamma(\Delta')$ (i.e., the least congruence containing the commutativity, associativity, and unit laws). We will generally denote bunches by uppercase greek letters $\Gamma, \Delta, \ldots$.*

*We will later need to use bunches over other entities, not just formulæ, defined analogously.*

### 3.2.1 Proof Theory

We summarise the Proof Theory from Pym [32]. We show the calculus LBI in Fig. 3.1, and the natural deduction calculus NBI in Fig. 3.2. Note the exact symmetry between the additive and multiplicative connectives: the rules for $I, *, \mathrel{-\!\!*}$ are *precisely* the same form as the rules for $\top, \wedge, \rightarrow$, with the substitution of ',' for ';'; of course the more familiar LJ rules for $\top, \wedge, \rightarrow$ are all derivable from those given using the structural rules.

We now state the important theorems about the proof theory.

$$\frac{}{P \vdash P}\,Axiom \qquad \frac{\Gamma \vdash P \quad \Delta(P) \vdash R}{\Delta(\Gamma) \vdash R}\,Cut$$

$$\frac{\Gamma(\Delta) \vdash P}{\Gamma(\Delta;\Xi) \vdash P}\,W \qquad \frac{\Gamma(\Delta;\Delta) \vdash P}{\Gamma(\Delta) \vdash P}\,C$$

$$\frac{\Gamma \vdash P}{\Delta \vdash P}\,\Delta \equiv \Gamma$$

$$\frac{\Gamma(\emptyset_m) \vdash P}{\Gamma(I) \vdash P}\,IL \qquad \frac{}{\emptyset_m \vdash I}\,IR$$

$$\frac{\Gamma(\emptyset_a) \vdash P}{\Gamma(\top) \vdash P}\,\top L \qquad \frac{}{\emptyset_a \vdash \top}\,\top R$$

$$\frac{}{\bot \vdash P}\,\bot L$$

$$\frac{\Gamma \vdash P \quad \Delta(\Xi, Q) \vdash R}{\Delta(\Xi, \Gamma, P \mathrel{-\!\!*} Q) \vdash R}\,\mathrel{-\!\!*}L \qquad \frac{\Gamma, P \vdash Q}{\Gamma \vdash P \mathrel{-\!\!*} Q}\,\mathrel{-\!\!*}R$$

$$\frac{\Gamma(P, Q) \vdash R}{\Gamma(P * Q) \vdash R}\,*L \qquad \frac{\Gamma \vdash P \quad \Delta \vdash Q}{\Gamma, \Delta \vdash P * Q}\,*R$$

$$\frac{\Gamma \vdash P \quad \Delta(\Xi; Q) \vdash R}{\Delta(\Xi; \Gamma; P \to Q) \vdash R}\,\to L \qquad \frac{\Gamma; P \vdash Q}{\Gamma \vdash P \to Q}\,\to R$$

$$\frac{\Gamma(P; Q) \vdash R}{\Gamma(P \wedge Q) \vdash R}\,\wedge L \qquad \frac{\Gamma \vdash P \quad \Delta \vdash Q}{\Gamma; \Delta \vdash P \wedge Q}\,\wedge R$$

$$\frac{\Gamma(P) \vdash R \quad \Gamma(Q) \vdash R}{\Gamma(P \vee Q) \vdash R}\,\vee L \qquad \frac{\Gamma \vdash P_i}{\Gamma \vdash P_1 \vee P_2}\,\vee R \quad (i = 1 \text{ or } 2)$$

Figure 3.1: LBI: A sequent calculus for BI

$$\frac{}{P \vdash P}\,Axiom \qquad \frac{\Gamma \vdash P \quad \Delta(P) \vdash R}{\Delta(\Gamma) \vdash R}\,Cut$$

$$\frac{\Gamma(\Delta) \vdash P}{\Gamma(\Delta;\Xi) \vdash P}\,W \qquad \frac{\Gamma(\Delta;\Delta) \vdash P}{\Gamma(\Delta) \vdash P}\,C$$

$$\frac{\Gamma \vdash P}{\Delta \vdash P}\,\Delta \equiv \Gamma$$

$$\frac{\Delta(\emptyset_m) \vdash P \quad \Gamma \vdash I}{\Delta(\Gamma) \vdash P}\,IE \qquad \frac{}{\emptyset_m \vdash I}\,II$$

$$\frac{\Delta(\emptyset_a) \vdash P \quad \Gamma \vdash \top}{\Delta(\Gamma) \vdash P}\,\top E \qquad \frac{}{\emptyset_a \vdash \top}\,\top I$$

$$\frac{}{\bot \vdash P}\,\bot L$$

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

$$\frac{\Delta \vdash P * Q \quad \Gamma(P,Q) \vdash R}{\Gamma(\Delta) \vdash R}\,*E \qquad \frac{\Gamma \vdash P \quad \Delta \vdash Q}{\Gamma,\Delta \vdash P * Q}\,*I$$

$$\frac{\Gamma \vdash P \to Q \quad \Delta \vdash P}{\Delta;\Gamma \vdash Q}\,\to E \qquad \frac{\Gamma;P \vdash Q}{\Gamma \vdash P \to Q}\,\to I$$

$$\frac{\Delta \vdash P \wedge Q \quad \Gamma(P;Q) \vdash R}{\Gamma(\Delta) \vdash R}\,\wedge E \qquad \frac{\Gamma \vdash P \quad \Delta \vdash Q}{\Gamma;\Delta \vdash P \wedge Q}\,\wedge I$$

$$\frac{\Gamma \vdash P \vee Q \quad \Delta(P) \vdash R \quad \Delta(Q) \vdash R}{\Delta(\Gamma) \vdash R}\,\vee E \qquad \frac{\Gamma \vdash P_i}{\Gamma \vdash P_1 \vee P_2}\,\vee I \quad \text{(i=1 or 2)}$$

Figure 3.2: NBI: a ‘natural deduction’ sequent calculus for **BI**

**Theorem 1** (NBI/LBI equivalence)**.** *NBI and LBI are equivalent theories; a sequent* $\Gamma \vdash P$ *is derivable in one if and only if it is derivable in the other.*

**Theorem 2** (Cut admissibility and elimination)**.** *Cut is an admissible rule in both calculi; however, to any proof of a sequent* $\Gamma \vdash P$ *in either calculus using Cut, there corresponds a proof without Cut.*

Just as for intuitionistic logic, with Gentzen's famous *Haupsatz*[16], the Cut-elimination results corresponds to the existence of a normal form for proofs; in [32] this is presented in terms of the $\alpha\lambda$-calculus, the **BI** analogue to the $\lambda$-calculus.

**Theorem 3** (Deduction Theorem)**.** $\Gamma, P \vdash Q$ *if and only if* $\Gamma \vdash P \mathrel{-\!\!*} Q$*. Furthermore,* $\Gamma; P \vdash Q$ *if and only if* $\Gamma \vdash P \to Q$*.*

*Examples*

We give some simple proofs in **BI**.

**Example 1.** *Two proofs of* $A * (A \mathrel{-\!\!*} B) \vdash B$*. In* ***LBI****:*

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{B \vdash B}}{A, A \mathrel{-\!\!*} B \vdash B}\,{\mathrel{-\!\!*}}L}{A * (A \mathrel{-\!\!*} B) \vdash B}\,{*}L$$

*and* ***NBI****:*

$$\dfrac{\dfrac{}{A * (A \mathrel{-\!\!*} B) \vdash A * (A \mathrel{-\!\!*} B)} \quad \dfrac{\dfrac{}{A \mathrel{-\!\!*} B \vdash A \mathrel{-\!\!*} B} \quad \dfrac{}{A \vdash A}}{A, A \mathrel{-\!\!*} B \vdash B}\,{\mathrel{-\!\!*}}E}{A * (A \mathrel{-\!\!*} B) \vdash B}\,{*}E$$

**Example 2.** *A proof of* $A \mathrel{-\!\!*} (B \wedge C) \vdash A \mathrel{-\!\!*} C$*:*

$$\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{\dfrac{}{C \vdash C}}{B; C \vdash C}\,W}{B \wedge C \vdash C}\,{\wedge}L}{A, A \mathrel{-\!\!*} (B \wedge C) \vdash C}\,{\mathrel{-\!\!*}}E}{A \mathrel{-\!\!*} (B \wedge C) \vdash A \mathrel{-\!\!*} C}\,{\mathrel{-\!\!*}}R$$

**Example 3.** *A proof of* $A * (B \wedge C) \vdash (A * B) \wedge (A * C)$*:*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{\dfrac{}{B \vdash B}}{B; C \vdash B}\,W}{B \wedge C \vdash B}\,{\wedge}L}{A, (B \wedge C) \vdash A * B}\,{*}R \quad \dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{\dfrac{}{C \vdash C}}{B; C \vdash C}\,W}{B \wedge C \vdash C}\,{\wedge}L}{A, (B \wedge C) \vdash A * C}\,{*}R}{A, (B \wedge C) \vdash (A * B) \wedge (A * C)}\,{\wedge}R}{A * (B \wedge C) \vdash (A * B) \wedge (A * C)}\,L$$

Note that the converse to this example does not hold.

**Example 4.**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}}{A \vdash A \vee B} \vee R \quad \dfrac{}{C \vdash C}}{A, C \vdash (A \vee B) * C} * R}{(A * C) \vdash (A \vee B) * C} * L \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash B}}{B \vdash A \vee B} \vee R \quad \dfrac{}{C \vdash C}}{B, C \vdash (A \vee B) * C} * R}{(B * C) \vdash (A \vee B) * C} * L}{(A * C) \vee (B * C) \vdash (A \vee B) * C} \vee L$$

The converse to this example does hold, however.

### 3.2.2 Model Theory

We present two related models for **BI**.

**Definition 13.** *By a* partially ordered commutative monoid $(M, e, \cdot, \sqsubseteq)$*, we mean that the commutative monoid* $(M, e, \cdot)$ *is equipped with a partial order* $\sqsubseteq$ *satisfying the* bifunctoriality *condition:*

$$m \sqsubseteq m', n \sqsubseteq n' \Longrightarrow m \cdot n \sqsubseteq m' \cdot n'$$

We can make a partially ordered commutative monoid into a model for BI by defining a forcing relation $\models$ between elements of the monoid — worlds — and BI propositions. For atomic propositions $A, B, C, \ldots$, we fix any particular relation which satisfies *monotonicity*:

$$n \models A, m \sqsubseteq n \Longrightarrow m \models B$$

Now we extend the forcing relation to all BI propositions by structural induction over formulæ:

- $m \models \top$ always;
- $m \models \bot$ never;
- $m \models P \wedge Q$ iff $m \models P$ and $m \models Q$;
- $m \models P \vee Q$ iff $m \models P$ or $m \models Q$;
- $m \models P \rightarrow Q$ iff for all worlds $n \sqsubseteq m$, if $n \models P$ then $n \models Q$;
- $m \models I$ iff $m \sqsubseteq e$;
- $m \models P * Q$ iff there are worlds $n$, $p$ such that $m \sqsubseteq n \cdot p$ and $n \models P$ and $p \models Q$;

- $m \models P -\!\!* Q$ iff for all $n$ such that $n \models P$, $m \cdot n \models Q$

The intuitive models we discussed in the introductory section were almost of this type; the forcing semantics are of course the Kripke semantics for intuitionistic logic[26] extended with extra rules for $I, *, -\!\!*$; furthermore these rules and only these rules exploit the monoid structure.

We define the obvious notion of semantic entailment with respect to these models; $P \models Q$ iff for all models $m$, if $m \models P$ then $m \models Q$. This notion turns out to be sound for LBI and NBI:

**Theorem 4** (Soundness for partially ordered monoids)**.** *If $P \vdash Q$ in LBI (and thus NBI), then $P \models Q$ in the partially ordered monoid semantics.*

Unfortunately, this system is not complete. The interactions between the multiplicative connectives and the counit $\bot$ prevent completeness; an example is explained in some detail in [32]. We can get a limited completeness result if we exclude $\bot$ from our logic:

**Theorem 5** (A limited completeness result)**.** *If $P \models Q$ in the partially ordered monoid semantics, and $P, Q$ are formulæ of the $\bot$-free fragment of BI, then $P \vdash Q$ is provable in LBI.*

In [32], Pym gives a series of increasingly abstract models for BI, including two models which yield completeness results, one based on topological sheaves and one on Grothendieck sheaves. However, the more recent work of Galmiche, Méry and Pym in[15] gave rise to an apparently small modification to the above model which nonetheless is complete for BI:

**Definition 14.** *A* partially ordered partial commutative monoid*, or PCM, $(M, \cdot, e, \sqsubseteq)$ is a generalization of a partially ordered monoid where the monoidal operation $\cdot$ need only be partially defined. The appropriate associativity conditions are that if $(a \cdot b) \cdot c$ is defined, then $b \cdot c$ must be defined, and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, and the symmetric condition. $e.a = a$ is always defined. The compatibility condition should correspondingly now be read as applying when both compositions are defined.*

*Later we will need the notion of* full submonoid *for a PCM. $M \subseteq M'$ is a full submonoid if it is a submonoid, and for all $a, b \in M$, if $a.b$ is defined in $M'$ then it is defined in $M$. We will abuse this definition by extending it to injections $j : M \hookrightarrow M'$ where $M \not\subseteq M'$.*

We use exactly the same definition of a forcing relation $\models$ as before, and remarkably the partiality of the monoid is enough to make it now sound and complete[15]:

**Theorem 6** (Soundness and Completeness for the PCM model)**.** *A sequent $P \vdash Q$ is derivable in LBI if and only if the semantic entailment $P \models Q$ holds in the PCM semantics.*

One way of understanding the completeness problem is that the original monoid model fails to deal satisfactorily with inconsistency. For example, consider the formula $P \wedge (P \mathrel{-\!\!*} \bot)$. This formula is not a contradiction (unlike $P * P \mathrel{-\!\!*} \bot$, which is), i.e. $P \wedge (P \mathrel{-\!\!*} \bot) \not\models \bot$. However, in the monoidal semantics, if $u \models P \wedge (P \mathrel{-\!\!*} \bot)$ then $u \models P$ and $u \models (P \mathrel{-\!\!*} \bot)$, so by completeness we should have $u \cdot u \models \bot$; but no world forces $\bot$.

The partial monoid semantics gives a solution to this problem: there is no problem if $u \cdot u$ is not defined.

There is an alternative view of the partial monoid semantics, which is to consider it a total monoid, with all undefined compositions composing to some distinguished world $x$. This world has the property that it, and it alone, is an inconsistent world, so $x \models \bot$; this approach is precisely equivalent to the previous one, in the same way that the category of partial functions is equivalent to the a subcategory of the category of functions on pointed sets using the pointed elements to represent undefined.

**Example 5.** *Since the semantics is complete, we can use it to exhibit counterexamples to non-theorems. Consider the non-theorem $A * B \wedge A * C \vdash A * (B \wedge C)$. A counterexample to this is as follows:*

- $M = \{e, a\}$
- $e \cdot e = e$, $e \cdot a = a$, $a \cdot a \uparrow$
- $e \models A, C$, $a \models A, B$

1. $a = e \cdot a$, $e \models A$, $a \models B$, *so* $a \models A * B$
2. $a = e \cdot a$, $e \models C$, $a \models A$, *so* $a \models A * C$
3. *From (1) and (2),* $a \models (A * B) \wedge (A * C)$
4. *But, neither* $a \models B \wedge C$ *nor* $e \models B \wedge C$, *so* $a \not\models A * (B \wedge C)$ *(as $a \cdot e$ is the only possible representation of $a$ as a binary sum).*

The completeness proof in [15] is a constructive one, and it yields considerably more than a basic completeness result. In particular:

**Theorem 7** (Decidability)**.** *There is a decision procedure for Propositional BI.*

**Theorem 8** (Finite Model Property)**.** *Any non-theorem has a finite countermodel.*

# Chapter 4

# Ribbon Proofs

---

In this chapter we describe ribbon proofs, a novel proof system for the logic **BI**. The system is an extension of box proofs to **BI**, coinciding with box proofs on the intuitionistic fragment. We first introduce the system relatively informally as a tool for making proofs, emphasising the links with the semantics of the logic. Then we formalize the system, building up some necessary algebraic concepts first.

Once ribbon proofs are described as a formal system, we can make explicit mappings between them and the **BI** proof system **LBI**, and use these mappings to prove that the system is correct — that is, it generates the same set of theorems as **LBI**.

## 4.1 Introduction

Consider the following theorem of **BI**:

$$(A \wedge B) * C \vdash (A * C) \wedge (B * C)$$

It is a natural theorem to think about when exploring the logic; although $\wedge$ and $*$ do not distribute over each other, they do distribute 'one-way'. The LBI and NBI proofs of this theorem, shown in Fig. 4.1 are straightforward, certainly, but they are surprisingly large, and somehow unintuitive.

On the other hand, it's easy to see the *semantic* proof of the theorem, in terms of the model theory given above. If a world $w$ forces the formula $(A \wedge B) * C$, then there are worlds $u, v$ s.t.

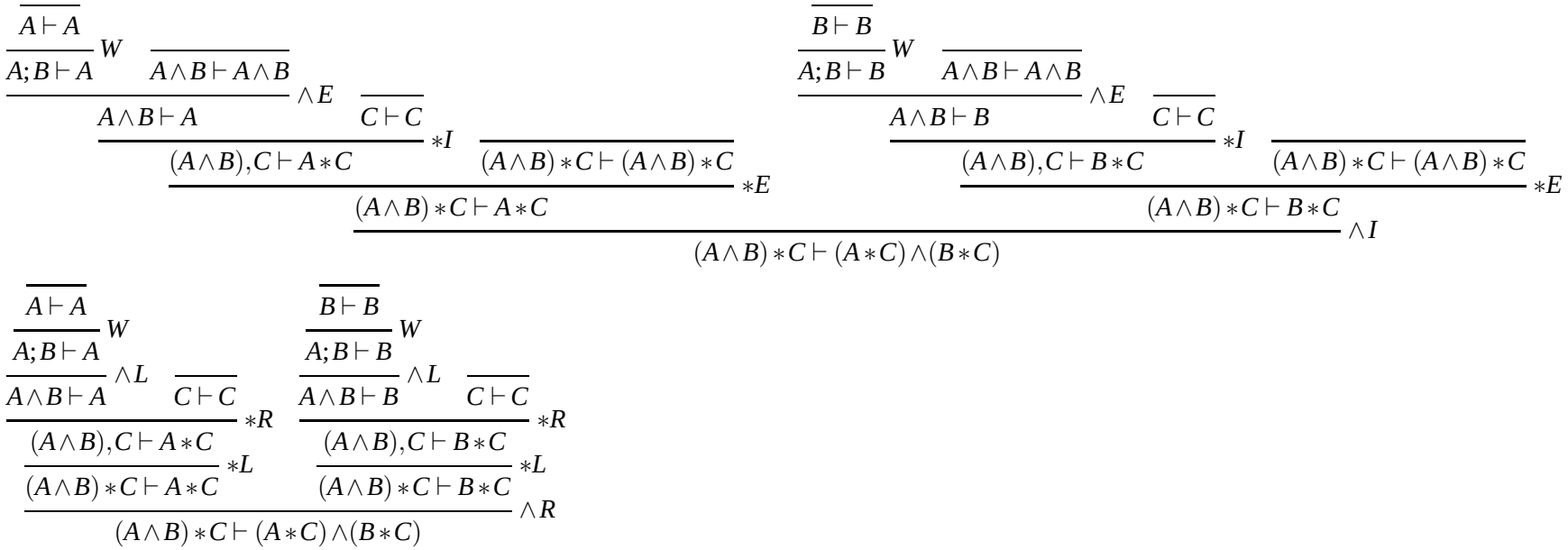


Figure 4.1: LBI/NBI proofs of $(A \wedge B) * C \vdash (A * C) \wedge (B * C)$

$w \sqsubseteq u \cdot v$, $u \models A \wedge B$ and $v \models C$. But then of course $u \models A$, and so $w \models A * C$; similarly $w \models B * C$.

The idea of the ribbon proof is to make the formal proof of this theorem as intuitively direct as the semantic proof. The ribbon proof is shown in Fig. 4.2.

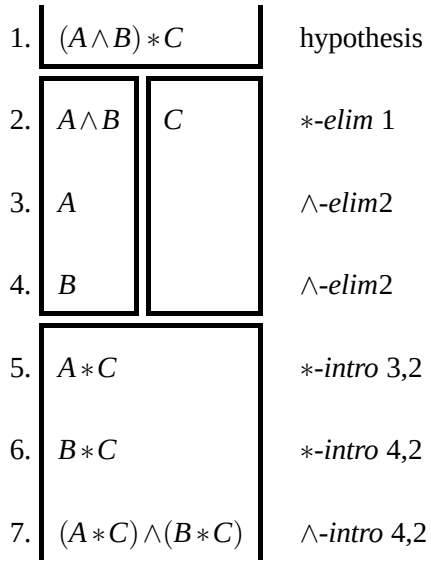



Figure 4.2: A ribbon proof

The heavily lined boxes, which we call ribbons, correspond to worlds of the semantics. The first line is a formula in a single ribbon, and the second line contains two ribbons – we will say that the ribbon has divided into two. Then in the fifth line, the two ribbons combine again, which takes the proof back to the original ribbon; we will say that $A * C$ holds in the same ribbon as $(A \wedge B) * C$.

This is the key to the intuitive reading of ribbon proofs. In a box proof, the informal reading of a formula is 'this formula holds, given the hypotheses', and for formulæ inside boxes 'hypotheses' must be considered to include the temporary assumptions of the box. This loosely corresponds to a truth-value reading of classical or intuitionistic logic. Any similar truth-theoretical reading of BI needs to consider not only whether a formula holds, but *where* it holds, and this is provided by the ribbons.

Ribbon proofs form an extension of box proofs, so all the box proof rules are used in the familiar way. Premisses and conclusion for the box proof rules must all be selected from the same ribbon. When boxes are used, they stretch the entire width of the proof[1], and are drawn as

[1]This is not an essential feature of the system, but a design decision which we will stick to in this paper, as the formalism we present incorporates it

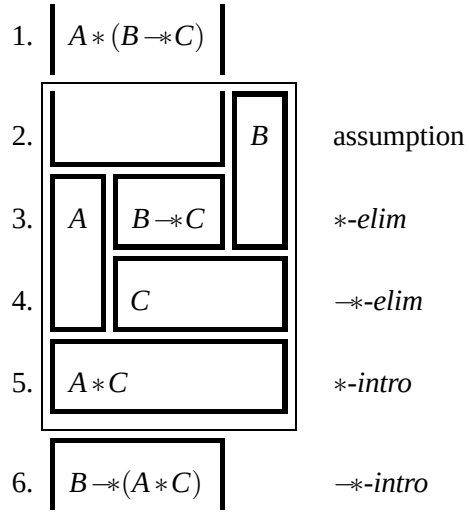



Figure 4.3: $\multimap *$ introduction and elimination

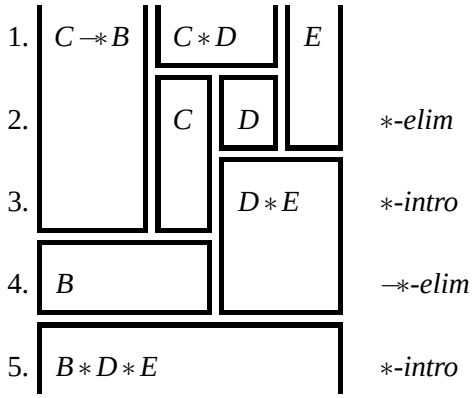



Figure 4.4: Associativity of ribbons

a lighter line to distinguish them from ribbons.

Ribbon proofs add to box proofs introduction and elimination rules for the connectives $I, *, \multimap *$. Figure 4.2 demonstrates both the ribbon proof rules for $*$. $*$-*elim* is a two conclusion rule: it splits a ribbon into two, concluding each conjunct in a new ribbons. $*$-*intro* is graphically symmetrical to $*$-*elim* and combines two ribbons into one. The system we are developing is in fact a multiple-conclusion logic, but quite different from the multiple-conclusion logic used to study natural deduction. In this logic the multiple conclusions are to be understood *conjunctively* (in the sense of $*$) rather than disjunctively. As will be pursued in more detail in later chapters, the diagrams enforce a discipline of 'book-keeping' to ensure than the multiple conlusions are used together correctly.

In Figures 4.3–4.6, we give several further examples of ribbon proofs. Figure 4.3 shows the rules for $\multimap *$. The elimination rule parallels $\rightarrow$ as expected, but rather than taking two premisses

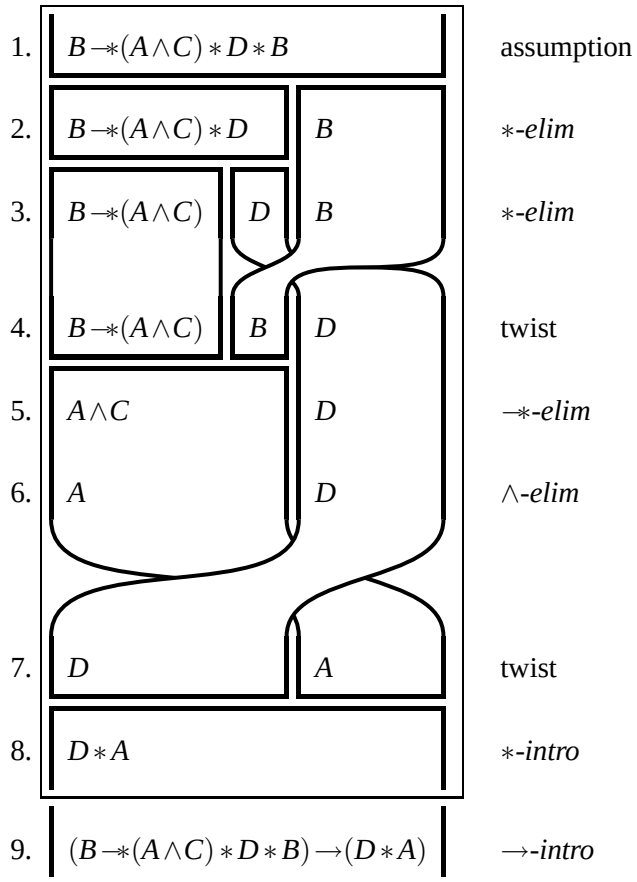



Figure 4.5: The ‘twist’ pseudo-rule

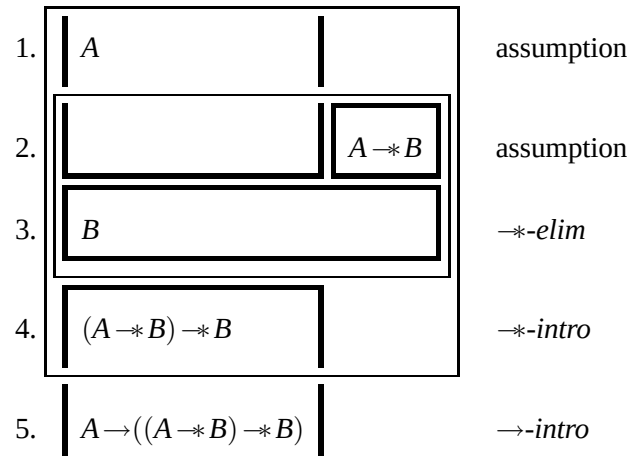



Figure 4.6: $\rightarrow$ and $\multimap$ used together

from the same ribbon, it takes two premisses from different ribbons like $*$-*intro*. The introduction rule uses a box like $\rightarrow$-*intro*, but the assumption is created in a fresh ribbon.

Figure 4.4 demonstrates how the notation for ribbons makes associativity of $*$ automatic. Figure 4.5 demonstrates a pseudo-rule 'twist', which permits horizontal permutation of ribbons; this makes the system respect commutativity for $*$ (it is perhaps better to think of this as commutativity for the bunch-combining comma). Figure 4.6 is an example of combining the two 'kinds' of boxes ($\rightarrow$ and $-\!\!*$) in a single proof.

## 4.2 Ribbon Monoids

We define a particular class of partial commutative monoids, which we will call Ribbon Monoids. This class characterises the monoids formed by the ribbons in ribbon proofs, and will be used in our formalization of ribbon proofs in the next section.

**Definition 15.** *The power set of a finite set $X$ can be made into a PCM, defining, for $a, b \subseteq X$ $a + b = a \cup b$ if $a$ and $b$ are disjoint, undefined if they intersect. Furthermore, such PCMs have a distinguished 'top' element, being the whole set. We will abuse notation and call this the PCM $P(X)$.*

Note also that a non-empty subset of a power set can similarly be considered as a PCM, as long as it closed under disjoint union and complement. (A non-empty set contains an element $a$, so it contains its complement $a^{\mathsf{c}}$ and then $a + a^{\mathsf{c}} =$ the entire set $X$, and $X^{\mathsf{c}} = \emptyset$ so it contains the identity.) Henceforth, we will always be working with these monoids which have distinguished top elements. As a point of notation, we will denote the identity of such a monoid $M$ as $e_M$, and the top element as $\top_M$.

**Definition 16.** *Define* **1** *to be the PCM $P(\{0\})$, the powerset of some singleton set. Define* **2** *to be $P(\{0,1\})$, the powerset of a doubleton viewed as a PCM.*

Now we define two composition operations on these monoids. Given two PCMs $M$ and $N$ with distinguished top elements:

**Definition 17.** *Define the vertical composition, $M \circ N$, as follows. It has as carrier set the quotient of $M \cup N$ under the equivalences $e_M = e_N$, $\top_M = \top_N$. The addition within each monoid is preserved, and for $a \in M$ and $b \in N$, $a + b$ is undefined unless one or the other of $a$ and*

*b is the identity. The definition as a quotient of $M \cup N$ gives rise to canonical injective maps $\iota_M : M \longrightarrow M \circ N$ and $\iota_N : N \longrightarrow M \circ N$.*

**Definition 18.** *Define the horizontal composition, $M \bullet N$ as follows. It has carrier set $M \times N$, and we consider $M$ and $N$ to be contained in $M \bullet N$ by the injections $\iota_M : m \mapsto \langle m, e_N \rangle$ and $\iota_N : n \mapsto \langle e_M, n \rangle$. We define $\langle a, b \rangle + \langle a', b' \rangle = \langle a + a', b + b' \rangle$ if both sums are defined in $M$ and $N$ respectively, noting that this definition preserves addition within the copies of $M$ and $N$ given by the injections. The top element of $M \bullet N$ is the sum of the two top elements of $M$ and $N$, that is $T_{M \bullet N} = \langle \top_M, e_N \rangle + \langle e_M, \top_N \rangle = \langle \top_M, \top_N \rangle$, and the identity is $\langle e_M, e_N \rangle$.*

We will occasionally refer to $M$ and $N$ as *factors* of $M \bullet N$.

**Example 6.** *The PCM* **2** *defined above is* $\mathbf{1} \bullet \mathbf{1}$.

**Example 7.** **1** *is an identity for* $\circ$.

In the exposition that follows, we shall see that all ribbon proofs have an underlying ribbon monoid. The vertical composition operation $\circ$ is the operation on these monoid which corresponds to a vertical composition of ribbon proofs, and the horizontal composition $\bullet$ similarly corresponds to a horizontal composition of proofs. There is a third operation, related to Cut (substitution):

**Definition 19.** *Given two monoids $M$ and $N$, we can define $M$* extended by $N$ at $a \in M$ *as follows. We partition $M$ into three sets: $A = \{b \in M : a + b \downarrow\}$, $B = \{b \in M : a + b \uparrow, b = a + c \text{ for some } c\}$, $C = \{b \in M : a + b \uparrow, b \neq a + c \text{ for any } c\}$. Now we make a monoid on $C \cup (A \times N)$.*

*We define addition as follows. For $c, c' \in C$, addition is as in $M$. $\langle b, n \rangle + \langle b', n' \rangle$ is defined as $\langle b + b', n + n' \rangle$, defined if and only if the sums in $M$ and $N$ are defined. Finally $c + \langle b, n \rangle$ is only defined for $n = e_N$, in which case it is defined as $b + c$ if that sum is defined in $M$.*

This can be seen as an extension of $M$ by the injections $b \mapsto b$ for $b \in C$, $b \mapsto \langle b, e_N \rangle$ for $b \in A$ and $a + c \mapsto \langle c, \top_N \rangle$ for $b = a + c \in B$. There is also an injection from $N$ by $n \mapsto \langle e_M, n \rangle$, Informally, the operation is 'refining' the monoid by adding more elements at $a$, identifying $a$ with $\top_N$. Each such new element (which has the form $\langle e_M, n \rangle$ in our construction) can be combined with every element which could be combined with $a$ (i.e. the elements of $A$), giving rise to the new elements of the form $\langle b, n \rangle$.

Note that $M \circ N$ is $M$ extended by $N$ at the top element, (or **1** extended by $M$ and then $N$ at the top element) and $M \bullet N$ is **2** extended by $M$ and $N$ at the two non-top, non-identity elements.

**Definition 20.** *We define the class of* ribbon monoids *to be the smallest class containing* **1** *and closed under* $\circ$, $\bullet$ *and extension.*

**Proposition 4** (Representation for ribbon monoids)**.** *Every* ribbon monoid *is isomorphic to a PCM arising as the subset of a power set.*

*Proof.* **1** has already been defined as a powerset. It suffices to show that for powerset monoids $M \subset P(X)$ and $N \subset P(Y)$, $M \circ N$, $M \bullet N$ and $M$ extended by $N$ at $a$ are isomorphic to powerset monoids.

For $M \circ N$, we need a powerset to embed $N$ and $M$. Our representations of $m \in M$ and $n \in N$ must not be disjoint for non-identity $m$ and $n$, since $m + n \uparrow \in M \circ N$. We use $P(X \times Y)$. Each element $m \in M$, which is really a set $m \subset X$ goes to $m \times Y \subset X \times Y$; similarly $n \in N \mapsto X \times n \subset X \times Y$. Then certainly $m + n \uparrow$ as required since $(m \times Y) \cap (X \times n) = m \times n$ which is $\emptyset$ iff either $m$ or $n$ is $\emptyset$. The new top element $\top_{M \circ N}$ is $X \times Y$.

For $M \bullet N$, we use $P(X \cup Y)$. $M$ and $N$ are included via the natural injections, and elements $\langle m, n \rangle$ are given by $m \cup n$.

For $M$ extended by $N$ at $a$, we recall that $a$ is a set $a \subset X$ and use the powerset $P((X \setminus a) \cup (a \times Y))$. The injection for $M$ is given by $m \mapsto (m \setminus a) \cup ((m \cap a) \times Y)$, and the injection for $N$ is $n \mapsto a \times n$. □

**Example 8.** *Consider the set M of all even-sized subsets of* $X = \{1,2,3,4,5,6\}$*. This is a PCM, but it is not a ribbon monoid.*

We also note that our ribbon monoids have a natural partial order, which is defined as follows:

**Definition 21.** *For* $r, s$ *elements of a ribbon monoid M, we define* $r \leq s$ *if and only if there is some* $t$ *such that* $s = r + t$.

That this is indeed a partial order follows from the representation theorem; on powerset monoids $\leq$ is $\subseteq$.

## 4.3 Formalising Ribbon Proofs

We present a formalization of ribbon proofs. The formalization will extend the formalization of box proofs in Chapter 2. Again, we work with a notion of an unfinished proof; in this case it will be a *ribbon structure*, extending the notion of box structure. In ribbon proofs, the individual

formulæ are not just arranged in lines, but also in ribbons. These ribbons will be formalized by ribbon monoids as presented in Section 4.2; we will refer to the elements of ribbon monoids simply as ribbons.

**Definition 22.** *We define the following:*

- *A* ribbon structure *is a distinguished* box*;*
- *A* box *is a ribbon monoid, and a sequence of* lines *and* boxes*. The monoid of each internal box is a full submonoid of the ribbon monoid of the enclosing box.*
- *A* line *is a set of triples* $\langle r, f, j \rangle$ *where*
  - *r is a* ribbon*,*
  - *f is a either a formula of (propositional) BI or the special non-formula 'nothing',*
  - *j is a* justification*;*
- *A* ribbon *is an element of the monoid;*
- *A* justification *is one of the special justifications 'hypothesis', 'assumption', or 'nothing', or the name of a ribbon proof rule, together with references to its premisses.*

This notion corresponds with our informal depiction of ribbon proofs. A line is formalization of a line on a page containing some formulæ. The triples are separated from each other by the thick lines which represent ribbons. We draw the $f$ component of the triples in the ribbons, and we display the $j$ component to the right of the proof; which works because the particular ribbon structures we are interested in — ribbon proofs themselves — contain at most one distinct non-*nothing* justification per line.

The most significant contribution of the notation is the way it indicates how one line relates to the next: the thick lines connect triples with the same ribbon component $r$; the $r$-component of the triples, and the structure of the ribbon monoid as a whole, are implicit in the way we join ribbons from one line to the next. The pseudo-rule *twist* allows us to represent graphically the situation when a ribbon we have chosen to draw on one side of the page appears in another line on the other side of the page. The informal notation also establishes some of the additive structure of the ribbon monoid: when we draw two ribbons $s$ and $t$, say, in one line spanning exactly the same horizontal space as one ribbon $r$ in the next, we are denoting the monoid equality $r = s + t$.

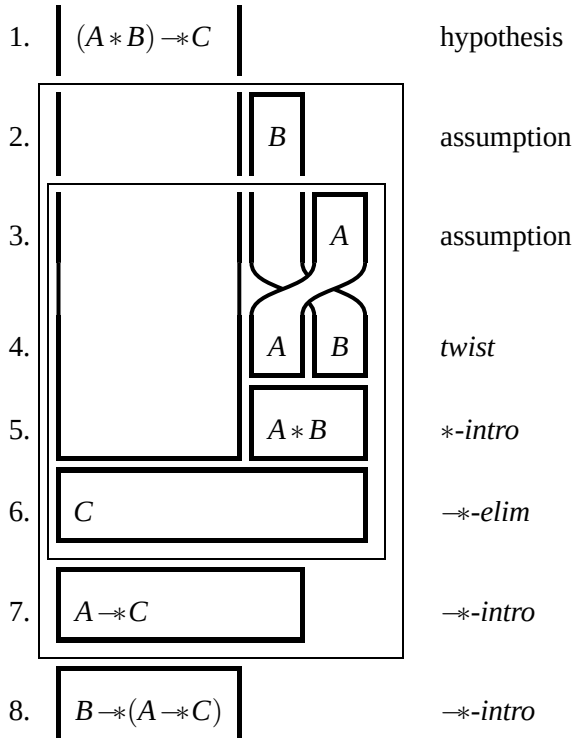



Figure 4.7: A proof using $-\!\!*$ and *twist*

**Example 9.** *As an example, consider the ribbon proof in Fig. 4.7:*

- *the outermost box (the proof itself) has the two-element monoid* $\{e,a\}$ *(i.e.* $P(\{0\})$*),*
- *the first line contains the triple* $\langle a, (A * B) -\!\!* C, \textit{hypothesis}\rangle$*,*
- *the second line begins a box, which has the four element monoid* $\{e,a,b,a+b\}$*,*
- *the first line of this box is* $\{\langle a, \textit{nothing}, \textit{nothing}\rangle, \langle b, B, \textit{assumption}\rangle\}$*,*
- *next there is a further box, with larger monoid* $\{e,a,b,a+b,c,a+c,b+c,a+b+c\}$*,*
- *the first line of this innermost box is* $\{\langle a, \textit{nothing}, \textit{nothing}\rangle, \langle b, \textit{nothing}, \textit{nothing}\rangle, \langle c, A, \textit{assumption}\rangle\}$*,*
- *the* twist *gives rise to the same line again (as our formalism works with sets of triples, not sequences),*
- *the next line of this box is* $\{\langle a, \textit{nothing}, \textit{nothing}\rangle, \langle b+c, A * B, *\textit{-intro}\rangle\}$*,*
- *the final line is* $\{\langle a+b+c, C, -\!\!*\textit{-elim}\rangle\}$*,*
- *returning to the intermediate box we have the line* $\{\langle a+b, A -\!\!* C, -\!\!*\textit{-intro}\rangle\}$*,*

- *and then finallly in the outermost box again the line* $\{\langle a, B \mathrel{-\!\!*} (A \mathrel{-\!\!*} C), \mathrel{-\!\!*}\text{-}intro\rangle\}$.

### 4.3.1 Proper Ribbon Structures

We are not interested in all ribbon structures, but rather ones obeying certain well-formedness conditions. We start with the notion of a ribbon structure *corresponding to a prebunch*:

**Definition 23.** *The ribbon structure* corresponding to a prebunch $\Gamma$, *written* $\mathsf{RS}_\Gamma$ *is defined as follows by induction over the structure of prebunches. We will write the monoid of* $\mathsf{RS}_\Gamma$ *as* $M_\Gamma$, *and the identity and greatest elements in that monoid as* $e_\Gamma$ *and* $1_\Gamma$.

- *The ribbon structure* $\mathsf{RS}_P$ *corresponding to a formula P has a single line, with a single triple* $\langle 1, P, hypothesis\rangle$, *and* $M_P$ *is* **1**.
- $\mathsf{RS}_{\emptyset_a}$, *corresponding to the 'additive empty prebunch' contains no lines, and* $M_{\emptyset_a}$ *is* **1**.
- $\mathsf{RS}_{\emptyset_m}$, *corresponding to the 'multiplicative empty prebunch' is* $\mathsf{RS}_I$.
- $\mathsf{RS}_{\Gamma;\Delta}$ *has the ribbon monoid* $M_{\Gamma;\Delta} = M_\Gamma \circ M_\Delta$. *We form the lines and boxes of* $\mathsf{RS}_{\Gamma;\Delta}$ *by taking the lines* $\mathsf{RS}_\Gamma$ *followed by all the lines of* $RS_\Delta$, *translating all the ribbon-components of the triples using the injections associated with* $\circ$ *(*$\iota_{M_\Gamma}$ *and* $\iota_{M_\Delta}$*).*
- $M_{\Gamma,\Delta} = M_\Gamma \bullet M_\Delta$. $\mathsf{RS}_{\Gamma,\Delta}$ *has all the lines of* $\mathsf{RS}_\Gamma$ 'alongside' *the lines of* $\mathsf{RS}_\Delta$. *Where there are enough lines, this means taking the (automatically disjoint) union of the sets of triples in each line (translating the ribbons using the injections). Where one structure has fewer lines (w.l.o.g.,* $\mathsf{RS}_\Gamma$*), it can be padded with lines of the form* $\{\langle 1_\Gamma, nothing, nothing\rangle\}$.

Note that this definition must be over prebunches; $\Gamma \equiv \Delta$ does not force $\mathsf{RS}_\Gamma = \mathsf{RS}_\Delta$.

Ribbon structures corresponding to bunches allow us to begin our proofs; they tell us how to write down the hypotheses of our proofs. To continue our formalization of proofs we extend these structures by accounting for the rules of ribbon proofs.

We define a set of ribbon structure transformations: rules for generating new ribbon structures from old. The transformations correspond to the application, or partial application, of ribbon proof rules. Most of the transformations have *premisses*, just like the corresponding logical rules. Most of them introduce a new line into a ribbon structure — the *conclusion* of the rule. The conclusion lines are always *based on* some previous line, meaning that the set of ribbons used is computed from the set of ribbons in that line.

Those rules which use boxes — $\rightarrow$*-intro*, $-\!\!*$*-intro* and $\vee$*-elim* — are split into partial applications; one which introduces the box(es), and one which uses it(them).

**Definition 24.** *A* premiss *for a ribbon structure transformation is a triple* $\langle r, f, j\rangle$ *which occurs somewhere in the ribbon structure.*

*The* scope *of a premiss contains all triples* $\langle s,g,k\rangle$ *which occur later in the same box, or within boxes which themselves occur later in this box, such that* $s \geq r$*. Conversely a premiss is* visible *from another triple precisely if that triple is within its scope.*

**Definition 25.** *There are in all eighteen transformations, which we will call* $\wedge$*-intro,* $\wedge$*-elim,* $\vee$*-intro,* $\rightarrow$*-elim,* $\top$*-intro,* $\bot$*-elim,* $*$*-intro,* $-\!\!*$*-elim,* $*$*-elim, I-intro, I-elim,* Box$\rightarrow$*-intro,* Use$\rightarrow$*-intro,* Box$-\!\!*$*-intro,* Use$-\!\!*$*-intro,* Box$\vee$*-elim,* Use$\vee$*-elim and from line.*

1. $\wedge$*-intro,*$\wedge$*-elim,* $\vee$*-intro,* $\rightarrow$*-elim,* $\top$*-intro, from line*

   *These transformations involve adding a single line to a structure. This line must be based on an existing line in the structure; that means it must have exactly the same set of ribbons in its triples. The new line will have only one non-*nothing *formula in it, called the* conclusion*, in ribbon r say. There will be from zero to two* premisses*, also in ribbon r. The premiss lines, and the line the conclusion is based on, must both be either in the same box as the conclusion or in some enclosing box. The triples in these lines are related as shown in the following table:*

| **Rule** | **Conclusion** | **Premisses** |
|---|---|---|
| $\wedge$*-intro* | $\langle r, P\wedge Q, \wedge$*-intro*$\rangle$ | $\langle r, P, j\rangle$ *and* $\langle r, Q, j'\rangle$ |
| $\wedge$*-elim* | $\langle r, P, \wedge$*-elim*$\rangle$ | $\langle r, P\wedge Q, j\rangle$ *or* $\langle r, Q\wedge P, j\rangle$ |
| $\vee$*-intro* | $\langle r, P\vee Q, \vee$*-intro*$\rangle$ | $\langle r, P, j\rangle$ *or* $\langle r, Q, j'\rangle$ |
| $\rightarrow$*-elim* | $\langle r, Q, \rightarrow$*-elim*$\rangle$ | $\langle r, P\rightarrow Q, j\rangle$ *and* $\langle r, P, j'\rangle$ |
| *from line* | $\langle r, P,$ *from line*$\rangle$ | $\langle r, P, j\rangle$ |
| $\top$*-intro* | $\langle r, \top, \top$*-intro*$\rangle$ | *none* |

2. $*$*-intro,* $-\!\!*$*-elim, I-elim*

   *These rules produce a new line with one fewer ribbons than the line it is based on. Some pair r,s of ribbons is replaced with the single ribbon* $r+s$*, containing the conclusion of the rule. The premiss triples may or may not come from the same line: one contains a formula*

*in the ribbon r and the other contains a formula in the ribbon s. (I-elim is not so much a rule in its own right, but a form of $*$-intro recognising the identity between P and $P*I$. It was included into the system out of a desire for symmetry; similar remarks apply to I-intro and $*$-elim). The forms of the premisses and conclusion are:*

| **Rule** | **Conclusion** | **Premisses** |
|---|---|---|
| $*$-*intro* | $\langle r+s, P*Q, *\text{-}intro\rangle$ | $\langle r,P,j\rangle$ *and* $\langle s,Q,j'\rangle$ |
| $-\!*$-*elim* | $\langle r+s, Q, -\!*\text{-}elim\rangle$ | $\langle r,P-\!*Q,j\rangle$ *and* $\langle s,P,j'\rangle$ |
| *I-elim* | $\langle r+s, P, *\text{-}intro\rangle$ | $\langle r,P,j\rangle$ *and* $\langle s,I,j'\rangle$ |

3. $*$-*elim, I-intro*

   *The $*$-elim and I-intro transformations modify the monoid of the ribbon structure. In each case some ribbon r is chosen, and then the monoid is modified by adjoining two fresh elements s and t such that $s+t=r$.*

   *The monoid $\{e,s,t,s+t\}$ is the monoid* **2**. *We alter the ribbon monoid by* extending it by **2** at *r*. *We adjust all the existing lines by replacing the ribbons with those corresponding under the natural injection into the extended monoid. Note that the ribbon structure this produces will not mention ribbons s and t anywhere.*

   *These rules have two conclusions, in a line based on some line which contained the ribbon r, with r replaced by the two new ribbons s and t. The line will take the form shown in this table:*

| **Rule** | **Conclusions** | **Premisses** |
|---|---|---|
| $*$-*elim* | $\langle s,P,*\text{-}elim\rangle$ *and* $\langle t,Q,*\text{-}elim\rangle$ | $\langle s+t,P*Q,j\rangle$ |
| *I-intro* | $\langle s,P,I\text{-}intro\rangle$ *and* $\langle t,I,I\text{-}intro\rangle$ | $\langle s+t,P,j\rangle$ |

4. Box$\rightarrow$*-intro introduces a new box into a ribbon structure. This box goes into an existing box, and inherits the monoid of that box (i.e. the injection between the monoids is an isomorphism). The new box contains a single line, based on some previous line, containing a single non-*nothing *triple, $\langle P,r,assumption\rangle$. The box is said to be focussed on ribbon r with assumption P.*

5. Use$\rightarrow$*-intro uses an existing box (created by* Box$\rightarrow$*-intro) to add a line to the structure. The box should contain only a single assumption P, in a ribbon r, say. The new line,*

*which is to be placed immediately after the box, should be based on the last line of the box, which should contain a formula $Q$ in the same ribbon $r$. The new line contains its only non-*nothing *triple $\langle r, P \to Q, \to\text{-intro}\rangle$.*

6. Box $-\!\!*$*-intro also adds a new box to a ribbon structure. However, in this case the monoid is different. It is the monoid of the enclosing box, with a new element $r$ freely adjoined — i.e. the new monoid is $M \bullet \mathbf{1}$, where $r$ is the top element of the* **1** *factor.*

   *The only line in the new box is based on some previous line, and contains all the ribbons in that line (with nothing in them) plus additionally the triple $\langle r, P, \text{assumption}\rangle$.*

7. Use $-\!\!*$*-intro uses box created by* Box $-\!\!*$*-intro — that is, the box must have a monoid of the form $M \bullet \mathbf{1}$, where $M$ is the monoid of the enclosing box, using $\iota_M$ from the definition of $\bullet$ as the injection, and it must contain a single assumption of the form $\langle r, P, \text{assumption}\rangle$, where $r$ is the ribbon corresponding to the* **1** *factor. A new line is created after the last line of the box, based on it. The last line of the box must contain a triple $\langle s + r, Q, j\rangle$. The new line has the ribbon $s + r$ replaced by $s$, with the triple $\langle s, P -\!\!* Q, -\!\!*\text{-intro}\rangle$.*

8. Box $\vee$*-elim, unlike the other* Box *transformations, has a premiss. This takes the form $\langle r, A \vee B, j\rangle$, and the transformation creates two single-line boxes, both of which must be based on the same line containing $r$, one containing $\langle r, A, \text{assumption}\rangle$ and the other $\langle r, B, \text{assumption}\rangle$.*

9. Use $\vee$*-elim can be used when both boxes have arrived at the same conclusion $\langle s, C, j\rangle$ (note that $s$ need not be $r$) in their last lines. The conclusion is $\langle s, C, \vee\text{-elim}\rangle$.*

**Definition 26.** *A* proper ribbon structure *is a member of the smallest class that contains all ribbon structures which correspond to bunches, and is closed under this set of ribbon structure transformations.*

Given such a proper ribbon structure, we want to be able to retrieve the bunch that it was 'based' on. That this is possible is a form of coherence theorem for the above inductive definition. Unfortunately, we can't derive a unique bunch up to $\equiv$. We need a stronger relation which we will call $\approx$.

**Definition 27.** *The relation $\approx$ on **BI** bunches is the smallest such which includes $\equiv$ and satisfies $\Delta(\emptyset_m) \approx \Delta(I)$.*

Note that since $I \vdash P$ if and only if $\emptyset_m \vdash P$, $\approx$ preserves consequents.

**Lemma 1.** *Given two bunches* $\Gamma$ *and* $\Delta$*, if* $\mathsf{RS}_\Delta = \mathsf{RS}_\Gamma$ *then* $\Gamma \approx \Delta$.

Given this lemma, we will refer to a structure as being based on $\Gamma$, remembering that $\Gamma$ will only be defined up to $\approx$. These proper ribbon structures formalize the notion of a ribbon proof under construction. A complete ribbon proof is simply such a structure which is 'finished':

**Definition 28.** *A* ribbon proof *is a proper ribbon structure in which every box except the outermost has been* used *by the appropriate rule, and whose last line contains only a single ribbon, containing a formula P. It is said to be a proof* of $\Gamma \vdash P$*, where* $\Gamma$ *is the bunch that the structure is based on (up to* $\approx$*).*

### 4.3.2 Soundness and Completeness

We show that ribbon proofs are a sound and complete system by proving their equivalence to LBI, which is known to be sound and complete. We will outline in some detail the proofs of relative soundness, that every theorem provable with ribbon proofs is LBI provable, and relative completeness, that every theorem provable in LBI has a ribbon proof. Since both proofs proceed by cases for each ribbon proof rule, we give only the base cases and a representative selection of the rule cases.

We need some auxiliary concepts.

**Definition 29.** *A* ribbon bunch *is a bunch based on ribbons (elements of the ribbon monoid of a box of a proof) instead of propositions. Given a particular ribbon monoid M, let the set of ribbon bunches over M be denoted* $\mathsf{RB}(M)$*. We define a partial interpretation function* $[-] : \mathsf{RB}(M) \rightharpoonup M$ *into ribbons as follows:*

- $[r] = r$ *for a ribbon r,*
- $[\mathcal{R};\mathcal{S}] = [\mathcal{R}] = [\mathcal{S}]$ *if they are indeed equal, undefined if they are not equal,*
- $[\mathcal{R},\mathcal{S}] = [\mathcal{R}] + [\mathcal{S}]$ *if that addition is defined in M, undefined if not.*

**Definition 30.** *The* visible hypotheses *from a particular triple* $\langle r, P, j\rangle$ *at a particular line of a proof are those* visible *triples* $\langle s, Q, k\rangle$ *such that one of the following holds:*

- *k is assumption or hypothesis, or*

- *k is $*$-elim or I-intro and the corresponding triple $\langle s', R, *\text{-elim}\rangle$ satisfies $s' \not\leq r$.*

What we are trying to do is, for each formula $P$ in the proof, work out which hypotheses it could have been proved from. The delicate part is the inclusion of the $*$-*elim* conclusions: these formulæ are neither assumptions nor hypotheses in the normal sense, but nonetheless they are the only way of formulating a local hypothesis notion like this.

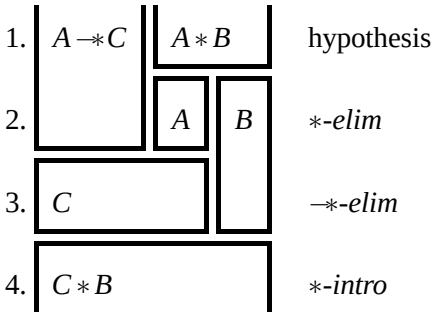



Figure 4.8: Example of visible hypotheses

For example, consider the proof in Figure 4.8. It has two hypotheses $A \mathrel{-\!\!*} C$ and $A * B$, structured as the bunch $A \mathrel{-\!\!*} C, A * B$. Now consider the formula $C$ in line 3: how is it justified? It is not the case that $A \mathrel{-\!\!*} C, A * B \vdash C$. However, using this definition of visible hypothesis, $A * B$ is not visible from $C$, since the $s \leq r$ condition is violated. However, the $*$-*elim* conclusion $A$ *is* a visible hypothesis, and the theorem which justifies $C$ is $A \mathrel{-\!\!*} C, A \vdash C$.

Continuing this example, in the final line of the proof according to the definition $A$ and $B$ are not visible hypotheses: only one of such a pair is ever a hypothesis; once both are visible they are no longer hypotheses in this sense, being superseded by $A * B$ – or rather, its hypotheses, and the final line is justified by the theorem $A \mathrel{-\!\!*} C, A * B \vdash C * B$. Indeed we will say that Figure 4.8 *is a proof of* $A \mathrel{-\!\!*} C, A * B \vdash C * B$. This account is slightly complicated by boxes, but only in an inessential way.

This complex notion of visible hypothesis then is designed to reduce to the notion of hypothesis at the end of a proof, where just a single ribbon of full width remains.

We construct inductively the hypotheses visible at each point in a proof. We start with a related notion: the ribbon bunch at every ribbon. This represents those ribbons which potentially contain hypotheses visible from that point.

**Definition 31.** *For every non-identity ribbon $r$ in a box $b$ of a proper ribbon structure* $\mathsf{RS}$ *we construct a ribbon bunch $\mathcal{R}_{\mathsf{RS}}(r,b)$. For the outermost box we will write simply $\mathcal{R}_{\mathsf{RS}}(r)$. We define it by induction over the construction of proper ribbon structures.*

*Let* $\mathsf{RS}_\Gamma$ *be the ribbon structure corresponding to the prebunch* $\Gamma$*. Let the ribbon monoid of* $\mathsf{RS}_\Gamma$ *be* $M_\Gamma$*. We define* $\mathcal{R}_{\mathsf{RS}_\Gamma}(r)$*:*

- $\Gamma = P$*, a single formula. Then the monoid* $M_\Gamma$ *has a single non-identity element, r say, and* $\mathcal{R}_{\mathsf{RS}_\Gamma}(r) = r$*;*

- $\Gamma = \Delta_1;\Delta_2$*. Then* $M_\Gamma = M_{\Delta_1} \circ M_{\Delta_2}$*. For any ribbon* $r \in M_{\Delta_i}$ *other than the top element* $1_{\Delta_i}$ *we set* $\mathcal{R}_{\mathsf{RS}_\Gamma}(r) = \mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)$ *identifying elements under the standard injection. . For the top element,* $\mathcal{R}_{\mathsf{RS}_\Gamma}(\top_\Gamma) = \mathcal{R}_{\mathsf{RS}_{\Delta_1}}(\top_{\Delta_1});\mathcal{R}_{\mathsf{RS}_{\Delta_2}}(\top_{\Delta_2})$*, identifying elements as before;*

- $\Gamma = \Delta_1, \Delta_2$*. Then* $M_\Gamma = M_{\Delta_1} \bullet M_{\Delta_2}$*. For every ribbon* $r \in M_{\Delta_i}$ *we set* $\mathcal{R}_{\mathsf{RS}_\Gamma}(r) = \mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)$ *identifying elements under the standard injection.*

  *For the new elements* $r+s$ *(*$r \in M_{\Delta_1}$*,* $s \in M_{\Delta_2}$*), we set* $\mathcal{R}_{\mathsf{RS}_\Gamma}(r+s) = \mathcal{R}_{\mathsf{RS}_{\Delta_1}}(r), \mathcal{R}_{\mathsf{RS}_{\Delta_2}}(s)$ *under the identification.*

*For ribbon structures produced by use of the transformations:*

- *Most of the rules —* $\wedge$*-intro,* $\wedge$*-elim,* $\vee$*-intro,* $\to$*-elim,* $\bot$*-elim,* $*$*-intro,* $-\!\!*$*-elim, I-elim,* $\mathsf{Use}-\!\!*$*-intro,* $\mathsf{Box}\to$*-intro,* $\mathsf{Use}\to$*-intro,* $\mathsf{Box}\vee$*-elim,* $\mathsf{Use}\vee$*-elim and from line — do not alter the monoid of the ribbon structure. These transformations leave all the* $\mathcal{R}(r)$ *unchanged. The boxes created by the additive* $\mathsf{Box}$ *rules inherit the monoid of the outer box, and they inherit all the ribbon bunches too;*

- $*$*-elim and I-intro add two fresh elements to the monoid; call them* $r_1$ *and* $r_2$*. We set* $\mathcal{R}(r_i) = r_i$*. The modification of the monoid also generates new elements* $r_i + t$ *for certain t, and we set* $\mathcal{R}(r_i + t) = \mathcal{R}(r_i), \mathcal{R}(t)$*;*

- $\mathsf{Box}-\!\!*$*-intro creates a box, c say, inside a box b; the monoid inside the box is based on the monoid outside the box. For ribbons which existed in b,* $\mathcal{R}(r,c) = \mathcal{R}(r,b)$ *— i.e. the ribbons are identified by the injection of ribbon monoids. The box introduces a new ribbon s, and we set* $\mathcal{R}(s,c) = s$*. For each old element r there is the element* $r+s$*, and we set* $\mathcal{R}(r+s,c) = \mathcal{R}(r,c), \mathcal{R}(s,c)$ *(*$= \mathcal{R}(r,b), s$*).*

**Lemma 2.** *For every triple* $\langle r, P, j\rangle$ *in a ribbon proof,* $\mathcal{R}(r)$ *has the following properties:*

1. $[\mathcal{R}(r)] = r$ ,

2. *$\mathcal{R}(r)$ contains all ribbons which contain hypotheses visible from the triple,*

*Proof.* Both 1 and 2 are proved by induction over the construction of proper ribbon structures. Note that $\mathcal{R}(r)$ is defined even for ribbons $r$ which never 'occur' in the proof: there may be no triples mentioning them.

1. We will prove part 1 for all $r$, not just $r$ occuring in triples, this slightly stronger statement helps the induction go through.

   For the ribbon structure corresponding to a bunch $\Gamma$:

   - $\Gamma = P$ a single formula, there is one ribbon $r$, $\mathcal{R}(r) = r$, and $[r] = r$,
   - $\Gamma = \Delta_1; \Delta_2$

     If $r$ is not the top ribbon, then $\mathcal{R}_{\mathsf{RS}_\Gamma}(r) = \mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)$, and $[\mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)] = r$ by induction, so $[\mathcal{R}_{\mathsf{RS}_\Gamma}(r)] = r$ because the canonical injections are homomorphisms.

     If $r$ is the top ribbon ($\top_\Gamma = \top_{\Delta_1} = \top_{\Delta_2}$ under the injections), then

     $$[\mathcal{R}_{\mathsf{RS}_\Gamma}(\top_\Gamma)] = [\mathcal{R}_{\mathsf{RS}_{\Delta_1}}(\top_{\Delta_1}); \mathcal{R}_{\mathsf{RS}_{\Delta_2}}(\top_{\Delta_2})] = \top_\Gamma$$

     since, by induction, $[\mathcal{R}_{\mathsf{RS}_{\Delta_i}}(\top_{\Delta_i})] = \top_{\Delta_i}$, and $\top_\Gamma = \top_{\Delta_1} = \top_{\Delta_2}$ under the injection.
   - $\Gamma = \Delta_1, \Delta_2$

     For $r \in M_{\Delta_i}$, $[\mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)] = r$ by induction, and $[\mathcal{R}_{\mathsf{RS}_\Gamma}(r)] = r$ because the injections are homomorphic.

     For $r = s_1 + s_2$, $s_i \in M_{\Delta_i}$,

     $$[\mathcal{R}_{\mathsf{RS}_\Gamma}(r)] = [\mathcal{R}_{\mathsf{RS}_{\Delta_1}}(s_1), \mathcal{R}_{\mathsf{RS}_{\Delta_2}}(s_2)] = [\mathcal{R}_{\mathsf{RS}_{\Delta_1}}(s_1)] + [\mathcal{R}_{\mathsf{RS}_{\Delta_2}}(s_2)] = s_1 + s_2 = r$$

     using once again the ribbon homomorphisms.

   For a ribbon structure generated by a transformation, we note, just as in Definition 31, that most of the transformations leave the monoid and the $\mathcal{R}(r)$ untouched, so there is nothing to prove. The ones that don't are *$*-elim*, *I-intro* and *Box$-\!\!*$-intro*.

   *$*-elim*, *I-intro* add two new elements to the monoid. There is an injection from the old monoid to the new, induced by the definition of extension. If a ribbon $r$ was present in the original monoid then $[\mathcal{R}(r)] = r$ by the fact that these injections are homomorphic. If $r$

is one of the two new elements introduced by the extension then $\mathcal{R}(r) = r$ and $[\mathcal{R}(r)] = r$ trivially. If $r = s + t$ is the sum of a new element and an original element, then we have

$$[\mathcal{R}(r)] = [\mathcal{R}(s), \mathcal{R}(t)] = [\mathcal{R}(s)] + [\mathcal{R}(t)] = s + t = r$$

Box $-\!\!*$-*intro* generates a new monoid for the its new box which is formed by adjoining a new element to the original monoid using $\bullet$, and once again there is an injection into the new monoid. For pre-existing $r$ we have again $[\mathcal{R}(r)] = r$ by the homomorphism. For the new element we have simply $\mathcal{R}(r) = r$. For elements of the form $r = s + t$, again:

$$[\mathcal{R}(r)] = [\mathcal{R}(s), \mathcal{R}(t)] = [\mathcal{R}(s)] + [\mathcal{R}(t)] = s + t = r$$

This completes the proof of part 1.

2. Recall that part of the definition of 'visible' entails that, for $\langle s, Q, j' \rangle$ to be visible from $\langle r, P, j \rangle$, $s \leq r$.

   For the ribbon structure $\mathsf{RS}_\Gamma$ corresponding to a prebunch $\Gamma$:

   - $\Gamma = P$ a single formula, there is one ribbon $r$, and the result is automatic,
   - $\Gamma = \Delta_1; \Delta_2$, so $M_\Gamma = M_{\Delta_1} \circ M_{\Delta_2}$

     For $r$ not a top ribbon, the $\circ$ operation adds no new ribbons $\leq r$, so $\mathcal{R}_{\mathsf{RS}_{\Delta_i}}(r)$ suffices.

     For $r$ the top ribbon, there *are* new ribbons $\leq r$ which might contain visible hypotheses. But these hypotheses are either in $\Delta_1$ or $\Delta_2$, and by induction, the ribbons they occur in are represented in $\mathcal{R}_{\mathsf{RS}_{\Delta_i}}$. Therefore they are included in $\mathsf{RS}_\Gamma$ by construction.
   - $\Gamma = \Delta_1, \Delta_2$

     For any ribbon $r$ which lies 'within' $\Delta_1$ or $\Delta_2$, there can be no new hypotheses visible from $r$. For a ribbon which is the sum $r_1 + r_2$ of ribbons from $\Delta_1$ and $\Delta_2$ respectively, the the construction ensures that $s \leq r$ must be $\leq r_1$ or $\leq r_2$. Any $s \leq r_1$ wiill be within $\Delta_1$, and hence be included there, and correspondingly with $r_2$.

   For a ribbon structure produced by transformations, we only have to consider the rules $*$-*elim*, *I-intro*, and Box $-\!\!*$-*intro*, since the others do not modify the ribbon monoid. These three transformations introduce new elements into the monoid which might (indeed, which

definitely do) contain hypotheses. For each such new element $s$ we only need to consider ribbons $r \geq s$. Such $r$ are newly created elements of the form $s+t$, and by definition the ribbon bunches there include $s$ as required.

□

We note that these bunches will in fact contain each ribbon at most once, but since we will not need to use this fact we will not prove it.

**Definition 32.** *The* corresponding sequent *to a formula $P$ in ribbon $r$ of a ribbon proof is a sequent $\Gamma \vdash P$, where $\Gamma$ is a bunch (of BI formulæ) constructed from the ribbon bunch $\mathcal{R}(r)$ of Lemma 2 by replacing each ribbon $s$ with a bunch $\Gamma_s$. $\Gamma_s$ is an additive bunch (i.e. semicolon-separated) containing each hypothesis in $s$ visible from $P$ in $r$. If there are none, then $\Gamma_s = \emptyset_a$.*

The notion of corresponding sequent, althought slightly delicately defined, is just a formalization of the question 'What have we proved at this point?'.

However, the bunches in the sequents may not be exactly the most natural; they may differ by a class of bunches which have no power for provability.

**Definition 33.** *A* generalized empty bunch *is a bunch built up from copies of the additive empty bunch $\emptyset_a$ using ',' and ';'. Formally,*

$$\mathsf{genempty} = \emptyset_a \quad | \quad \mathsf{genempty}, \mathsf{genempty} \quad | \quad \mathsf{genempty}; \mathsf{genempty}$$

*We will use $\emptyset_x, \emptyset_y$ to stand for generalized empty bunches.*

**Lemma 3.** *Generalized empty bunches have no provability power: if $\emptyset_x \vdash P$ and $\emptyset_x$ is a generalized empty bunch then $\emptyset_a \vdash P$. In fact, if $\Delta(\emptyset_x) \vdash P$ and $\emptyset_x$ is a generalized empty bunch then $\Delta(\emptyset_a) \vdash P$*

*Proof.* Let $Q$ be the formula obtained from $\emptyset_x$ by replacing each $\emptyset_a$ with $\top$, each comma with $*$ and each semicolon with $\wedge$.

If $\Delta(\emptyset_x) \vdash P$, then by repeated application of $*L$, $\wedge L$ and $\top L$, we have $\Delta(Q) \vdash P$. It suffices to prove that $\emptyset_a \vdash Q$ for then the result follows by Cut.

$$\dfrac{\begin{array}{l}\Delta(\emptyset_x) \vdash P \\ \quad \vdots\ *L, \wedge L, \top L \\ \Delta(Q) \vdash P\end{array} \qquad \begin{array}{l}\\ \vdots \\ \emptyset_a \vdash Q\end{array}}{\Delta(\emptyset_a) \vdash Q}\ \mathit{Cut}$$

$Q$ is a formula constructed entirely from $\top, *, \wedge$. We prove that $\emptyset_a \vdash Q$ by induction on structure of $Q$.

- The base case $\emptyset_a \vdash \top$ is $\top R$;
- If $Q = Q_1 \wedge Q_2$ then by induction $\emptyset_a \vdash Q_i$ $(i = 1,2)$, and:

$$\dfrac{\dfrac{\emptyset_a \vdash Q_1 \quad \emptyset_a \vdash Q_2}{\emptyset_a;\emptyset_a \vdash Q_1 \wedge Q_2}\wedge R}{\emptyset_a \vdash Q_1 \wedge Q_2}\equiv$$

- If $Q = Q_1 * Q_2$ then by induction $\emptyset_a \vdash Q_i$ $(i = 1,2)$, and:

$$\dfrac{\dfrac{\dfrac{\dfrac{\emptyset_a \vdash Q_1}{\emptyset_a;\emptyset_m \vdash Q_1}W}{\emptyset_m \vdash Q_1}\equiv \quad \emptyset_a \vdash Q_2}{\emptyset_m,\emptyset_a \vdash Q_1 * Q_2}*R}{\emptyset_a \vdash Q_1 * Q_2}\equiv$$

□

Given this lemma, we will add a derived proof rule '*GenEmpty*':

$$\dfrac{\Delta(\emptyset_x) \vdash P}{\Delta(\emptyset_a) \vdash P}\,GenEmpty$$

As a particular case, we will slightly abuse notation by absorbing uses of $\equiv$ into the rule, as in:

$$\dfrac{\Delta;\emptyset_x \vdash P}{\Delta \vdash P}\,GenEmpty$$

**Proposition 5.** *In a ribbon proof of* $\Gamma \vdash P$ *the corresponding sequent to the conclusion* $P$ *as it occurs in the final line of the proof is* $\Gamma \vdash P$.

*Proof.* Note that only hypotheses from the inital bunch can still be visible at $P$ (all boxes must have closed, and both conclusions of any $*$-*elim* will be visible). □

We will prove relative soundness by showing, for every proof rule, that the corresponding sequent at the conclusion can be deduced from the corresponding sequents at the premisses of the rule, and the structure of the proof. To do this, we need to be able to characterise to an extent the syntactic form of these sequents.

**Definition 34.** *Given a bunch* $\Gamma$*, consider the set of* weakenings *of* $\Gamma$*. That is, the set containing* $\Gamma$ *and closed under the transformation* $\Delta(\Xi) \mapsto \Delta(\Xi;\Xi')$*. We will use the notation* $W(\Gamma)$ *to stand*

*for any member of this set. Note that, by repeated application of the rule of weakening, if* $\Gamma \vdash P$ *then* $W(\Gamma) \vdash P$.

*Similarly, given a ribbon bunch* $\mathcal{R}$*, we define the set of* weakenings *of* $\mathcal{R}$*, where any subbunch* $\mathcal{S}$ *can be replaced by* $\mathcal{S};\mathcal{S}'$ *as long as* $[\mathcal{S}'] = [\mathcal{S}]$*; i.e., as long as the new bunch still obeys the conditions of Lemma 2. We will use the notation* $W(\mathcal{R})$ *for any weakening of* $\mathcal{R}$.

**Definition 35.** *Fixing a particular ribbon proof, given a ribbon bunch* $\mathcal{R}$ *over some ribbon monoid in the proof, we define a bunch* $\mathcal{R}^*$ *which is the result of iteratively replacing within* $\mathcal{R}$ *every subbunch of the form* $s_1, s_2$*, where* $s_1$ *and* $s_2$ *are a pair of ribbons introduced by a use of the* $*$*-elim or I-intro rules, with the ribbon bunch* $\mathcal{R}(s_1 + s_2)$.

*We will use the notation* $W^*(\mathcal{R})$ *for this procedure having been applied to some weakening of* $\mathcal{R}$.

**Definition 36.** *For a fixed ribbon proof, given a bunch* $\Gamma$*, we consider the set of containing all weakenings of* $\Gamma$ *and further closed under the operation of replacing subbunches of the form* $S, T$ *with* $\Xi$ *where* $\Xi \vdash S * T$ *is a corresponding sequent in the proof. We will use the notation* $W^*(\Gamma)$ *to refer to a generic member of this set.*

Observe that when constructing a corresponding sequent, if the ribbon bunch has the form $W^*(\mathcal{R})$ then the sequent will be of the form $W^*(\Gamma) \vdash P$, where $\Gamma$ would be the result of constructing a corresponding sequent from $\mathcal{R}$.

**Lemma 4.** *The following hold of corresponding sequents in a ribbon proof:*

1. *If P, Q occur in that order in the same ribbon r in a proof, with P visible from Q, then the corresponding sequents will have the form* $\Gamma; \emptyset_x \vdash P$, $\Gamma; \Delta \vdash Q$.

2. *If P is a formula in ribbon r, Q in s and R in* $r + s$*, with P and Q visible from R, then the corresponding sequents take the form* $\Gamma; \emptyset_x \vdash P$, $\Delta; \emptyset_y \vdash Q$ *and* $W^*(\Gamma, \Delta) \vdash R$.

*Proof.* Again we proceed by induction over the construction of ribbon structures, starting with those corresponding to a bunch $\Gamma$.

1. By induction over the construction of ribbon structures. For ribbon structures corresponding to bunches, this is by induction of the structure of bunches:

   - For $\emptyset_a$, there are no formulæ in the structure.

- For $\emptyset_m$, and for a single formula $P$, there is only one formula in the proof so $P = Q$ and both sequents are the same.
- For a bunch of the form $\Gamma, \Delta$, there are no ribbons in common between the $\Gamma$ and $\Delta$ portions of the structure, so both formulæ must occur together in one or the other, and the result holds by induction on those simpler structures.
- For a bunch of the form $\Gamma; \Delta$, if the ribbon $r$ is not the top ribbon, then it either lies entirely within the portion of the structure corresponding to $\Gamma$, or the portion corresponding to $\Delta$, and we are done by induction.

  It remains to consider the case that the ribbon $r$ is the top ribbon. Each of $P$ and $Q$ lies either within $\Gamma$ or $\Delta$. The two formulæ, being in the same ribbon, share the same ribbon bunch $\mathcal{R}(r)$. The corresponding sequents are constructed in each case from this bunch.

  If both $P$ and $Q$ lie entirely within $\Gamma$ then, by induction, they have bunches of the form $\Gamma'; \emptyset_y \vdash P$, and $\Gamma'; \Delta' \vdash Q$ when viewed as formulæ in $\mathsf{RS}_\Gamma$. Now viewed in $\mathsf{RS}_{\Delta;\Gamma}$ each corresponding bunch has been augmented to, say, $\Xi_1; \Gamma'; \emptyset_y \vdash P$, and $\Xi_2; \Gamma'; \Delta' \vdash Q$, but $\Xi_1 = \Xi_2$ since no hypothesis in $\Delta$ is visible from $\Gamma$ — in fact, the $\Xi_i$ will be generalized empty bunches — so the bunches have the required form.

  The case when $P$ and $Q$ lie entirely within $\Delta$ is almost identical, except that all hypotheses in $\Gamma$ will be visible from $P$ and $Q$ alike, and thus the $\Xi_i$ will now each contain all hypotheses in $\Gamma$ and once again $\Xi_1 = \Xi_2$ and the bunches have the required form.

  The remaining case is that $P$ lies in $\Delta$ and $Q$ in $\Gamma$. Now $\Delta$ is actually of the form $\Delta_0; P; \Delta_1$, and $\Gamma = \Gamma_0; Q; \Gamma_1$, so that the whole bunch is $\Delta_0; P; \Delta_1; \Gamma_0; Q; \Gamma_1$. Now the corresponding sequent for $P$ is of the form $\Delta_0; P; \emptyset_x \vdash P$, and for $Q$ is of the form $\Delta_0; P; \Delta_1; \Gamma_0; Q; \emptyset_y \vdash Q$, and these are as required.

For ribbon structures produced by transformations, most of the transformations do not introduce new hypotheses and hence do not change corresponding sequents. However they do introduce new formulæ and hence introduce new corresponding sequents. Each new formula has in its corresponding sequent the same antecedent as a pre-existing formula in the same ribbon in the previous line, and we are done by induction.

The rules $\mathsf{Box}{\to}$*-intro*, $\mathsf{Box}{\vee}$*-intro* and $\mathsf{Box}{-\!\!*}$*-intro* do introduce new hypotheses and

need separate treatment. In each case the condition that $P$ be visible from $Q$ requires that $Q$ must be the new hypothesis. In the case of the additive connectives, the corresponding sequent at $Q$ is then of the form $\Gamma; Q \vdash Q$ where $\Gamma$ is the antecedent of any previous formulae in that ribbon at that point in the proof, which is the required form by induction. In the case of Box $-\!\!*$*-intro*, both $P$ and $Q$ must be the new hypothesis, since no other formula is visible from it, and both corresponding sequents are simply $P \vdash P$.

2. We first prove that we will always have $\mathcal{R}(r+s) = W^*(\mathcal{R}(r), \mathcal{R}(s))$.

   For ribbon structures corresponding to bunches, this is proved by induction over the construction of bunches: in the base case there are no such (non-trival) sums of ribbons.

   Construction using ';' yields no new sums, but can modify the ribbon bunch of the top ribbon to include a weakening.

   Construction using ',' by definition produces $\mathcal{R}(r+s) = \mathcal{R}(r), \mathcal{R}(s)$ for all new sums and leaves old sums untouched.

   For ribbon structures produced by transformations, the rules — $\wedge$*-intro*, $\wedge$*-elim*, $\vee$*-intro*, $\rightarrow$*-elim*, $\bot$*-elim*, $*$*-intro*, $-\!\!*$*-elim*, *I-elim*, Use $-\!\!*$*-intro*, Box $\rightarrow$*-intro*, Use $\rightarrow$*-intro*, Box $\vee$*-elim*, Use $\vee$*-elim* and *from line* do not alter the monoid and produce no new sums.

   $*$*-elim* and *I-intro* add a pair of new elements, and several new sums to consider. Let $s_1, s_2$ be the new elements, and note that $\mathcal{R}(s_i) = s_i$. For sums of the form $r + s_i$ with $r$ a pre-existing element, the definition sets $\mathcal{R}(r+s_i) = \mathcal{R}(r), s_i$ and the result holds. For sums of the form $(r+s_i)+r'$ with $r, r'$ pre-existing, we have:

$$\mathcal{R}(r+s_i+r') = \mathcal{R}(r+r'+s_i)$$

   This is the case we just dealt with, so

$$= \mathcal{R}(r+r'), s_i$$

   Now, $r+r'$ was a ribbon which existed before this rule use, and its ribbon bunch is unchanged. There we invoke induction and:

$$= W^*(\mathcal{R}(r), \mathcal{R}(r')), s_i = W^*(\mathcal{R}(r), s_i, \mathcal{R}(r'))$$

   as required.

The final kind of sum is one of the form $(r+s_1)+(r'+s_2)$:

$$\mathcal{R}(r+s_1+r'+s_2) = \mathcal{R}(r+r'+s_1+s_2)$$

But $r+r'$ and $s_1+s_2$ are both preexisting ribbons, their ribbon bunches have not changed, and using the inductive hypothesis once on this sum, and then once again on the sum $r+r'$ we have:

$$= W^*(\mathcal{R}(r+r'),\mathcal{R}(s_1+s_2)) = W^*(\mathcal{R}(r),\mathcal{R}(r'),\mathcal{R}(s_1+s_2))$$

Now using the definition of $W^*$ we can replace a copy of $\mathcal{R}(s_1+s_2)$ with $s_1,s_2$

$$= W^*(\mathcal{R}(r),\mathcal{R}(r'),s_1,s_2)$$

We already remarked that $\mathcal{R}(r+s_1) = \mathcal{R}(r),s_1$ so

$$= W^*(\mathcal{R}(r+s_1),\mathcal{R}(r'+s_2))$$

Box $-\!\!*$-*intro* is a slightly simpler case. The monoid inside the new box has one additional element, $s$, say. We need to consider new sums $r+s$ and $(r+s)+t$, which are precisely the same as two of the cases for $*$-*elim*.

This completes the proof that $\mathcal{R}(r+s)$ is of the form $W^*(\mathcal{R}(r),\mathcal{R}(s))$.

Therefore we conclude that given corresponding sequents for $P$ and $Q$ of the form $\Delta;\emptyset_x \vdash P$ and $\Delta;\emptyset_y \vdash Q$, (being based on the ribbon bunches $\mathcal{R}(r)$ and $\mathcal{R}(s)$), the corresponding sequent for $R$ will be of the form $W^*(\Gamma,\Delta)$, as required.

□

Now we are in a position to move on to our main results.

**Theorem 9** (Relative Soundness)**.** *If there is a ribbon proof of a sequent $\Gamma \vdash P$, then it is a theorem of LBI.*

*Proof.* We prove the stronger statement that in a given ribbon proof, every corresponding sequent is a theorem of LBI.

We fix a particular ribbon proof $\Gamma \vdash P$, and we work through the proof line by line, proving for each triple that the corresponding sequent is an LBI-theorem. By induction, we assume that all corresponding sequents for triples in previous lines are LBI-theorems. Our proof goes into boxes as they occur in the proof.

The base step concerns hypotheses. The corresponding sequent to a hypothesis is a (generalized) axiom sequent $\Gamma; P \vdash P$.

There is an inductive step for each of the ribbon proof rules. We will prove here a representative selection. For each case, we show in Fig. 4.9 a general ribbon proof using the rule, annotated with the corresponding sequents at the important points, and show the LBI proof that the sequent corresponding to the conclusion follows from the other sequents.

The proof is quite formulaic, and we display the key ideas in a table. For each rule, we indicate the form of the corresponding sequent(s) for the premiss triple(s), the form of the corresponding sequents for the conclusion triple, and the LBI rules needed to make the proof. For most of the cases, an illustration of the ribbon proofs, together with the required LBI proofs, are shown in Figs. 4.9– 4.10. In each case the fact that the corresponding sequents do indeed take the form shown is a consequence of the various parts of Lemma 4. The strange entry for $*$-*elim* merits a comment. The 'rule' corresponding to $*$-*elim* is of course $*L$, but the nature of the system is that the $*L$ rules are applied not when $*$-*elim* is applied, but later in the proof, when the ribbons are eventually recombined. In fact, the uses of $*L$ are all concealed within the technical notion of $W^*$ defined above.

| **Rule** | **Premisses** | **Conclusion** | **LBI rules** |
|---|---|---|---|
| $\wedge$-*intro* | $\Gamma_0; \emptyset_x \vdash P$, $\Gamma_0; \Gamma_1; \emptyset_y \vdash Q$ | $\Gamma_0; \Gamma_1; \Gamma_2 \vdash P \wedge Q$ | $\wedge R$, structurals |
| $\wedge$-*elim* | $\Gamma_0; \emptyset_x \vdash P_1 \wedge P_2$ | $\Gamma_0; \Gamma_1 \vdash P_i$ | $\wedge L$, Cut, structurals |
| $\vee$-*intro* | $\Gamma_0; \emptyset_x \vdash P$ | $\Gamma_0; \Gamma_1 \vdash P \vee Q$ | $\vee R$, structurals |
| $\vee$-*elim* | $\Gamma_0; \emptyset_x \vdash P \vee Q$, $\Gamma_0; \Gamma_1; P \vdash R$, $\Gamma_0; \Gamma_1; Q \vdash R$ | $\Gamma_0; \Gamma_1 \vdash R$ | $\vee L$, structurals |
| $\rightarrow$-*intro* | $\Gamma_0; P \vdash R$ | $\Gamma_0 \vdash P \rightarrow R$ | $\rightarrow R$ |
| $\rightarrow$-*elim* | $\Gamma_0; \emptyset_x \vdash P \rightarrow Q$,$\Gamma_0; \Gamma_1; \emptyset_y \vdash P$ | $\Gamma_0; \Gamma_1; \Gamma_2 \vdash Q$ | $\rightarrow L$, Cut, structurals |
| $*$-*intro* | $\Gamma_0 \vdash P$,$\Gamma_1 \vdash Q$ | $W^*(\Gamma_0, \Gamma_1) \vdash P * Q$ | $*R$, weakening |
| $*$-*elim* | $\Gamma_0 \vdash P * Q$ | $P \vdash P$,$Q \vdash Q$ | axioms |
| $-\!\!*$-*intro* | $\Gamma_0, P \vdash Q$ | $\Gamma_0 \vdash P -\!\!* Q$ | $-\!\!* R$ |
| $-\!\!*$-*elim* | $\Gamma_0 \vdash P$, $\Gamma_1 \vdash P -\!\!* Q$ | $W^*(\Gamma_0, \Gamma_1) \vdash Q$ | $-\!\!* L$, weakening |
| *I-intro* | $\Gamma_0 \vdash P$ | $P \vdash P$, $\emptyset_m \vdash I$ | axioms, $IR$ |
| *I-elim* | $\Gamma_0 \vdash P$, $\Gamma_1 \vdash I$ | $W^*(\Gamma_0, \Gamma_1) \vdash P$ | $IL$, Cut |

□

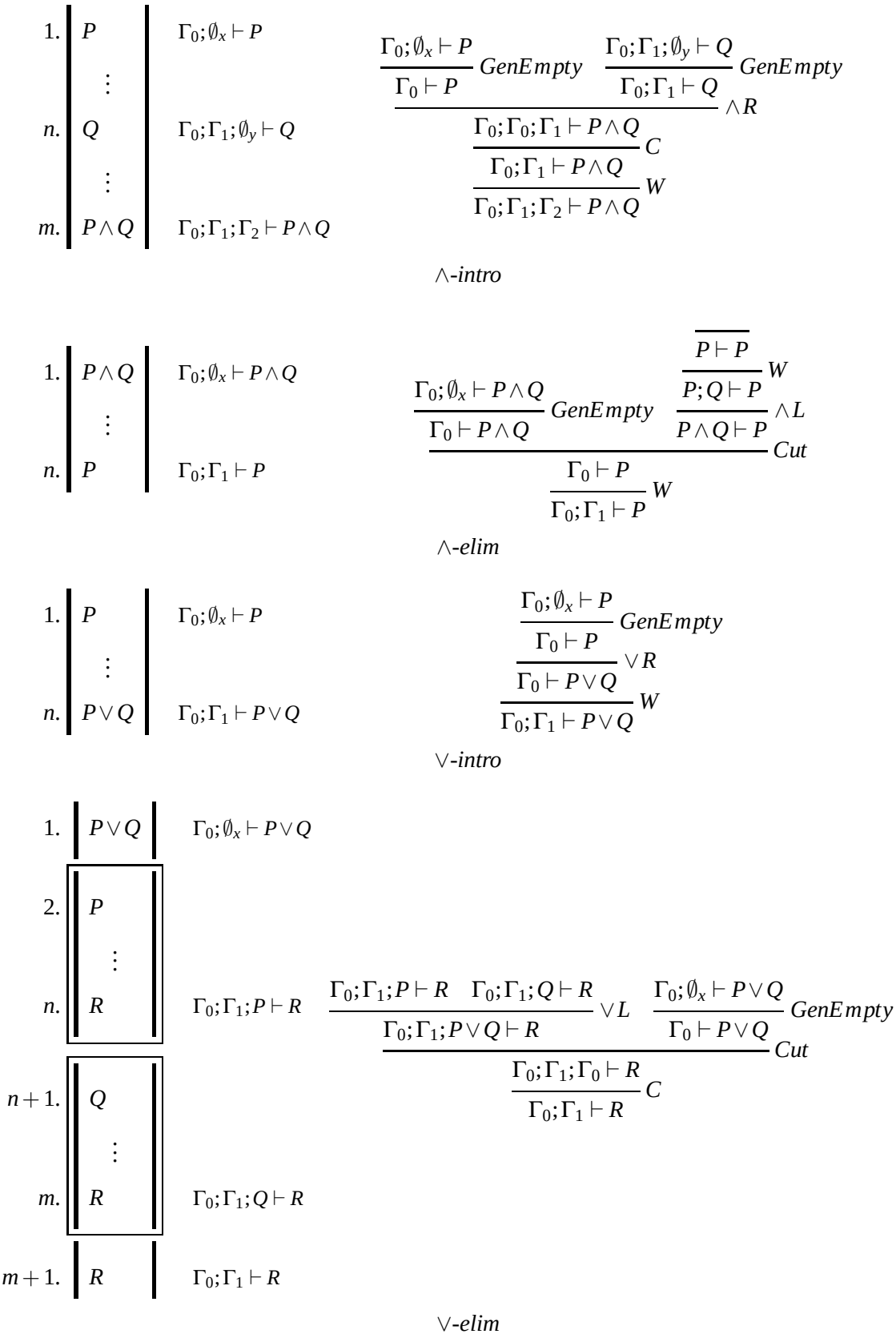
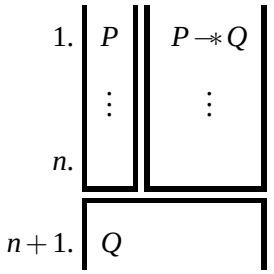


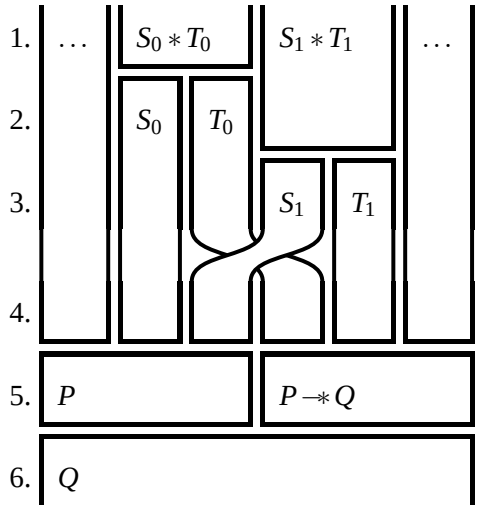

Figure 4.9: Some cases of relative soundness




$\Gamma_0 \vdash P \quad \Gamma_1 \vdash P \mathrel{-\!\!*} Q \quad ((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3));\Gamma_4 \vdash Q$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_0 \vdash P}{\Gamma_0;\Gamma_2 \vdash P}W \quad \dfrac{Q \vdash Q}{}}{(\Gamma_0;\Gamma_2), P \mathrel{-\!\!*} Q \vdash Q}{-\!\!*}L \quad \dfrac{\Gamma_1 \vdash P \mathrel{-\!\!*} Q}{\Gamma_1;\Gamma_3 \vdash P \mathrel{-\!\!*} Q}W}{((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3)) \vdash Q}\mathit{Cut}}{((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3));\Gamma_4 \vdash Q}W$$

$-\!\!*$-*elim* - simplest case




$\Sigma_0 \vdash S_0 * T_0 \quad \Sigma_1 \vdash S_1 * T_1$

$S_0 \vdash S_0 \quad T_0 \vdash T_0$

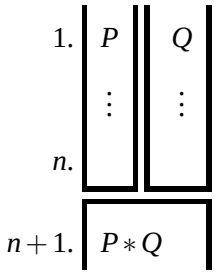
$S_1 \vdash S_1 \quad T_1 \vdash T_1$

$\Gamma_0, S_0, S_1 \vdash P \quad \Gamma_1, T_0, T_1 \vdash P \mathrel{-\!\!*} Q$

$\Gamma_0, \Sigma_0, \Sigma_1, \Gamma_1 \vdash Q$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma_0, S_0, S_1 \vdash P \quad Q \vdash Q}{\Gamma_0, S_0, S_1, P \mathrel{-\!\!*} Q \vdash Q} \quad \Gamma_1, T_0, T_1 \vdash P \mathrel{-\!\!*} Q}{\Gamma_0, S_0, T_0, S_1, T_1, \Gamma_1 \vdash Q}\ \text{Cut (and } \equiv)}{\Gamma_0, S_0, T_0, S_1 * T_1, \Gamma_1 \vdash Q} *L}{\Gamma_0, S_0 * T_0, S_1 * T_1, \Gamma_1 \vdash Q} *L \quad \Sigma_0 \vdash S_0 * T_0}{\Gamma_0, \Sigma_0, S_1 * T_1, \Gamma_1 \vdash Q}\mathit{Cut} \quad \Sigma_1 \vdash S_1 * T_1}{\Gamma_0, \Sigma_0, \Sigma_1, \Gamma_1 \vdash Q}\mathit{Cut}$$

$-\!\!*$-*elim* - with 2 pairs of $*$-*elim* formulæ involved




$\Gamma_0 \vdash P \quad \Gamma_1 \vdash Q \quad ((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3));\Gamma_4 \vdash P * Q$

$$\dfrac{\dfrac{\dfrac{\Gamma_0 \vdash P}{\Gamma_0;\Gamma_2 \vdash P}W \quad \dfrac{\Gamma_1 \vdash Q}{\Gamma_1;\Gamma_3 \vdash Q}W}{((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3)) \vdash P * Q}*R}{((\Gamma_0;\Gamma_2),(\Gamma_1;\Gamma_3));\Gamma_4 \vdash P * Q}W$$

$*$-*intro*

Figure 4.10: Some more cases of relative soundness

**Theorem 10** (Relative Completeness)**.** *For every theorem* $\Gamma \vdash P$ *of LBI, there is a ribbon proof with a single ribbon containing the formula P as its last line, such that the sequent at P is* $\Gamma \vdash P$.

*Sketch.* We prove this by induction over the rules used in the LBI proof of $\Gamma \vdash P$, showing that there is a ribbon proof of every sequent occuring in the proof.

The base case is again the axiom sequent $P \vdash P$. The ribbon proof of that is two lines, containing the formula $P$ once as a hypothesis, and once as the conclusion of the *from line* rule.

There is an induction step for each of the LBI rules. Again, we prove here only a few cases. Every case is a straightforward transformation on proofs. Each case discussed is illustrated in Figs. 4.11– 4.13.

- $\wedge L$ : By induction, we have a ribbon proof $\Gamma; A; B \vdash P$. We transform it by adding a hypothesis $A \wedge B$ above the hypotheses $A$ and $B$. Then we change the justifications of $A$ and $B$ from *hypothesis* to $\wedge$-*elim*, with the premiss the newly added hypothesis, and leave the rest of the proof the same.

- $*R$ : By induction we have ribbon proofs of $\Gamma_0 \vdash P$ and $\Gamma_1 \vdash Q$, and we place them side-by-side and add a final $*$-*intro* step. The formal definition of placing side-by-side is analagous to the notion used to construct the ribbon structure corresponding to a bunch $(\Delta, \Gamma)$.

- $*L$ : We have by induction a ribbon proof of $\Gamma, A, B \vdash P$. We write this proof such that $A$ and $B$ occur as horizontally adjacent hypotheses (may require use of *twist*), and we transform it by placing immediately above $A$ and $B$ the new hypothesis $A * B$. We then alter $A$ and $B$ to no longer be hypotheses, but instead derived from $A * B$ by $*$-*elim*, and leave the rest of the proof intact.

- $-\!\!*R$ : By induction we have a ribbon proof of $\Gamma, P \vdash Q$. We construct a proof of $\Gamma \vdash P -\!\!* Q$ using $-\!\!*$-*intro* as the final step, and inserting the given proof inside the box produced by the $-\!\!*$-*intro* rule.

- $\equiv$ : Suppose we have a ribbon proof of $\Gamma \vdash P$ and we wish to convert it into a ribbon proof of $\Delta \vdash P$ where $\Delta \equiv \Gamma$. Consider the different aspects of $\equiv$. A commutation around a ';' simply changes the order of some hypotheses without altering their ribbons, and won't affect the proof. A commutation around a ',' will make no different at all, since our formalization is in fact commutative for ',' (because we deal with sets of triples not sequences). Addition

or removal of $\emptyset_a$ is totally invisible to ribbon proofs. The most problematic is the unit laws for $\emptyset_m$, since $\emptyset_m$ is represented in ribbon proofs by $I$. However the original structure can be restored in each case by applications of *I-intro* and *I-elim*.

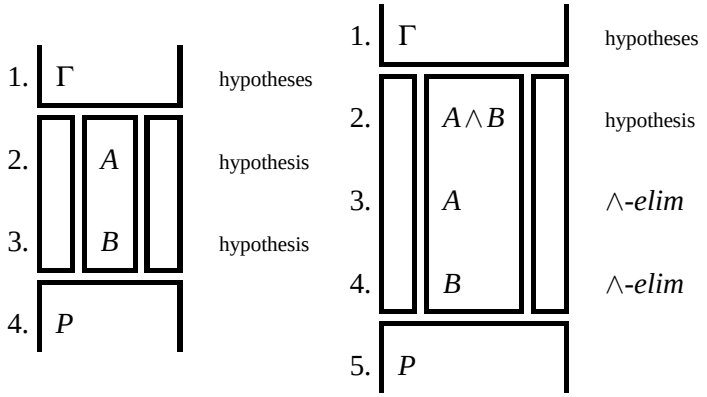
□

So, ribbon proofs really do form a proof system for BI. Ribbon proofs incorporate ribbon monoids in a way which appears to be modelling a fragment of BI. Noting this fact, it is tempting to make the following conjecture:

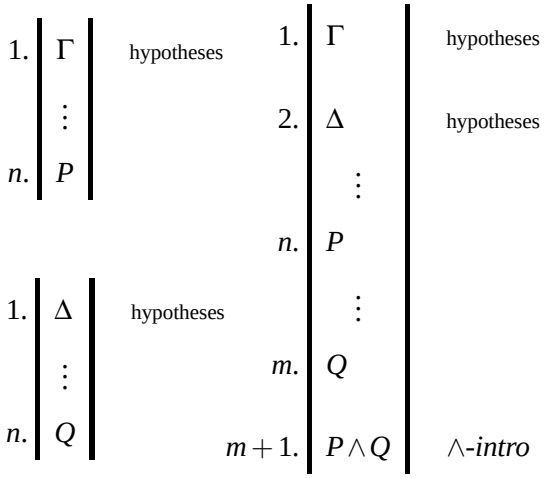
**Conjecture 1.** *BI is complete not only for PCMs, but for the smaller class of ribbon monoids.*

This however does not hold. For example, consider is the non-theorem $(A,B);I \vdash (A \wedge I) * (B \wedge I)$. The counterexample to this non-theorem must contain non-identity worlds $a$, $b$ such that $a+b=e$. This is not possible in ribbon monoids.

$$\frac{\Gamma(A;B)\vdash P}{\Gamma(A\wedge B)\vdash P}\wedge L$$




$$\frac{\Gamma\vdash P \quad \Delta\vdash Q}{\Gamma;\Delta\vdash P\wedge Q}\wedge R$$




$$\frac{\Gamma(P)\vdash R \quad \Gamma(Q)\vdash R}{\Gamma(P\vee Q)\vdash R}\vee L$$

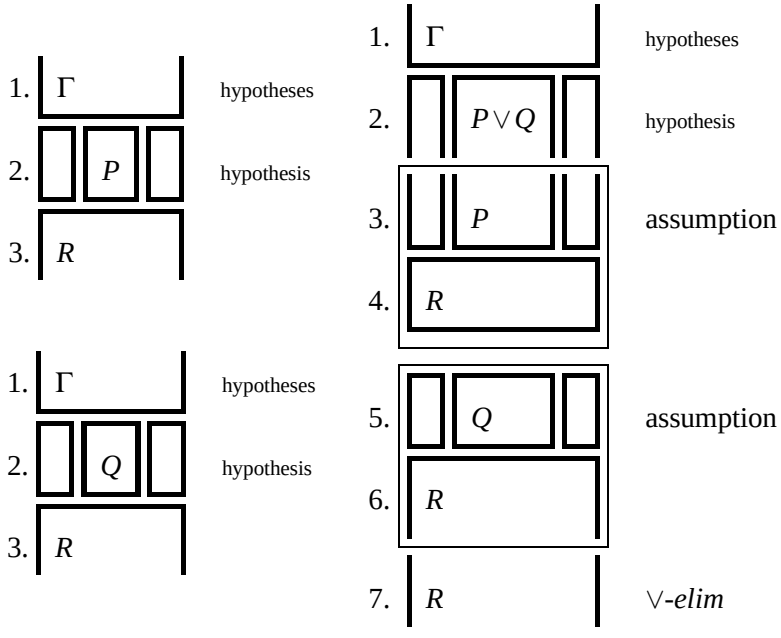



Figure 4.11: Some cases of relative completeness

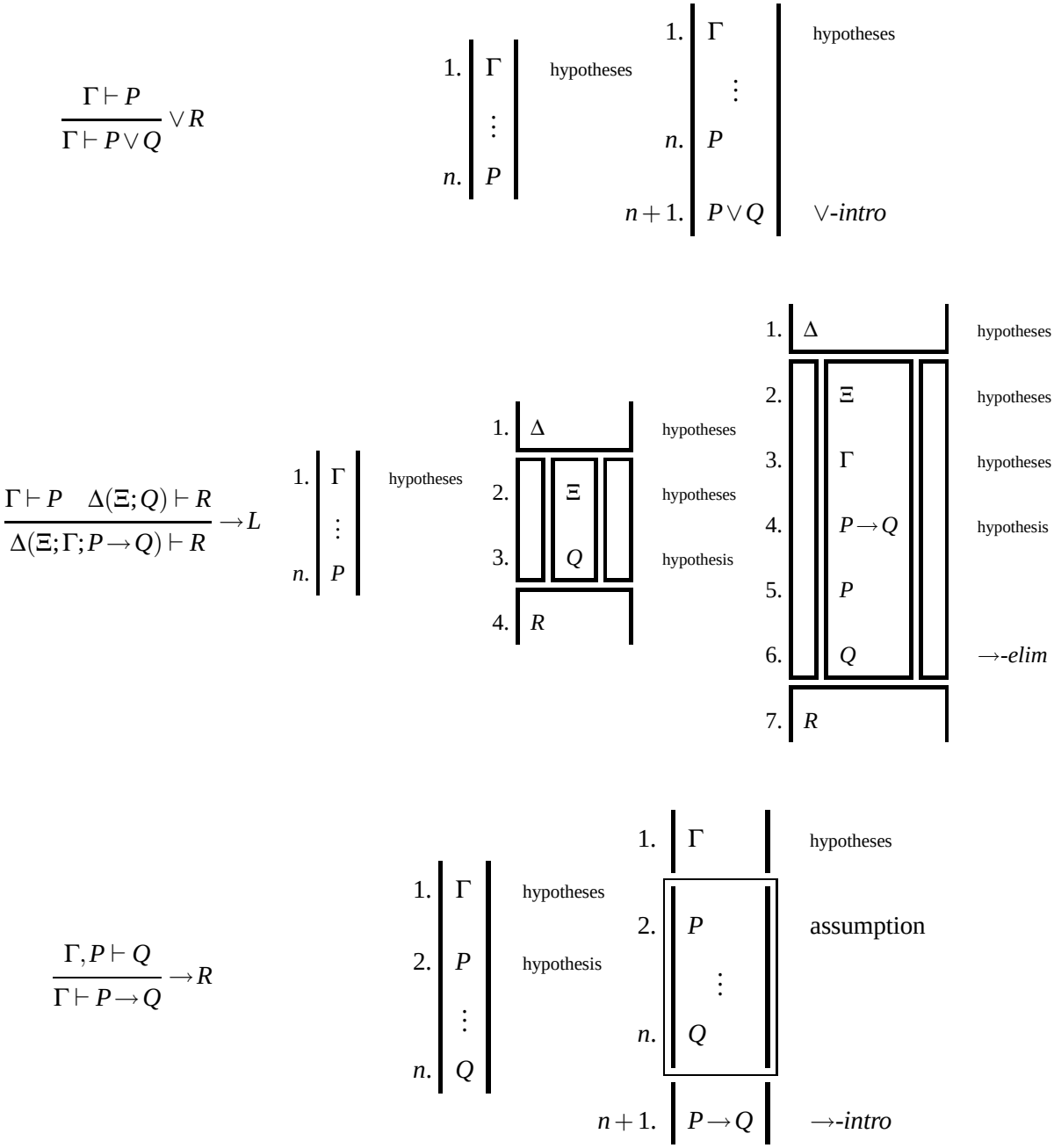
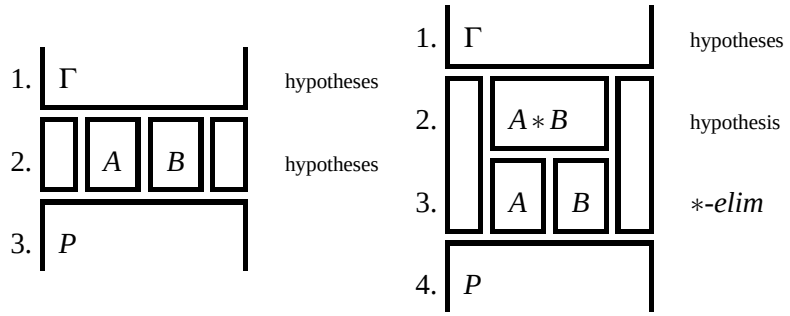
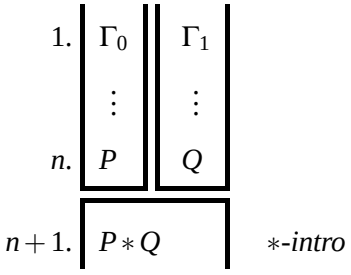



Figure 4.12: Some cases of relative completeness

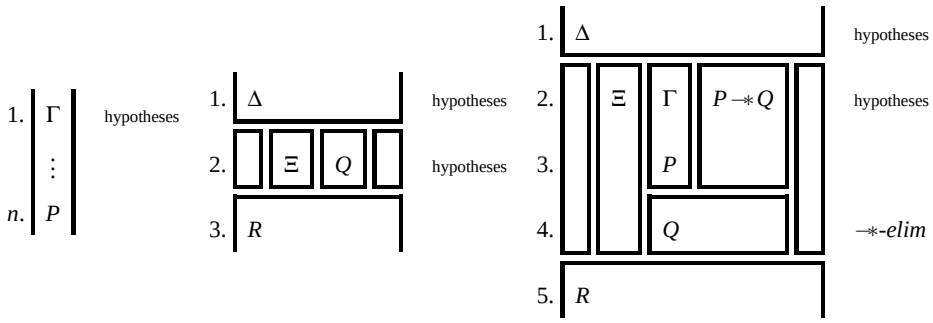
$$\frac{\Gamma(A,B)\vdash P}{\Gamma(A*B)\vdash P}\,{*L}$$




$$\frac{\Gamma_0\vdash P \quad \Gamma_1\vdash Q}{\Gamma_0,\Gamma_1\vdash P*Q}\,{*R}$$




$$\frac{\Gamma\vdash P \quad \Delta(\Xi,Q)\vdash R}{\Delta(\Xi,\Gamma,P\mathrel{-\!\!*}Q)\vdash R}\,{\mathrel{-\!\!*}L}$$




$$\frac{\Gamma,P\vdash Q}{\Gamma\vdash P\mathrel{-\!\!*}Q}\,{\mathrel{-\!\!*}R}$$

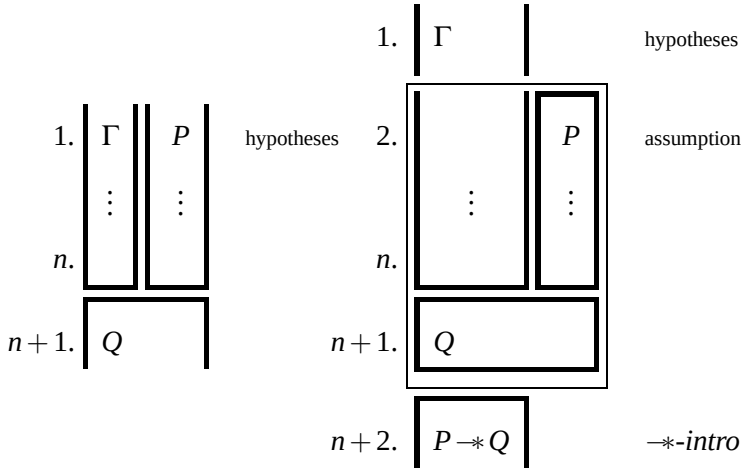



Figure 4.13: Some cases of relative completeness

# Chapter 5

# Ribbon Proof Theory

---

In Chapter 2 we discussed translations between box proofs and natural deduction, as well as between box proofs and the sequent calculus. This gave us a setting in which we were able to discuss normalization and box proofs.

In the same spirit, we want to consider normalization for Ribbon Proofs. Recall that there is a normalization theorem for **NBI** via the $\alpha\lambda$-calculus, and equivalently a cut-elimination theorem for **LBI**.

Since there is no direct analogue of **NJ** for **BI**, the translations we can consider are between ribbon proofs and **NBI** or **LBI**. Our soundness and completeness results essentially construct such translations.

## 5.1 Substitution

One of the properties we expect from a formal proof system is some kind of substitution property; if we can prove something using a hypothesis $P$, and we have another proof of $P$ from hypothesis $Q$, we expect to able to combine these two proofs to form a proof of the original conclusion using $Q$ instead of $P$ as hypothesis.

We use a notation $\Gamma(P)$ to denote a bunch which contains zero or more occurences of a formula $P$, and then $\Gamma(\Delta)$ to denote a similar bunch with those occurences replaced by a bunch $\Delta$.

**Theorem 11** (Substitution)**.** *Given a ribbon proof* $\mathsf{RP}_1$ *of* $\Gamma(P) \vdash Q$*, and a ribbon proof* $\mathsf{RP}_2$ *of*

$\Delta \vdash P$, *we can produce a ribbon proof* $\mathsf{RP}_3$ *of* $\Gamma(\Delta) \vdash Q$

*Proof.* (Sketch) Firstly we combine the monoids $M_1$ and $M_2$ of the two proofs. For each hypothesis $P$ in $\mathsf{RP}_1$, which occurs in a ribbon $r$, say, we incorporate a copy of the entire monoid $M_2$. I.e, $M_3$ is $M_1$ extended by $M_2$ at every $r \in M_1$ s.t. $r$ contains a copy of the hypothesis $P$.

Now we actually insert copies of the proof $\mathsf{RP}_2$ at each hypothesis $P$. We delete each $P$, and below the line it occurred in, we insert a copy of $\mathsf{RP}_2$ line by line: Each line is based on the line of $\mathsf{RP}_1$ that $P$ occurred in, with the $r$-triple replaced by the set of all the triples in this line of $\mathsf{RP}_2$, with the ribbons translated using the injection from the definition of extentsion.

It remains to show that this is indeed a ribbon proof, by showing that it can be constructed starting with the structure corresponding to $\Gamma(\Delta)$ and applying the inductive construction of $\mathsf{RP}_1$ and a number of copies of the construction of $\mathsf{RP}_2$, a straightforward but rather longwinded verification we omit here. □

## 5.2 Normalization

In [32] the normalization theory of **BI** is discussed via the $\alpha\lambda$ calculus. The $\beta$ and $\eta$ rules in particular for $*$ and $-\!\!*$ can illustrated diagrammatically for ribbon proofs, see Fig. 5.1.

However, developing this observation into a full normalization theory for ribbon proofs turns out to be less elegant. The problem lies within the box proofs fragment of ribbon proofs. In Section 2.3 we explained the problems inherent in presenting normalization in for box proofs, which all stem from the fact that a formula in a box proof (just as in a ribbon proof) may be used as a premiss for more than one rule use. The simple illustrations of Fig. 5.1 are local: they suggest that premisses will necessarily occur just before conclusions with no intervening lines. This would be a possible arrangement only if each line was used as a premiss exactly once, which is not in general the case. Any of the formulæ which should be eliminated by a reduction might be used elsewhere in the proof as a premiss for another rule. For the purely multiplicative fragment these problems may not arise:

**Conjecture 2.** *In the* $I, *, -\!\!*$ *fragment of* ***BI****, it is possible to require that each line is used as a premiss by exactly one other line, and therefore all normalizations can be localised.*

However, if the goal is simply to establish that ribbon proofs have some form of strong normalization property it suffices to use the translations between ribbon proofs and LBI proofs

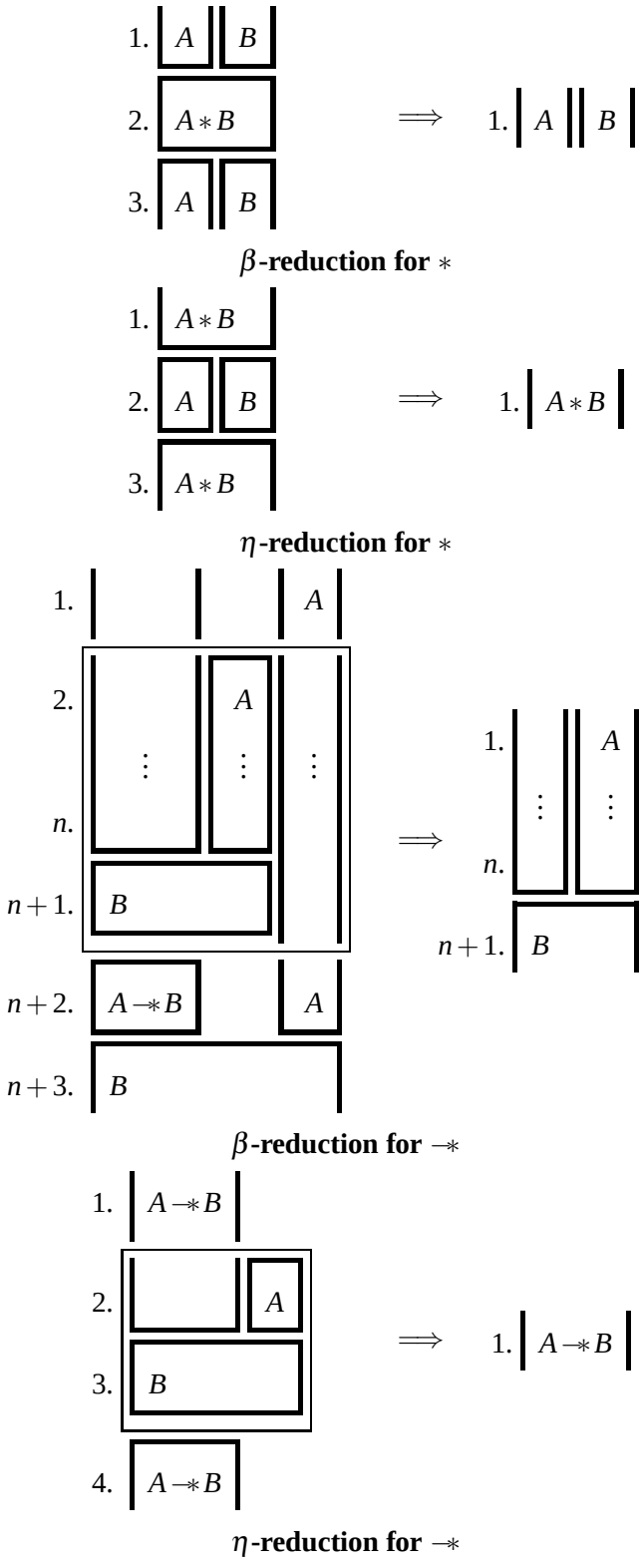



Figure 5.1: Ribbon Proof reductions

established in the relative soundness and completeness results proved in Chapter 4. We can define an equivalence class on ribbon proofs such that two proofs are equivalent if they translate to the same LBI proof. We can then talk about normalization in terms of the normalizations (cut-eliminations) of the underlying LBI proofs. Since it is proved in [32] that the $\alpha\lambda$-calculus is strongly normalising, we have a unique normal form which we can lift back to ribbons.

Note that the lifting map (which is derived from the relative completeness result, Theorem. 10) always produces ribbon proofs which are 'tree-shaped' in the sense that they don't use formulæ twice. These are of course precisely the proofs on which reductions can safely be applied, mirroring the structure of the reductions in LBI.

## 5.3 A Spatial 'Term Model'

As we have seen, part of the formalism of a ribbon proof is a ribbon monoid – and these ribbon monoids are commutative partial monoids. Now partially ordered commutative partial monoids form a model of BI. The leads us to wonder if the ribbon monoid within a particular ribbon proof is, in any sense, a model.

We will discuss this idea very precisely in the restricted framework of the $\wedge, *$-fragment of BI, where we can very simply derive a strong result amounting to a 'term model' of this fragment, which is classical in nature.

We will then continue more generally to discuss the effects the other connectives have on this picture.

### 5.3.1 Models from $\wedge, *$-proofs

Consider a ribbon proof within the $\wedge$, $*$ fragment of BI. Now we *close* the proof in the appropriate sense: we apply the rules $\wedge$-*elim* and $*$-*elim* to each applicable formula until there is no such formula which has not been used:

**Definition 37.** *A ribbon proof is said to be $\wedge, *$-closed if:*

- *It lies entirely within the $\wedge, *$ fragment of **BI**;*
- *Every formula of the form $P * Q$ has been used as the premiss of a $*$-elim rule;*
- *Every formula of the form $P \wedge Q$ has been used as the premiss of two $\wedge$-elim rules, one to derive $P$ and the other to derive $Q$.*

It is possible to close any proof simply by iteratively applying the rules. This process is terminating since the number of rule applications is bounded by the number of subformulæ in the proof.

The resulting proof remains a proof of the original theorem, albeit with various apparently unnecessary rule uses. However, it can also yield a model of the theorem: by a model of a theorem $\Gamma \vdash P$, we mean a witnessing model $M$ such that $\top_M \models \Gamma$ and $\top_M \models P$.

There are two ways of extracting this model. Most abstractly, we extract the model as the ribbon monoid of the proof.

**Definition 38.** *Given a $\wedge, *$-closed ribbon proof, define a model of **BI**. The monoid $M$ is the ribbon monoid of the outermost box. We set the forcing relation for atoms to be*

$$r \models A \iff \langle r, A, j \rangle \text{ occurs in the outermost box of the proof}$$

Note that this is a Boolean model: there is no partial order.

**Proposition 6.** *For every triple $\langle r, P, j \rangle$ in the outermost box of the proof, we have $r \models P$.*

*Proof.* By induction over the length of the formula $P$. The base case of atomic formula is Definition 38.

- $P = Q * R$

  As the proof is $\wedge, *$-closed, $*$-*elim* will have been applied to $P$. This rule creates triples $\langle s, Q, *\text{-}elim \rangle, \langle t, R, *\text{-}elim \rangle$ such that $r = s + t$. By induction $s \models Q$ and $t \models R$; by the forcing rule for $*$, $r \models Q * R$.

- $P = Q \wedge R$

  $\wedge$-*elim* will have been applied to $P$ twice, creating triples $\langle r, Q, \wedge\text{-}elim \rangle, \langle r, R, \wedge\text{-}elim \rangle$. By the forcing rule for $\wedge$, $r \models Q \wedge R$.

□

**Proposition 7.** *The model constructed is a witnessing model: $\top_M \models P$ and $\top_M \models \Gamma$.*

*Proof.*

- $\top_M \models P$: Immediate from the above proposition, since there must be some triple $\langle \top_M, P, j \rangle$ occuring in the proof.

- $\top_M \models \Gamma$

  By this we really mean that $\top_M \models Q$, where $Q$ is the **BI** proposition formed by textually substituting $*$ for , and $\wedge$ for ; in $\Gamma$. In this proof will will routinely use the notation $\models \Gamma$ for $\models Q$. The monoid $M$ is in fact defined as $M_\Gamma$, in the notation of Definition 23.

  The construction of ribbon structures allows us to pick out an unique element of the monoid for every sub-bunch $\Delta$ of $\Gamma$; this is the top element of the monoid $M_\Delta$ invoked during the inductive construction. We will use the notation $\top_\Delta$ for this element; this is a slight abuse since we really refer to a particular *occurence* of $\Delta$ as a sub-bunch. We then prove the stronger result that for every such sub-bunch $\Delta$ this element $\top_\Delta \models \Delta$, and we proceed by induction over the length of $\Delta$.

  - $\Delta = P$

    Then the ribbon $\top_\Delta$ will actually include the formula $P$ as a hypothesis, and therefore $\top_\Delta \models P$.

  - $\Delta = \Delta_1; \Delta_2$

    By induction $\top_{\Delta_i} \models \Delta_i$. However, $\top_{\Delta_1} = \top_{\Delta_2} = \top_\Delta$, so $\top_\Delta \models \Delta_i$ and by the forcing clause for $\wedge$, $\top_\Delta \models \Delta_1 \wedge \Delta_2$, i.e. $\top_\Delta \models \Delta$.

  - $\Gamma = \Delta_1, \Delta_2$

    By induction $\top_{\Delta_i} \models \Delta_i$. Since $\top_{\Delta_1} + \top_{\Delta_2} = \top_\Delta$, we have $\top_\Delta \models \Delta_1 * \Delta_2$, and we are done.

□

More concretely, we can produce a geometrical model based on the actual representation of the proof on paper, by 'squashing' the proof vertically and taking the model to be, for each ribbon $r$, an open interval of the real line. Then the monoid operation '$\cdot$' is almost the union of sets: in fact, we take the interior of the closure of the union. To make the model work, we need to be careful that no $r \neq s$ map to exactly the same set. These ideas are taken further in the next chapter.

The geometrical model is, of course, the same monoid as the first, so the same model in an algebraic sense; it provides a concrete representation of it.

### 5.3.2 Proofs with $I$

In the classical model theory (that is, a PCM without a partial order), the unit $I$ is only forced at the identity element of the monoid. So if the proof contains the triple $\langle r, I, j\rangle$, the procedure above fails to produce a model: $r \models I$ is not permitted. Intuitively, by proving $I$, we have established that $r$ was in fact, the identity ribbon. Formally, we would like to take some quotient of the monoid such that $r = e$. Unfortunately, since the monoid is a partial monoid, this quotient is not a straightforward affair.

### 5.3.3 General proofs

To generalize the above ideas, we could consider the following idea for the forcing relation of a model:

> $r \models P$ if and only if there exists some proof extending the given proof, in the sense of applying some finite number of proof rules, in which a triple $\langle r, P, j\rangle$ occurs.

Note that this is indeed a generalization of the $\wedge, *$ model given above. However in the general case it fails to be a model in the same sense. For example:

- The are problems, mentioned above, with $I$. The problem is to find, within a partial monoid, the least *congruence* which maps a particular set of elements to the identity. With total monoids this is easy; with partial monoids it is more involved. The other problems described below mean it is not worth spending time on here.

- The are more serious problems with $-\!\!*$. Consider the trivial (one line) proof of $Q \vdash Q$. It generates a monoid with a single non-identity element, $r$ say. Now the forcing clause for $-\!\!*$ in this case is

  $$r \models P -\!\!* Q \iff \forall s \text{ such that } s \cdot r \text{ is defined } s \cdot r \models Q$$

  Since there are *no* sums $s \cdot r$ defined, the universal is vacuously satified, and $r \models P -\!\!* Q$. However, there is certainly no way of proving $P -\!\!* Q$ in the ribbon $r$, so the analogue of Proposition 6 cannot be proved.

# Chapter 6

# Geometry

---

Our formalism of ribbon proofs is essentially set-theoretic in form, and it fails to really draw out the geometric intuition present in the informal depiction of ribbon proofs. In this chapter, we will refer to this informal notation as 'ribbon pictures'. We will give some interpretations of ribbon proofs in spaces based on $\mathbf{R}^2$, in a fashion which is intended to formalize ribbon pictures.

We will work throughout in $\mathbf{R}^2$ with a chosen coordinate system. We will consider the first $\mathbf{R}$ factor to be horizontal ('the $x$ direction'), and the second vertical ('the $y$ direction'). We will consistently use the term 'rectangle' to denote an orthogonal rectangle, aligned with these axes.

We will work with half-open sets:

**Definition 39.** *A set $I \subset \mathbf{R}$, is a* half-open interval *if it is of the form $[a,b)$. A set $r \subset \mathbf{R}^2$ is a* half-open rectangle *if it is of the form $[a,b) \times [c,d)$. More generally a set $r \subset \mathbf{R}$ (resp. $\mathbf{R}^2$)is a* half-open *set if it is the union of a finite number of half-open intervals (resp. rectangles).*

We define an operation which can make sets of half-open sets into partial commutative monoids.

**Definition 40.** *Given two half-open sets $r,s$,*

$$r \cdot s = \begin{cases} r \cup s \text{ if } r \text{ and } s \text{ disjoint} \\ \text{undefined otherwise} \end{cases}$$

Now we define a notion of a 'geometric' ribbon proof for the fragment of ribbon proofs without boxes. We will show how to interpret ribbon proofs in this geometric setting, and demonstrate that various proof-theoretically natural notions are interpreted as geometrically natural notions.

**Definition 41.** *A* geometric ribbon structure *is a set of disjoint half-open sets $O$ in $\mathbf{R}^2$ together with a finite set $L$ of labels $\langle p, P, j\rangle$ such that*

- *All the open sets $o \in O$ are of the form $U \times [a,b)$ for $U$ a finite union of half-open intervals;*
- $\bigcup_{o\in O} o$ *is a half-open rectangle.*
- *Each label $\langle p, P, j\rangle$ is a point $p$ contained in some $o \in O$, together with a **BI** proposition $P$ and a justification $j$ just as in a ribbon proof.*

Each such geometric ribbon structure has an underlying ribbon structure.

**Definition 42.** *The* underlying *ribbon structure to a geometric ribbon structure is defined as follows. It has as ribbon monoid the set of horizontal projections of $O$ under the operation '$\cdot$'.*

*For each distinct y-coordinate $y_i$ inhabited by the points $p$ in the labels, we define a line. For each label $\langle p, P, j\rangle$ such that $p_y = y_i$, we include the triple $\langle r, P, j\rangle$ where $r$ is the horizontal projection of the set $o$ to which $p$ belongs. In the event that the $\cdot$-sum of all the $r$ is not the top element of the monoid, a triple $\langle r', nothing, nothing\rangle$ is added with $r'$ as necessary to complete the line.*

Conversely, we can construct a geometric ribbon structure from any ribbon structure.

**Proposition 8.** *We can construct a geometric ribbon structure from any box-free ribbon structure.*

In ribbon pictures, the horizontal dimension controls the algebra of the ribbons (that is, the ribbon monoid). In order to translate our ribbon proofs into this geometric model, we need to embed the ribbon monoid into an interval. The key result in proving this proposition is the embedding of an arbitrary ribbon monoid into $[0,1) \subset \mathbf{R}$, such that the monoid operation is realised by the '$\cdot$' of Definition 40.

**Proposition 9.** *Every ribbon monoid is isomorphic to a ribbon monoid of half-open subsets of $[0,1)$ under $\cdot$.*

*Proof.* Consider the ribbon monoid via its representation in the subset algebra of some finite set $X$. Suppose $|X| = n$, and number the elements of $X$ as $x_0, \ldots, x_{n-1}$. Then define a function $I : X \to P(\mathbf{R})$ as

$$I(x_i) = \left[\frac{i}{n}, \frac{i+1}{n}\right)$$

Now extend $I$ to a function $P(X) \to P(\mathbf{R})$, defining for $Y \subset X$

$$I(Y) = \bigcup \{I(x_i) : x_i \in Y\}$$

Finally define $I(\emptyset) = \emptyset$.

It is easy to check that, with $\cdot$ defined as in Definition 40, $I$ is a monoid isomorphism. □

We will assume that some such interpretation has been fixed for every ribbon monoid we deal with, and we will continue to use the notation $I(r)$ for the half-open subset of $[0,1)$ corresponding to a ribbon $r$. We continue to use $[0,1)$ for definiteness, but the interval chosen is arbitrary.

The vertical dimension in our ribbon pictures stands for 'progress' in the construction of the proof: proving new propositions from old. To convert our ribbon proofs into geometric ribbon proofs, we arbitrarily assign some vertical space to each line of the proof: The $i$th line of the proof is embedded in the interval $[i-1,i)$.

**Definition 43.** *Define a map $L : R \to P(\mathbf{N})$ ('occurs-in lines') as*

$$L(r) = \{i : \textit{There is some triple } \langle r,P,j\rangle \textit{ in line } i\}$$

**Definition 44.** *Define an interpretation of ribbons $[[-]] : R \to \mathbf{R}^2$:*

$$[[r]] = \bigcup_{i \in L(r)} I(r) \times [i, i-1)$$

Now we come to the proof of Proposition 8.

*Proof.* Let

$$O = \{[[r]] : r \text{ a ribbon in the ribbon monoid}\}$$

For each triple $\langle r,P,j\rangle$ in the proof add a label $\langle p,P,j\rangle$ where $p = \left\langle x, i+\frac{1}{2}\right\rangle$ such that $x \in I(r)$ and $i$ is the 'line-number' of the line in which $\langle r,P,j\rangle$ occurs. □

Now we formulate from the notion of geometric ribbon structure a notion of geometric ribbon proof:

**Definition 45.** *A* geometric ribbon proof *is a geometric ribbon structure whose underlying ribbon structure is a proper ribbon structure.*

We note that our definitions relating ribbon structures and geometric ribbon structures are related in the way we would expect:

**Proposition 10.** *Composing the construction of a geometric ribbon structure given by Proposition 8 with the notion of underlying ribbon structure in Definition 42 recovers a ribbon structure equivalent to the original.*

What have we gained by moving to geometric proofs? We have related the informal diagrams which we have been using to notate proofs to some formal geometric ideas, giving substance to the suggestion that the proofs contain real spatial content. This approach to the proof system is highly suggestive of possibly geometrical applications of **BI** and related logics to reasoning about area or volume.

Various natural proof constructions become very simple geometrically. Most strikingly, putting two proofs alongside one another is literally the obvious geometrical concept – a scale and a translation. Of course, the resultant object is a slight generalization of a proof, since it has two conclusions, but it isn't hard to see what it means. To formalize that observation, we need to explain how to apply an affine map to our geometric proof objects. In particular, we are interested in non-degenerate — that is, invertible — affine maps.

**Definition 46.** *We define the effect of applying an non-degenerate affine map to a geometric proof as follows:*

- *We apply the map to each element of O;*
- *We apply the map to each point $p$ of every label.*

We aren't particularly interested in rotating or shearing our proofs, so we restrict:

**Definition 47.** *A rectangular map is a non-degenerate scaling followed by a translation.*

Now, we take two proofs, one in rectangle $[l_1, r_1) \times [t_1, b_1)$ and the other in rectangle $[l_2, r_2) \times [t_2, b_2)$. We can now make precise the notion of putting the two proofs 'side-by-side'. There is a unique rectangular map $\phi$ under which the second proof can be scaled and translated into the rectangle $[r_1, r_1 + 1) \times [t_1, b_1)$: i.e. we put the second proof 'immediately to the right of' the first proof.

Now, the object formed by taking the simple set theoretic union of these two proofs' *O*s and labels is another geometric proof. Note that geometric proofs, unlike ribbon proofs, do not have a requirement to have a unique conclusion. Such a requirement does not seem natural in the light of geometric transformations like this.

Also, cut has an appealingly simple interpretation. Given a (geometric interpretation of) a ribbon proof of $\Delta(P) \vdash Q$ and one of $\Gamma \vdash P$, we want to use a rectangular map to map the entire proof $\Gamma \vdash P$ into each of the 'rectangles' containing $P$ as a hypothesis. Of course, the set containing $P$ need not be a half-open rectangle at all, in general it will be a finite union of half-open rectangles all having the same vertical ($y$) dimensions. Therefore rather than a rectangular map we have to use a piecewise continuous map, the sum of a finite number of rectangular maps, to map into each of the components of the finite union.

Note that this geometric approach is not quite that depicted in our informal diagrams. The approach here allows ribbons to be disconnected unions of half-open rectangles; our diagrams go to considerable lengths to make ribbons connected, and this is the real force of the twist pseudo-rule.

To generalize this technique to boxes we use a third dimension. We will only sketch this generalization here but its formal interpretation should be clear. This third dimension doesn't need to be continuous, so we may as well work in $\mathbf{R} \times \mathbf{R} \times \mathbf{N}$. Since each box is a subproof, and a subproof may in general be based on a different monoid, we map the subproof onto $\mathbf{R} \times \mathbf{R} \times \{1\}$. If a particular box is being mapped into $\mathbf{R} \times \mathbf{R} \times \{n\}$, and it has subproofs, then those proofs are interpreted in $\mathbf{R} \times \mathbf{R} \times \{n+1\}$.

In the outer box there is nothing at all during the $y$-coordinate interval occupied by the inner box.

As an example of the construction, we can now give a geometric proof of the Deduction Theorems:

**Theorem 12** (Deduction Theorems)**.**

1. $\Gamma; P \vdash Q$ *if and only if* $\Gamma \vdash P \to Q$
2. $\Gamma, P \vdash Q$ *if and only if* $\Gamma \vdash P \mathrel{-\!\!*} Q$

*Proof.*

1. Take a geometric proof of $\Gamma; P \vdash Q$. Translate all the proof except the hypotheses $\Gamma$ (but including the hypothesis $P$) by $+1$ in the $z$ direction. Relabel $P$ as being an assumption. Add a new 'line' to the end of the proof concluding $P \to Q$.

   Conversely, take a geometric proof of $\Gamma \vdash P \to Q$, add a new hypothesis $P$ and use $\to$*-elim* to conclude $Q$.

2. Take the geometric proof of $\Gamma, P \vdash Q$ and translate all the proof except the hypotheses $\Gamma$ by $+1$ in the $z$ direction. Totally remove the vertical strip previously occupied by $P$ from the lower part of the proof, drawing the remainder together if necessary. Relabel $P$ as being an assumption and add a new conclusion $P \mathrel{-\!\!*} Q$ at the end of the proof.

□

# Chapter 7

# Pointer Logic

---

In this chapter we consider a particular application of **BI**. Yang's paper[40] gives a proof of correctness of the Schorr-Waite graph marking algorithm — a technique for doing a depth-first traversal in constant space: without requiring either an explicit stack, or the use of the host programming language's call stack. This is a practical algorithm, which could form the basis of the mark phase of a garbage collector, and it uses pointer manipulation 'tricks' which have rendered it opaque to traditional approachs to program proof. There is an earlier formal attempt by Bornat in [5], who recognised the key ideas of separation underlying succesful proof, but this is significantly improved on by Yang's method, which works in the system described by Ishtiaq and O'Hearn in [24] which is in turn based on **BI**.

Yang's proof, however, is reasoned semantically and in particular it relies upon the truth of some 'semantic implications' given in the appendix of [40]. In this chapter we will set out a system based on ribbon proofs, containing some additional proof rules relevant to this domain, and use it to give nearly formal proofs of some of Yang's semantic implications. The proofs are only nearly formal because we have a slightly informal approach to variables and subsitution for the purpose of this chapter.

We describe a simple version of Ishtiaq and O'Hearns's heap cell model. Each world in the model is a finite set of heap cells, where a heap cell is an address and a value. For simplicity, let addresses and values be natural numbers; then a heap is a partial function $\mathbf{N} \rightarrow \mathbf{N}$. We will work exclusively with finite heaps; that is partial functions of finite domain. This model is a partial commutative monoid defining '$\cdot$' as union of partial functions where the domain is disjoint; where

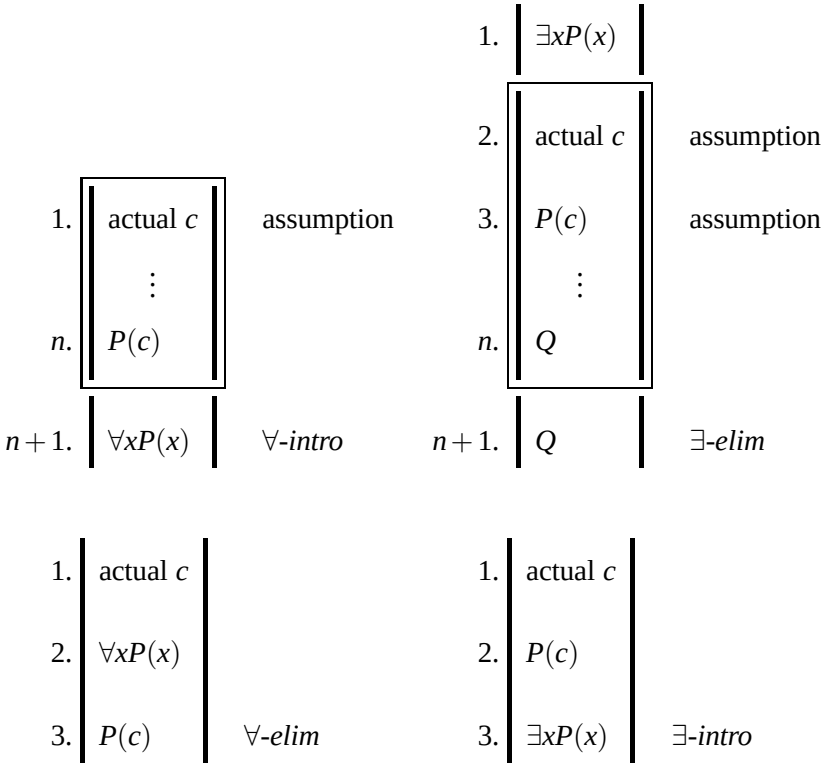



Figure 7.1: Boxproof rules for $\forall, \exists$

the domains are not disjoint the composition is undefined.

The basic predicate of the system, apart from equality, is the 'points to' relation, $x \mapsto a$. This denotes 'the current heap contains exactly one cell, address $x$, with value $a$'. From that we define a predicate which denotes 'the current heap contains the cell with address $x$, having value $a$':

**Definition 48.** $x \hookrightarrow a \equiv x \mapsto a * \top$

## 7.1 Proof rules for metavariable quantification

The logical notation used in [29, 40] uses variables and quantifies over them. In this thesis we have considered only propositional **BI**. In [32], Pym develops a full theory of predicate **BI**, in which variables are bunched into contexts. The use of variables in [29] is much simpler than this; it deals only with global metavariables.

The global nature of these variables means that if $x = y$ is proved in any ribbon, in is globally true, and the fact can be used in any ribbon. We adopt therefore a special rule *copy* which allows us to copy such 'global facts' between ribbons freely (but respecting the box discipline).

We adopt a formalization for quantifiers which the author was introduced to whilst assisting on a logic course taught by Bornat, using formulae of the form 'actual $c$' to denote 'some object

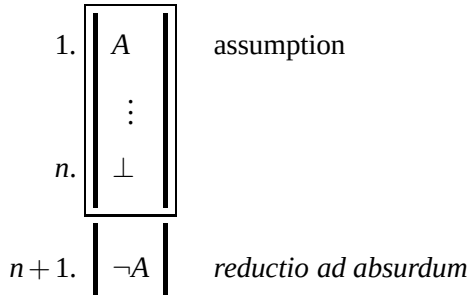



Figure 7.2: Box proof notation for *reductio ad absurdum*

*c* exists'. A very similar formal treatment can be found in [3]. Our rules are shown in Figure 7.1.

## 7.2 Reductio ad absurdum

We are forced to adopt a rather unattractive form of the rule *reductio ad absurdum*. A common schema for this rule in box proofs contexts is shown in Figure 7.2. Separation logic has a classical semantics, so we need to adopt some form of this rule. However, there is a strange interaction between ribbons and contradiction. It makes good semantic sense to permit the rule to begin with an assumption in any ribbon, deduce a contradiction in an arbitrary (possibly different) ribbon, and conclude the negation of the assumption in the initial ribbon; for, once a situation is shown to be contradictory somewhere, then the whole situation is contradictory. However, this rule is unpleasantly non-local in application.

## 7.3 Proof rules to validate certain semantic lemmas

We add proof rules to our system to make certain fundamental semantic facts provable.

### 7.3.1 Heaps with only one cell

When there is only one cell in the heap, all $\mapsto$ formulæ must be referring to the same cell.

**Lemma 5.** $x \mapsto a \wedge y \mapsto b \models x = y \wedge a = b$

This lemma can be made provable by including the proof rule

$$\frac{\Gamma \vdash x \mapsto a \quad \Delta \vdash y \mapsto b}{\Gamma;\Delta \vdash x = y \wedge a = b}$$

but that is not the most general. Since $x \mapsto a$ tells us there is only one cell in the heap, given $x \mapsto a$, we can conclude from $y \mapsto b * P$ not only that the $x$ and $y$ cells are the same, but also that $P$ must hold of an empty heap.

**Rule 1.** *Onecell*

$$\frac{\Gamma \vdash x \mapsto a \quad \Delta \vdash y \mapsto b * P}{\Gamma;\Delta \vdash x = y \wedge a = b \wedge (x \mapsto a * (P \wedge \mathsf{emp}))}$$

### 7.3.2 Matching cells

When two $\hookrightarrow$ formulæ refer to the same address, they must refer to the same cell:

**Lemma 6.** $x \hookrightarrow a \wedge x \hookrightarrow b \models a = b$

We need a rule which allows us to conclude that two heap cells are the same:

**Rule 2.** *Location*

$$\frac{\Gamma \vdash x \mapsto a * P \quad \Delta \vdash x \mapsto b * Q}{\Gamma;\Delta \vdash (x \mapsto a \wedge x \mapsto b) * (P \wedge Q)}$$

### 7.3.3 Case analysis

Given a $\hookrightarrow$ formula in additive conjuction with a spatial conjunction as in $(x \hookrightarrow a) \wedge (B * C)$, there are precisely two possibilities for the location of the $x$ cell. The formula $B * C$ says that the heap divides into two, and the $x$ cell must be in one half, or the other. So we must have either $(x \hookrightarrow a \wedge B) * C$ or $(B * (x \hookrightarrow a \wedge C)$. We can write this as a proof rule:

**Rule 3.** $\hookrightarrow$ *cases*

$$\frac{\Gamma \vdash x \hookrightarrow a \quad \Delta \vdash B * C}{\Gamma;\Delta \vdash \big((x \hookrightarrow a \wedge B) * C\big) \vee \big(B * (x \hookrightarrow a \wedge C)\big)}$$

There is similar situation with $\mapsto$. If $x \mapsto a \wedge (B * C)$ holds, then either the $x$ cell holds in the heap where $B$ holds — in which case, that heap is one cell, and the heap where $C$ holds is empty — or vice versa:

**Rule 4.** $\mapsto$ *cases*

$$\frac{\Gamma \vdash x \mapsto a \quad \Delta \vdash B * C}{\Gamma;\Delta \vdash \big((x \mapsto a \wedge B) * (C \wedge \mathsf{emp})\big) \vee \big((B \wedge \mathsf{emp}) * (x \mapsto a \wedge C)\big)}$$

### 7.3.4 Empty heaps

Heaps are either empty, or they have at least one cell. If a heap contains exactly one cell, then it satisfies

$$\mathsf{Cell} \equiv (\exists x, a)(x \mapsto a)$$

and if it contains at least one cell, it satisfies $\mathsf{Cell} * \top$.

We encapsulate this with two reversible rules.

**Rule 5.** *Emptyheap*

$$\frac{\neg\mathsf{emp}}{\mathsf{Cell} * \top} \qquad \frac{\mathsf{emp}}{\neg(\mathsf{Cell} * \top)}$$

It is straightforward to check that these rules are all sound for our system.

**Theorem 13** (Soundness)**.** *These rules are sound for the system of partial functions in* $\mathbf{N} \to \mathbf{N}$ *described above.*

## 7.4 Some proofs

### 7.4.1 Making pointer logic count

The following problem was posed by Reynolds and Yang. We can specify 'the heap contains at least two cells' as $\mathsf{Cell} * \mathsf{Cell} * \top$, and we can specify 'the heap contains at least three cells' similarly as $\mathsf{Cell} * \mathsf{Cell} * \mathsf{Cell} * \top$.

We would hope to be able to prove that if we have at least two cells, but not at least three cells, then we have exactly two:

$$\mathsf{Cell} * \mathsf{Cell} * \top; \neg(\mathsf{Cell} * \mathsf{Cell} * \mathsf{Cell} * \top) \vdash \mathsf{Cell} * \mathsf{Cell}$$

And we can prove exactly that using the *Heapempty* rule, as shown in Figure 7.3.

### 7.4.2 All cells contain ones

The following example is the essential content of Rule B.1.1 of [40]. Suppose we define the predicate $\mathsf{AllOnes}$ to mean that every heap cell contains a one:

$$\mathsf{AllOnes} \equiv (\forall x)(x \hookrightarrow ?) \to (x \hookrightarrow 1)$$

Then the following should certainly hold:

$$\mathsf{AllOnes} * (y \mapsto 1) \vdash \mathsf{AllOnes}$$

The notation $x \hookrightarrow ?$ is a convenient short-hand for $(\exists z)(x \hookrightarrow z)$ which avoids having to 'waste' a name for the variable. We show a proof of this lemma in Figure 7.4.

### 7.4.3 Valid pointers

Now another example motivated by Yang's paper, this time based on B.1.5. Suppose that $\mathsf{VP}(x)$ is a predicate meaning that $x$ is a valid pointer: we define it as $\mathsf{VP}(x) \equiv x = \mathsf{nil} \vee x \hookrightarrow ?$. Note

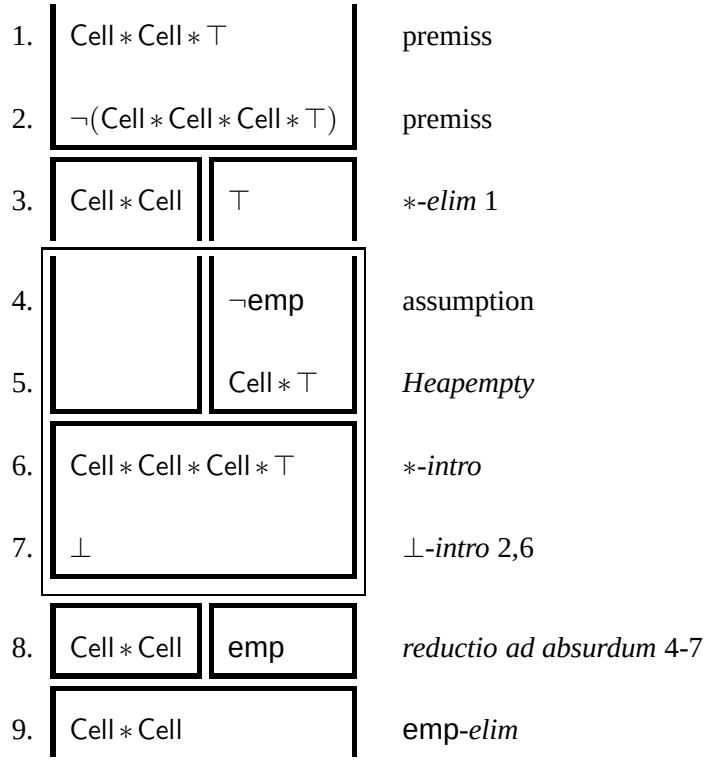



Figure 7.3: Making pointer logic count

that null pointers are considered valid: invalid pointers are those which point outside the current heap.

Now we wish to prove $x \hookrightarrow a; \mathsf{VP}(y) \vdash x \mapsto a * (x \mapsto a' \mathrel{-\!\!*} VP(y))$, which says something like 'if $x$ points to $a$ and $y$ is valid, then $y$ would still be valid in the similar heap where $x$ pointed instead to $a'$'. The proof is shown in Figure 7.5.

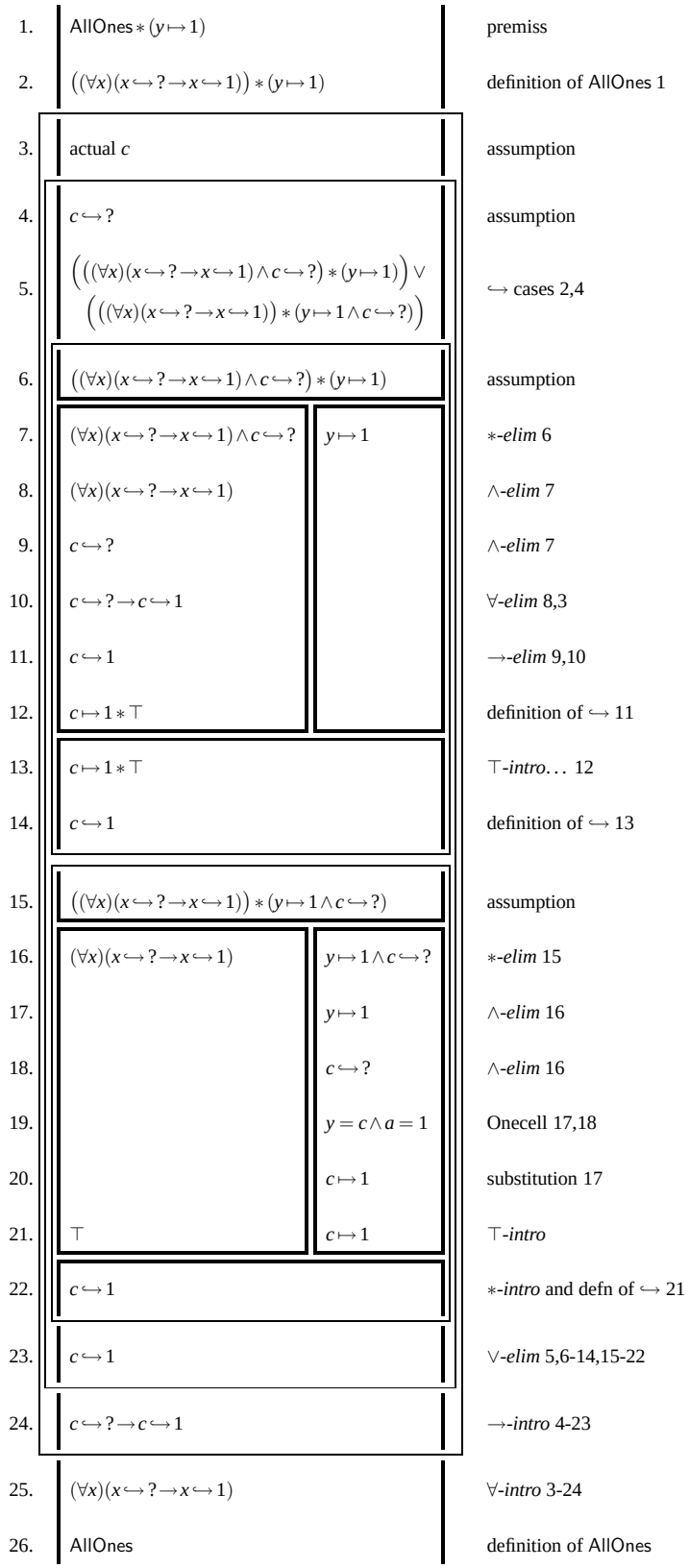



Figure 7.4: All cells contain ones

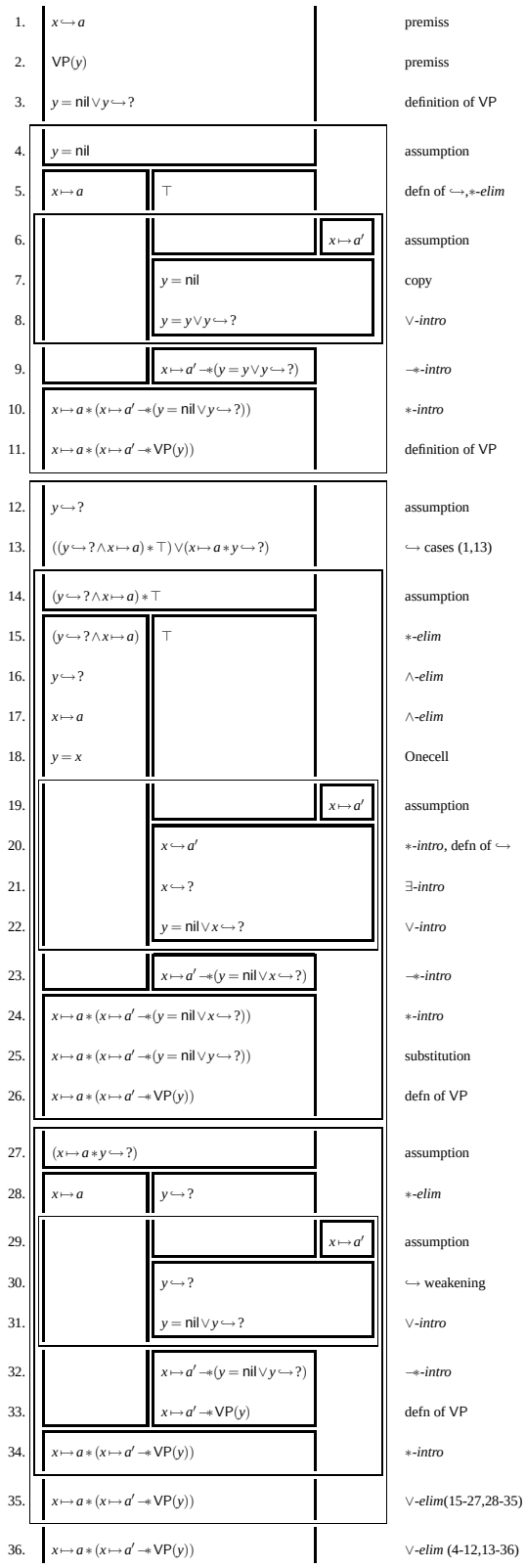



Figure 7.5: Valid Pointers

# Chapter 8

# Implementation

---

One of the problems which has faced almost every attempt to describe a proof system in a genuinely formal fashion has been a tendency to gloss over a few apparently harmless features which seem obvious, but in actual fact require some delicacy to make precise. One example in the formalization of natural deduction for intuitionistic logic is the notion of discharged hypothesis, and the problems of linking the point of discharge with the discharged formulae; one in the formulation of Gentzen-style sequent calculi is the question of the multiplicative or additive presentation of proof rules.

This was certainly a problem which we encountered (and we are grateful to the referees of the MFPS paper which preceded this thesis for pointing out some substantial areas of vagueness). One tool which we adopted to help resolve some of these problems was the development of an implementation of the formalization in ML. Encoding the the formalization into ML's type system raised (and suggested the solution to) several issues.

The implementation was never developed to the point of being an independently useful program, and lessons having been learnt from it, it is now somewhat out of step with the formalization as it is described in Chapter 4.

In the appendix, we give the listing of the program (in the OCaml dialect of ML). It consists of a simple recursive descent parser for parsing formulae and bunches, and the types for the basic concepts of ribbon proofs. The important types are:

- `formula` for formulæ in **BI**. Actually, the type represents general binary expressions,

- `bunch` for bunches of formulæ,
- `oneribbonline` to encode the notion of triple, encoding ribbon and formula but omitting justification. The structures therefore lack the notion of consequence tying the lines of the proof together, and represent ribbon structures rather than ribbon proofs,
- `line` to encode the notion of a set of triples; but we use an ML list, not a set, for programmatic reasons. This creates some equivalent proofs but we are not attempting to use any underlying ML notion of equality,
- `boxpart`, either a `box` or a `line` where
- `box` is a list of `boxpart`s, and
- `ribbonproof` is just a `box`,

In fact, a `ribbonproof` is a closer approximation to the notion of ribbon structure than that of ribbon proof. The intention was that a family of high-level constructors would model the notion of ribbon proof. The implementation listed does in fact contain the necessary notion of ribbon structure corresponding to a bunch; the other requirement is a constructor for each of the ribbon transformations of Definition 25.

The implementation fails to model the ribbon monoid within each box. The type `splits` was an ultimately unsuccessful attempt to model a slightly simpler structure that at one stage we hoped was sufficient, by modelling only the branching structure of bunches. This was in an attempt to avoid incorporating a general theorem-prover in the theory of partial commutative monoids. Ultimately this led to the definition in Section 4.2 of a ribbon monoid. Given the fact, proved in that section, that all ribbon monoids can be represented in powerset monoids, there is a natural way of representing these monoids computationally. However it is not without problems since the size of the set required can in principle be exponential in the size of the bunch of hypotheses.

The following functions merit a brief explanation:

- `bunchtoproof` implements the notions of ribbon structure corresponding to a bunch,
- `nIntro`, and the commented-out `starIntroduction` are the beginnings of an implementation of the ribbon structure transformations corresponding to ribbon proof rules,

- `calc_widths` and related functions perform the task of formatting a text-only representation of ribbon proofs for use with interactive sessions.

As an exercise with very limited goals, the implementation was a success — it enabled us to pin down precisely our formalization. However, given more time we would very much like to extend it to a full toolkit for manipulating and displaying ribbonproofs, especially to explore their compositional and geometrical nature.

# Bibliography

[1] AR Anderson and JD Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, 1975.

[2] A.R. Anderson, J.M. Dunn, and N.D. Belnap. *Entailment: the Logic of Relevance and Necessity, volume II*. Princeton University Press, 1992.

[3] J Barwise and J Etchemendy. *The Language of First-Order Logic*. Number 34 in Lecture Notes. CSLI, Stanford, 1992.

[4] R Bornat and BA Sufrin. Animating formal proof at the surface: the Jape proof calculator. *The Computer Journal*, 43(3):177–192, 1999.

[5] Richard Bornat. Proving pointer programs in hoare logic. In *Mathematics of Program Construction*, pages 102–126, 2000.

[6] K Broda, S Eisenbach, H Khoshnevisan, and Steven Vickers. *Reasoned Programming*. Prentice Hall, 1994.

[7] L Cardelli, P Gardner, and G Ghelli. A spatial logic for querying graphs. In *ICALP'02*, 2002.

[8] Luca Cardelli and Andrew D Gordon. Anytime, anywhere. Modal logics for mobile ambients. In *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, pages 365–377, 2000.

[9] JM Dunn. Relevance logic and entailment. In D Gabbay and F Guenthner, editors, *Handbook of Philosophical Logic*, volume III, chapter 3, pages 117–224. D Reidel, 1986.

[10] Frederic B Fitch. *Symbolic Logic: An Introduction*. The Ronald Press Company, 1952.

[11] JA Fodor. Modules, frames, fridgeons, sleeping dogs and the music of the spheres. In Z Pylyshyn, editor, *The robot's dilemma: The frame problem in artificial intelligence*. Ablex, Norwood, NJ, 1987.

[12] D Gabbay, C Hogger, and J Robinson, editors. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Oxford University Press, Oxford and New York, 1994.

[13] DM Gabbay. *Fibring Logics*. Oxford University Press, 1998.

[14] D Galmiche and D Méry. Proof-search and countermodel generation in propositional BI logic. 2001.

[15] Didier Galmiche, Daniel Méry, and David Pym. Resource tableaux (extended abstract). In *Proceedings of CSL'02*, LNCS. Springer-Verlag, 2002.

[16] G Gentzen. *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969. English translation, edited and introduced by ME Szabo.

[17] J-Y Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[18] J-Y Girard. Linear logic: its syntax and semantics. In J-Y Girard, Y Lafont, and L Regnier, editors, *Advances in linear logic*. London Mathematical Society Lecture Note Series, Cambridge University Press, 1995.

[19] J-Y Girard, Y Lafont, and P Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[20] S Hanks and D McDermott. Default reasoning, nonmonotonic logic, and the frame problem. In *Proceedings of the American Association for Artificial Intelligence*, pages 328–333, 1986.

[21] D Hilbert and W Ackermann. *Grundzge der theoretischen Logik*. Springer, Berlin, 1967.

[22] CAR Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–583, 1969.

[23] Michael R A Huth and Mark D Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England, 2000.

[24] S.S. Ishtiaq and P. O'Hearn. **BI** as an assertion language for mutable data structures. In *28th ACM-SIGPLAN Symposium on Principles of Programming Languages, London*, pages 14–26. Association for Computing Machinery, 2001.

[25] S Jaskowski. On the rules of suppositions in formal logic. In McColl, editor, *Polish Logic 1920-1939*. Oxford, 1967.

[26] S Kripke. Semantic analysis of intuitionistic logic I. In JN Crossley and MAE Dummett, editors, *Formal Systems and Recursive Functions*, pages 92–130. North-Holland, 1965.

[27] J Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–170, 1958.

[28] J Lambek and PJ Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge University Press, 1984.

[29] Peter O'Hearn, John Reynolds, and Hongseok Yang. Local reasoning about programs that alter data structures. In *Proceedings of CSL'01*, LNCS, pages 1–19. Springer-Verlag, 2001.

[30] PW O'Hearn and DJ Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[31] DJ Pym. On bunched predicate logic. In *LICS 1999*. IEEE, 1999.

[32] DJ Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series. Kluwer Academic Publishers Boston/Dordrecht/London, 2002.

[33] D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: The semantics of **BI**. *Theoretical Computer Science*, 2004. To appear. Preprint available at `http://www.cs.bath.ac.uk/~pym/recent.html`.

[34] S Read. *Relevant Logic: A Philosophical Examination of Inference*. Basil Blackwell, 1988.

[35] R Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial intelligence and mathematical theory of computation: Papers in honor of John McCarthy*, pages 359–380. Academic Press, Boston, 1991.

[36] G Restall. *An Introduction to Substructural Logics*. Routledge, 2000.

[37] JC Reynolds. Intuitionistic reasoning about shared mutable data structure. In *Millenial Perspectives in Computer Science*. Palgrove, 2000.

[38] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.

[39] Richmond H Thomason. *Symbolic Logic: An Introduction*. Macmillan, Toronto, 1970.

[40] Hongseok Yang. An example of local reasoning in BI pointer logic: the Schorr-Waite graph marking algorithm. *SPACE workshop, London*, 2001. Available from http://ropas.snu.ac.kr/ hyang/paper/SchorrWaite.ps.

# Appendix A

# Implementation: ML source code

---

```
(* some basic utility functions *)

let explode s =
  let rec explode_aux s n m =
    if n=m then [] else (s.[n])::(explode_aux s (n+1) m)
  in
      explode_aux s 0 (String.length s);;

let rec implode clist =
  match clist with
      [] -> ""
    | c :: tail -> (String.make 1 c) ^ implode tail
;;

let rec takewhile pred list =
  match list with
      [] -> []
    | hd :: tl ->
          if (pred hd) then hd :: takewhile pred tl
          else []
;;

let rec dropwhile pred list =
  match list with
      [] -> []
    | hd :: tl ->
          if (pred hd) then dropwhile pred tl
          else list
;;

let rec nchars c n = if n=0 then "" else (c^(nchars c (n-1)));;
let nspaces=nchars "␣";;
```

```
let min a b = if a < b then a else b;;
let max a b = if a > b then a else b;;

(* an interface to associative arrays which I prefer to the ocaml
   standard one *)

(* Check if a key occurs *)
let member_assoc k a =
  List.mem_assoc k a
      ;;

(* Set a key, removing old value if possible *)
let set_assoc (k,v) a =
  (k,v) :: (List.remove_assoc k a)
;;

(* Lookup a key : throws Not_found *)
let get_assoc k a =
  List.assoc k a
;;

(* Lookup a key with a default value for use if it does not occur *)
let get_assoc_def d k a =
  try List.assoc k a
  with Not_found -> d
;;

(* Apply a function to the value at a key : throws Not_found *)
let alter_assoc k f a =
  let v = get_assoc k a
  in let vv = f v
  in set_assoc (k,vv) a
;;

(* Merge a list of assocs into a single assoc, applying the
   function 'choose' to choose which of a conflicting pair
   to use *)

let rec assoc_merge choose a b =
  match b with
      [] -> a
    | (k,v) :: t ->
          if not (member_assoc k a) then
              assoc_merge choose ((k,v)::a) t
          else
              assoc_merge choose (set_assoc (k,choose k v (get_assoc k a)) a
;;
```

```
(* utility functions for tokenising (first phase of parsing)*)

module Tokeniser = struct
  type token = T2_ID of string | T2_Op of string

  type charclass = IDLetter | OpChar | Other

  module OrderedChar = struct type t=char let compare=compare end
  module CharSet = Set.Make(OrderedChar)

  let rec make_charset = function
      [] -> CharSet.empty
    | x::xs -> CharSet.add x (make_charset xs)

  let letterset = make_charset (explode (
                                    "abcdefghijklmnopqrstuvwxyz"^
                                    "ABCDEFGHIJKLMNOPQRSTUVWXYZ_"))
  and opset = make_charset (explode "*-&><|")

  let classof c =
    if CharSet.mem c letterset then IDLetter
    else if CharSet.mem c opset then OpChar
    else Other

  let isClass cls c = ((classof c) = cls)

  let rec tokenise s = tokenise_clist (explode s)
  and tokenise_clist clist =
    match clist with
        [] -> []
      | c :: tl ->
            match classof c with
                IDLetter ->
                    T2_ID (implode (takewhile (isClass IDLetter) clist)) ::
                    tokenise_clist (dropwhile (isClass IDLetter) clist)
              | OpChar ->
                    T2_Op (implode (takewhile (isClass OpChar) clist)) ::
                    tokenise_clist (dropwhile (isClass OpChar) clist)
              | Other ->
                    T2_Op (implode [c]) :: tokenise_clist tl
end;;

(* utility functions for parsing *)

module Parse = struct

          (* Toolkit for building parsers *)

          (* Each production in your grammar should be a function
             of type 's -> 'a * 's, where 's represents the parser state
             and 'a is the abstract type corresponding to the production
```

```
                For example, 's might well be token list, for some
                appropriate definition of token. This is the commonest
                case. Certain other parsing constructions might want
                addition state (a stack of enclosing productions, maybe).

                You should write the function so it examines the current
                state, decides if the production 'matches' or is 'valid'
                here. If it is valid, it should calculate the new state
                after matching, and return the pair (result,new
                state). If it is invalid, it should thrown the
                ParseFailure exception.

                For token lists, the new state will typically be the tail
                of the list after a certain number of items have been
                removed from the head.

                The module defines some useful combinators to build
                complex productions from simple ones.

                Based on, for example, Paulson's book.
                *)

  exception ParseFailure

(* sequencing:  P    Q
   Matches P, followed by Q
   Has result type (P*Q)
*)

  let (--) a b toks1 =
    let (aa,toks2) = a toks1 in
        let (bb,toks3) = b toks2 in
            ((aa,bb),toks3)

(* alternatives, P || Q
   Matches either P, or if that fails, then Q.
   P and Q must have the same result type.
   Note asymmetry: P has priority.
*)

  let (|||) a b toks1 =
    try a toks1
    with ParseFailure -> b toks1

(* post process: P >> f
   Matches just like P, but maps the result through f.
*)
  let (>>) a f toks1 =
    let (aa,toks2) = a toks1 in (f(aa),toks2)
```

```
(* generic infix operator parser.
   Matches a sequence prod (op prod)* in regexp notation
   and returns a list of the things that prod returns.
   However, the list is return as a pair (head,tail),
   to guarantee non emptiness.
   (whatever op returns is thrown away)
*)

  let rec infix op prod t = begin
  (prod -- op -- (infix op prod ||| (prod >> fun x -> x,[])))
      >> fun ((x,_),(xx,xxs)) -> (x,(xx::xxs))
  end t


(* Useful filters for dealing with the non empty lists returned
   by infix. (Non empty avoids explicit unit).
   *)

  let rec right_assoc f (x,xs) =
    match xs with [] -> x
      | xx::xxs -> f x (right_assoc f (xx,xxs))

  let rec left_assoc f (x,xs) =
    match xs with [] -> x
      | xx::xxs -> (left_assoc f ((f x xx),xxs))
end;;

(* The Types! *)
(* Types for ribbon proofs *)

type formula =
    Atom of string
  | Binary of formula * string * formula
  | Empty
  | Dots;;

type bunch =
    Formula of formula
  | Comma of bunch * bunch
  | Semicolon of bunch * bunch
;;


type oneribbonline = {r: int; f: formula;};;
type line = oneribbonline list;;

type boxpart = Line of line | Box of box
 and box = boxpart list;;

type splits =
    Unsplit of int
```

```
  | Split of int * (splits list list)
;;

type ribbonproof = {b: box};;

type proofpointer = int list;;

type rporerror = RP of ribbonproof * splits
        | RuleError;;

exception BunchStructureError;;

(* Functions operating on proofpointers *)
let rec pplowest p1 p2 =
  match p1,p2 with
      ([],_) -> p2
    | (_,[]) -> p1
    | (p1h :: p1t, p2h :: p2t) ->
          if (p1h > p2h) then p1
          else if (p1h < p2h) then p2
          else
              (p1h :: pplowest p1t p2t)
;;

(* Functions operating on ribbonproofs and boxes *)

let rec getboxpartbox p b =
  match p with
    [] -> Box b
  | hd :: tl ->
      match List.nth b hd with
        Box b2 -> getboxpartbox tl b2
      | Line l -> Line l (* error if not finished? *)

;;

let getboxpart p rp =
  getboxpartbox p rp.b;;

let rec getformulafromline r l =
  match l with hd::tl ->
      if hd.r=r then hd.f
      else getformulafromline r tl
;;

let getformula r p rp =
  match getboxpart p rp with
      Line l -> getformulafromline r l
;;

let rec lineContainsRibbon r l = match l with
```

```
    [] -> false
  | hd::tl -> if (hd.r=r) then true else lineContainsRibbon r tl
;;

(* Functions for building ribbon proofs *)

let rec sidejoin p q =
  match p,q with
      [],[] -> []
    | p,[] -> p
    | [],q -> q
    | (Line phl)::pt,(Line qhl)::qt ->
          (Line (phl@qhl))::(sidejoin_aux pt qt phl qhl)
and  sidejoin_aux p q pdef qdef =
  let empty_line l = List.map (fun {r=r;f=_} -> {r=r;f=Empty}) l
  in match p,q with
      [],[] -> []
    | p,[] -> List.map (fun (Line l) -> Line (l@(empty_line qdef))) p
    | [],q -> List.map (fun (Line l) -> Line ((empty_line pdef)@l)) q
    | (Line phl)::pt,(Line qhl)::qt ->
          (Line (phl@qhl))::(sidejoin_aux pt qt phl qhl)
;;

let rec bunchtoproof bch =
  bunchtoproof_aux bch 1 2
and bunchtoproof_aux bch n m =
  match bch with
      Formula f -> ([Line [{r=n;f=f}]],Unsplit n,m)
    | Semicolon (a,b) ->
          let (p1,w1,m1) = bunchtoproof_aux a n m
          in let (p2,w2,m2) = bunchtoproof_aux b n m1
          in (p1@p2,
              (match (w1,w2) with
                (Unsplit n,Unsplit m) (* when n=m ? *) -> Unsplit n
              | (Unsplit n,Split (m,l)) -> Split (n,([Unsplit n])::l)
              | (Split (n,l),Unsplit m) -> Split (n,l@[[Unsplit m]])
              | (Split (n,l1),Split (m,l2)) -> Split (n,l1@l2)
              ),
              m2)
    | Comma (a,b) ->
          let (p1,w1,m1) = bunchtoproof_aux a m (m+1)
          in let (p2,w2,m2) = bunchtoproof_aux b m1 (m1+1)
          in (sidejoin p1 p2,
              (match (w1,w2) with
                (Unsplit m1,Unsplit m2) -> Split (n,[[w1;w2]])
                    (* patterns below match lists with 1 elt only *)
              | (Unsplit m1,Split (m2,[l])) -> Split (n,[w1::l])
              | (Split (m1,[l]),Unsplit m2) -> Split (n,[l@[w2]])
                    (* default case *)
              | (_,_) -> Split (n,[[w1;w2]])),
              m2)
```

```
;;

(* Toolkit of assertions to build proof rules *)

(* Proof rp contains ribbon r at line p *)
let assertContainsRibbon r p rp =
  let Line l = getboxpart p rp in lineContainsRibbon r l
;;

let assertDots p rp=true;;
let assertRibbonSum r1 r2 r3 rp=true;;

let assertActualFormula r p rp=
  match getformula r p rp with
      Empty -> false
    | Dots -> false
    | Atom _ -> true
    | Binary _ -> true;;


let rec insert_at p b mapper=
  match b with bphd :: bptl ->
      match p with hd :: tl ->
          if (hd=1) then
              match bphd with
                  Line l -> (Line (mapper l)::b)
                | Box bb -> ((Box (insert_at tl bb mapper))::bptl)
          else
              bphd :: (insert_at ((hd-1)::tl) bptl mapper)
;;

let rec insert_after p b mapper=
  match b with bphd :: bptl ->
      match p with hd :: tl ->
          if (hd=0) then
              match bphd with
                  Line l -> (bphd :: Line (mapper l) :: bptl)
                | Box bb -> ((Box (insert_after tl bb mapper))::bptl)
          else
              bphd :: (insert_after ((hd-1)::tl) bptl mapper)
;;


let rec replacedots r f l =
  match l with
      [] -> []
    | hd::tl ->
          (if (hd.r = r) then
              ({r=r;f=f})
          else
              ({r=hd.r;f=Empty}))
```

```
          ::(replacedots r f tl)
;;


(* Ribbon proof rules, implemented as transformers from
   partial proof to partial proof *)

(*let andIntroduction r p1 p2 p3 rp =*)
(*  if*)
(*    (* Check p1, p2 and p3 all contain ribbon r *)*)
(*      assertContainsRibbon r p1 rp &*)
(*      assertContainsRibbon r p2 rp &*)
(*      assertContainsRibbon r p3 rp &*)
(*    (* Check p1,p2 accessible from p3 *)*)
(*    (* Check p3 is a Dots line *)*)
(*      assertDots p3 rp &*)
(*      assertActualFormula r p1 rp &*)
(*      assertActualFormula r p2 rp *)
(*  then let a=getformula r p1 rp and b=getformula r p2 rp in*)
(*     RP {b=(insert_at p3 rp.b (replacedots r (Binary(a,"n",b))))}*)
(*  else*)
(*      RuleError*)
(*;;*)

(*let starIntroduction r1 p1 r2 p2 r3 p3 rp =*)
(*  if*)
(*      assertContainsRibbon r1 p1 rp &*)
(*      assertContainsRibbon r2 p2 rp &*)
(*      assertContainsRibbon r3 p3 rp &*)

(*      assertRibbonSum r1 r2 r3 rp &*)

(*      assertDots p3 rp &*)
(*      assertActualFormula r1 p1 rp &*)
(*      assertActualFormula r2 p2 rp*)
(*  then let a=getformula r1 p1 rp and b=getformula r2 p2 rp in*)
(*     RP{b=(insert_at p3 rp.b (replacedots r3 (Binary(a,"*",b))))}*)
(*  else*)
(*      RuleError*)
(*;;*)

let nIntro r p1 p2 (rp,s) =
  if
    (* Check p1 and p2 contain ribbon r *)
      assertContainsRibbon r p1 rp &
      assertContainsRibbon r p2 rp &
    (* Check p1,p2 accessible from p3 *)
      (* Check p1,p2 both have actual formulae *)
      assertActualFormula r p1 rp &
      assertActualFormula r p2 rp
  then let a=getformula r p1 rp and b=getformula r p2 rp in
```

```
      RP ({b=(insert_after (pplowest p1 p2)
                    rp.b (replacedots r (Binary(a,"n",b))))} , s)
  else
      RuleError
;;


(* Parsing functions to build bunches and formulae from
   textual representations of them *)

open Parse;;

(* literal string match for operators *)
let op s = function
    Tokeniser.T2_Op t :: tail -> if (s=t) then (s,tail) else raise ParseFail
    | _ -> raise ParseFailure
;;

(* any string match for identifiers *)
let id = function
    Tokeniser.T2_ID t :: tail -> (t,tail)
    | _ -> raise ParseFailure
;;


(* First , we define the parser for formulae *)

(* Two utility maps to build Binary terms *)
let left_assoc_connective op =
 left_assoc (fun x y -> Binary(x,op,y))
;;
let right_assoc_connective op =
 right_assoc (fun x y -> Binary(x,op,y))
;;

(* The formula production , and its subproductions *)

let rec formula t = begin
  (op_exp"*"             >> left_assoc_connective "*")
||| (op_exp"&"             >> left_assoc_connective "&")
||| (op_exp"->"             >> right_assoc_connective "->")
||| (op_exp"-*"             >> right_assoc_connective "-*")
||| atomic_form
end t

and op_exp opn t = (infix (op opn) atomic_form) t

and atomic_form t = begin
   (id                              >> (fun x -> Atom x))
|||(( op"(" -- formula -- op")" ) >> (fun ((_,f),_) -> f))
end t
```

```
;;


(* Now bunches, very similar *)

let rec bunch t = (add_bunch ||| mult_bunch ||| atomic_bunch) t

and add_bunch t = begin
    (infix (op";") atomic_bunch) >> (left_assoc (fun a b -> Semicolon(a,b)))
end t
and mult_bunch t = begin
    (infix (op",") atomic_bunch) >> (left_assoc (fun a b -> Comma(a,b)))
end t
and atomic_bunch t = begin
    (formula                    >> (fun f -> Formula f))
||| ((op"(" -- bunch -- op ")") >> (fun ((_,b),_) -> b))
end t
;;

(* useful wrapper functions *)

let parse_formula s = match formula (Tokeniser.tokenise s) with
    (f,[]) -> f
  | _ -> raise ParseFailure;;

let parse_bunch s = match bunch (Tokeniser.tokenise s) with
    (b,[]) -> b
  | _ -> raise ParseFailure;;

let proof_bunch s = bunchtoproof (parse_bunch s);;

(* pretty print routines to convert proofs and formulae
   to ASCII representations *)

let rec formulalength f = match f with
    Atom s -> String.length s
  | Binary (a,o,b) -> (formulalength a)+(String.length o)+(formulalength b)+
  | Empty -> 0
  | Dots -> 3
;;

let rec formula_to_string f =
  match f with
      Atom a -> a
    | Binary (a,o,b) ->
          "(" ^
          (formula_to_string a) ^ o ^ (formula_to_string b)
          ^ ")"
    | Empty -> ""
    | Dots -> "..."
;;
```

```
let rec printpointer p =
  match p with
      [] -> ()
    | hd :: h2 :: tl -> print_int hd; print_string "."; printpointer (h2::tl
    | hd :: tl -> print_int hd; printpointer tl;;



let rec calc_ribbon_widths b =
  calc_rw_simple b []
and calc_rw_simple b w =
  match b with
      [] -> w
    | hd :: tl -> match hd with
          Line l -> calc_rw_simple tl (calc_rws_oneline l w)
        | Box bx -> calc_rw_simple tl (calc_rw_simple bx w)
and calc_rws_oneline l w =
  match l with
      [] -> w
    | {r=rib;f=form} :: tl ->
          let w2 =
            if (member_assoc rib w) then
                alter_assoc rib
                    (fun n -> max n ((formulalength form)+4))
                    w
            else
                (rib , (formulalength form)+4) :: w
          in calc_rws_oneline tl w2
                ;;

let rec calc_widths b s =
  let rw = calc_ribbon_widths b
  in let rw2 = calc_widths_p1_aux s rw
  in calc_widths_p2_aux s rw2
(* pass one makes sure each superribbon is as big as
   its widest division into subribbons *)
and calc_widths_p1_aux s rw =
  match s with
      Unsplit n -> [(n,get_assoc n rw)]
    | Split (n,l) ->
          let n_wid = get_assoc_def 0 n rw in
              let (listwidths,maxwidth) = calc_widths_p1_dolist l rw in
                  set_assoc (n, max n_wid maxwidth) listwidths
and calc_widths_p1_dolist l rw =
  match l with
      [] -> ([],0)
    | l :: tl ->
          let (lw,mw) = calc_widths_p1_dolist tl rw
          in let (rw_l) = (List.map (fun x -> calc_widths_p1_aux x rw) l)
```

```
          in let (n_l) = (List.map split_get_num l)
          in let w =
            max (List.fold_left (+) 0
                        (List.map2 (fun x y -> (List.assoc x y))
                                n_l rw_l))
                mw
          in
              ((List.fold_left
                      (assoc_merge (fun a b c -> max b c))
                      [] (lw::rw_l)),
                   w)
(* pass two pads out subribbons which aren't as wide as
   their superribbon *)
and calc_widths_p2_aux s rw =
  match s with
      Unsplit n -> rw
    | Split (n,l) ->
          let totalwidth = get_assoc n rw
          in
              (* iteratively apply calc_widths_p2_padlist to rw over
                 each element of l using fold_left *)
              List.fold_left
                  (fun rw l -> calc_widths_p2_padlist s l rw totalwidth)
                  rw l
and calc_widths_p2_padlist s l rw totalwidth =
  match l with
      [] -> rw
    | h :: t ->
          let w = calc_widths_p2_getlistwidth l rw (* width of ribs now *)
          in let n = List.length l (* number of ribs across this split *)
          in let space = totalwidth - w (* amount to of pad to add *)
          in let r = split_get_num h (* number of first ribbon *)
          in let thispad = (space/n) (* pad first by this much *)
          in let rw2 = (* change width of leftmost ribbon *)
            alter_assoc r (fun x -> x + thispad) rw
          in let rw3 = calc_widths_p2_aux h rw2 (* recurse on h *)
          in (* recurse on t *)
              calc_widths_p2_padlist s t rw3 (totalwidth-(get_assoc r rw3))
and calc_widths_p2_getlistwidth l rw =
  match l with
      [] -> 0
    | h :: t ->
          (get_assoc (split_get_num h) rw) +
              calc_widths_p2_getlistwidth t rw
and split_get_num spl =
   match spl with
     Unsplit n -> n
   | Split (n,_) -> n
;;
```

```
let rec newproof_string b s =
  let rw = calc_widths b s in nps_aux b rw
and nps_aux b rw  = match b with
    [] -> ""
  | Line l :: tl -> (nps_oneline l rw ) ^ "\n" ^ (nps_aux tl rw)
and nps_oneline l rw  = match l with
    [] -> ""
  | {r=ribnum;f=form} :: tl ->
        let wid = List.assoc ribnum rw
        in let lpad = (wid - formulalength form - 2)/2
        in let rpad = (wid - formulalength form - 2) - lpad
        in "[" ^ (nspaces lpad) ^
            (formula_to_string form) ^
            (nspaces rpad) ^ "]" ^
            nps_oneline tl rw
                ;;

let init_proof s t =
   let (box,splits,num) = proof_bunch s in
       print_string
           (newproof_string
                  (box @
                      [Line [{r=(split_get_num splits);f=Dots}];
                          Line [{r=split_get_num splits;f=parse_formula t}]
                  splits
                  )
;;


let print_bunchproof str =
  let (b,s,_) = proof_bunch str in print_string (newproof_string b s);;

let pprint (r,s) = print_string (newproof_string r.b s);;

let ppbunch str = let (b,s,_) = proof_bunch str in ({b=b},s);;

let pprpoe = function
    RP (r,s) -> pprint (r,s)
  | RuleError -> print_string "RuleError"
;;
```