Graphical Representation of Canonical Proof: Two case studies

Willem Bernard Heijltjes

Doctor of Philosophy
Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh
2011

Abstract

An interesting problem in proof theory is to find representations of proof that do not distinguish between proofs that are 'morally' the same. For many logics, the presentation of proofs in a traditional formalism, such as Gentzen's sequent calculus, introduces artificial syntactic structure called 'bureaucracy'; e.g., an arbitrary ordering of freely permutable inferences. A proof system that is free of bureaucracy is called *canonical* for a logic. In this dissertation two canonical proof systems are presented, for two logics: a notion of proof nets for additive linear logic with units, and 'classical proof forests', a graphical formalism for first-order classical logic.

Additive linear logic (or *sum–product logic*) is the fragment of linear logic consisting of linear implication between formulae constructed only from atomic formulae and the additive connectives and units. Up to an equational theory over proofs, the logic describes categories in which finite products and coproducts occur freely. A notion of proof nets for additive linear logic is presented, providing canonical graphical representations of the categorical morphisms and constituting a tractable decision procedure for this equational theory. From existing proof nets for additive linear logic without units by Hughes and Van Glabbeek (modified to include the units naively), canonical proof nets are obtained by a simple graph rewriting algorithm called *saturation*. Main technical contributions are the substantial correctness proof of the saturation algorithm, and a correctness criterion for saturated nets.

Classical proof forests are a canonical, graphical proof formalism for first-order classical logic. Related to Herbrand's Theorem and backtracking games in the style of Coquand, the forests assign witnessing information to quantifiers in a structurally minimal way, reducing a first-order sentence to a decidable propositional one. A similar formalism 'expansion tree proofs' was presented by Miller, but not given a method of composition. The present treatment adds a notion of cut, and investigates the possibility of composing forests via cut-elimination. Cut-reduction steps take the form of a rewrite relation that arises from the structure of the forests in a natural way. Yet reductions are intricate, and initially not well-behaved: from perfectly ordinary cuts, reduction may reach unnaturally configured cuts that may not be reduced. Cut-elimination is shown using a modified version of the rewrite relation, inspired by the game-theoretic interpretation of the forests, for which weak normalisation is shown, and strong normalisation is conjectured. In addition, by a more intricate argument, weak normalisation is also shown for the original reduction relation.

Acknowledgements

The past four years of my life have been defined by an opportune decision to apply for a position in Edinburgh, knowing nothing of the place except it was supposed to rain quite a lot, and by the—in my opinion, brave—decision by Alex Simpson to accept my application. I am deeply grateful to him for making these years enjoyable, productive, and possible. Alex provided me with the perfect topic for my dissertation, a hard combinatorial problem with a deep mathematical motivation—twice. Throughout this period, he has been a fantastic guide through the world of logic and computer science; I was the recipient of endless support, patience, and little yellow correction tags. Needless to say, without Alex I would not be where I am now—behind my desk, at 4am, making corrections to this document. But seriously, thank you.

I am also deeply thankful for all the good times I had with my colleagues and friends: Julian and Teresa, whose slow-cooker makes the most fabulous Colombian dishes; Fulvio and Micaela, true Romans; Lorenzo, who has never met a pun he didn't like; Grant, who is awesome; Giulia, who fully embraced the English language—and in particular the words 'cake' and 'ice-cream'; Ben, who, being the tallest man in the world, has a computer screen still large enough to hide behind; Gavin, our token Scotsman; Tom and John, who will hopefully explain Paris; Matteo, who always carries a little—actually, a significant—bit of Italy with him; Ohad the ultimate; Jeff, who is an Austrian, a Canadian, and a Scot, and judging by his capacity for holding alcohol, the disjoint union of all three; Rob, Peggy, and Harry; Chris, the brave Celtish warrior against bureaucracy; Miles, who now lives in Glasgow.

In addition, I would like to thank Robin Cockett, Roy Dyckhoff, Alessio Guglielmi, John Longley, Richard McKinley, Michel Parigot, Albert Visser, and Philip Wadler.

Last, and most of all, I would like to thank my wife Saskia, who joined me in this adventure, and promised to join me in the next. On y va!

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Willem Bernard Heijltjes*)

Table of Contents

Chapter 1

Canonical proof

1.1 Introduction

Proof theory is the study of formal proofs as mathematical objects. Modern proof theory has its roots in the introduction of two proof formalisms by Gerhard Gentzen in the 1930s ([40]), *natural deduction* and the *sequent calculus*. However, the representation of proof in these formalisms, in particular in the sequent calculus, is often not *canonical*: the formalism distinguishes between proofs that are ‘morally’ the same. The introduction of such artificial distinctions between proofs by a proof system was termed *bureaucracy* by Jean-Yves Girard. In the seminal paper [41] that introduced linear logic, Girard initiated a programme to eliminate bureaucracy from the new logic by finding geometric representations of proof, called *proof nets*.

The question of what constitutes bureaucracy in a proof formalism, of what are natural and what are artificial distinctions between proofs, is also the question of what is a good notion of *proof identity* for a logic: the question of when are proofs ‘morally’ the same. For many logics a notion of proof identity is clear from an established semantics. For others, most famously for classical logic, it is open to debate. Still, also in the absence of an established notion of proof identity, forms of bureaucracy can often be identified. The archetypical example of bureaucracy, also for classical logic, is that of two permutable inferences in the sequent calculus. The shape of a sequent proof, in which inferences form a tree, means that it is necessary to choose an order for two inferences, while the actual order in which the inferences are carried out may be inessential.

One example of a canonical proof system is natural deduction for negative intuitionistic logic—the fragment consisting of implication and conjunction. The normal forms of proofs in this formalism are free of bureaucracy, and also canonical from a semantic perspective: for a suitable notion of normal form, they correspond one–to–one with morphisms in free Cartesian closed categories (see e.g. [69]). Another example are Girard's proof nets for multiplicative linear logic without units [41]. These factor out precisely the bureaucracy of permutable inferences in the sequent calculus presentation of linear logic.

Three main reasons why canonical proof representations are interesting, are as follows. Firstly, a canonical proof formalism can be very informative of a logic. By eliminating bureaucracy, the intrinsic features of the logic itself become more prominent. Indeed, properties of the formalism cannot be attributed to bureaucracy, which is absent, and instead are likely to be inherent to the logic. For example, the non-confluence of proof reductions in the classical sequent calculus has in the past been attributed to the behaviour of the structural rules of contraction and weakening. However, in formalisms that bring these structural rules under control, such as the proof forests presented in Part II of this dissertation, reduction remains non-confluent. Thus it seems as if non-confluence may be an even more strongly intrinsic property of classical proof normalisation than previously thought. Secondly, canonical proof representations, such as proof nets for linear logic, hold the promise of unlocking the computational content of logics. The reasoning to support this idea will be expanded on later in this chapter, but briefly, it can be summarised as follows. In the computational interpretation of a logic, formulae correspond to types, proofs correspond to programs, and cut elimination corresponds to computation. If cut reduction is confluent, then the computation it embodies is deterministic, which in many cases means the proof system may be employed, more or less directly, as a language of computation. One of Girard's original motivations for proof nets was that they have confluent normalisation, suggesting the possibility of employing linear logic for computation. Thirdly, in many cases, a main reason for studying a logic is its semantics. For example, for both intuitionistic and linear logic the categorical semantics consists of categories with a natural, common structure, and models of (fragments of) these logics are ubiquitous throughout mathematics. In the presence of an accepted semantics, a notion of proof for a logic is canonical if it captures precisely the identifications made by the semantics. The canonical representation of mathematical structure is a useful tool in its investigation, and may be expected to enable efficient algorithms for its decision problems (such as term equality in categories). Examples of semantically canonical proof are intuitionistic natural deduction, Girard's proof nets for multiplicative linear logic, and the proof nets for multiplicative–additive linear logic of Hughes and Van Glabbeek [59]. Also, the proof nets presented in Part I of this dissertation are canonical for categories with finite products and coproducts.

This thesis investigates two canonical, graphical representations of proof, for two different logics. The first, presented in Part I, is a novel notion of proof net, for *additive linear logic*. This notion of proof net offers a canonical treatment of the two additive *units*, which have thus far not appeared in proof nets. The second, in Part II, is a canonical proof formalism for first-order classical logic called *classical proof forests*, for which cut-elimination is investigated.

The remainder of the present chapter will discuss the background and motivation of this work, starting with a quick exposition of the relevant general background in Section 1.2. This section mainly concerns the success story of intuitionistic natural deduction, which served as a template for a modern approach to linear logic and classical logic to which this thesis subscribes. Section 1.3 will discuss linear logic and proof nets, the background of the proof nets for additive linear logic presented in Part I, and summarise the results presented there. Section 1.4 will do the same for Part II, discussing the relevant background to classical proof forests and giving an overview of the results obtained for them.

This dissertation assumes some familiarity with classical logic and linear logic, and their presentation in the sequent calculus. Introductions to these can be found in [44] and [92]. In addition, a basic knowledge of category theory will be helpful, in particular, for Part I, acquaintance with category theory as far as the notion of limit and colimit. For an introduction, see [71].

1.2 Background

Proof theory, the study of formal proof, is considered one of the four pillars of mathematical logic, along with model theory, recursion theory, and set theory. The formalisation of mathematical reasoning began with Gottlob Frege, Bertrand Russell, and David Hilbert. The idea of regarding proofs as mathematical objects in their own right is usually attributed to the latter, as the basis of his famous program of proving the consistency of all of mathematics.

The foundations of modern proof theory were laid in the mid-1930s, when Gerhard Gentzen presented natural deduction and the sequent calculus [40]. Characteristic of these formalisms are the proof transformations they allow: cut-elimination, in the case of sequent calculus, described by Gentzen; and normalisation for natural deduction, described by Dag Prawitz in the 1960s [83]. [1] The key concepts of Gentzen's approach are the following.

Subformula property An inference rule has the *subformula property* if its premises are all subformulae of its conclusions. In the sequent calculus, and any well-behaved variant of it, the only rule that does not have the subformula property is the cut-rule. Then any cut-free proof contains only subformulae of the conclusion. As immediate consequences, the consistency of a cut-free calculus—that it cannot prove a contradiction—is easily established by an inspection of the rules. Also, proof search is strongly constrained in a calculus with the subformula property, in some cases to the point of being decidable, for instance for many propositional logics.

Cut-elimination The cut rule, pictured in a general form below, embodies composition, or transitivity of implication, and is a generalisation of modus ponens (from A and $A \to B$, conclude B).

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\,\text{Cut}$$

In a sequent calculus, *cut-elimination* is the process of removing instances of the cut-rule; the *cut-elimination property* is the property that cut-elimination can be carried out. As the calculus without cut is easily shown to be consistent (as was argued above), cut-elimination shows consistency of the calculus with cut. That the classical sequent calculus has the cut-elimination property was a main theorem (*Hauptsatz*) of Gentzen in [40].

The situation is analogous for normalisation in intuitionistic natural deduction, where normal proofs, which are the equivalent of cut-free proofs in the sequent calculus, have the subformula property.

The Curry–Howard correspondence

A landmark development at the end of the 1960s was the discovery, independently by William Howard and Nicolaas de Bruijn, of a close correspondence between on the one hand, proofs and formulae, and on the other, functional expressions in the λ-calculus

---

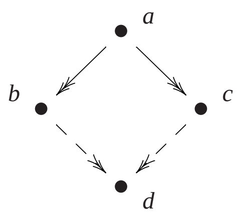
[1] Recently, drafts on normalisation for natural deduction by Gentzen have surfaced [94].

and their types [52], [30]. Now known as the Curry–Howard isomorphism—in recognition of similar connections for combinatoric logic and Hilbert-style deduction discovered by Haskell Curry [27]—or in its most general form as the mantra ‘proofs are programs’, this correspondence describes a link between logic and computation that is at the basis of modern type theory and functional programming. At its heart, the Curry–Howard isomorphism is the observation that β-reduction in the simply typed lambda calculus is essentially the same operation as normalisation in natural deduction for implication-only intuitionistic logic. Proofs and lambda terms are in a one–to–one correspondence, and normalisation steps in natural deduction corresponding to β-reduction steps in the lambda calculus.

Normalisation, and likewise, cut-elimination, is a relation between the proofs of a deductive system; from a given proof, multiple reduction steps may be possible. The following are central concepts describing reduction behaviour.

Weak/strong normalisation A reduction relation on proofs, such as normalisation in natural deduction or cut-elimination in the sequent calculus, is *weakly normalising* if some reduction paths reach a normal form, and *strongly normalising* if there are no infinite reduction paths, and all reduction paths eventually reach a normal form.

Confluence *Confluence* is the property that different reduction paths of a proof may always be extended to reach a common form. The confluence property is expressed in the diagram below, which states that if there are reduction paths from a to b and from a to c, then there must be reduction paths from b and from c to a common d. Note that in the diagram all arrows represent reduction paths, not individual reduction steps.



If a reduction relation is confluent and weakly normalising then every proof has a unique normal form—there may still be infinite reduction paths, unless also strong normalisation holds. In the 1960s Dag Prawitz put forward the idea of *proof identity by normality* (see e.g. [84]): the idea that unique normal forms are a natural notion of proof identity, in the sense that two proofs are the same if and only if they have the same normal form. In the view of proof reduction as computation, this is a generalisation of the idea that the meaning of a functional expression is the value it evaluates to.

In the 1970s the Curry–Howard correspondence was extended to category theory by Joachim Lambek, who showed that Cartesian closed categories are a semantics for intuitionistic natural deduction and the simply typed lambda calculus (see e.g. [69]). The categorical semantics identifies proofs if and only if they have the same normal form, and thus may be seen as a natural concretisation of the idea of proof identity by normality. There are technical subtleties: mainly, the categorical semantics equates proofs up to β-η normal form. The presence of disjunction adds significantly to the problem of rewriting to canonical representations of the natural semantics, bi-Cartesian closed categories. In addition to β- and η-equalities there are *commuting conversions*, and further semantic identities; obtaining canonical rewrites for these equations requires considerable ingenuity [70].

The sequent calculus

The sequent calculus introduces bureaucracy in the form of *permutations*, as follows.

Inferences in the sequent calculus operate on one or more formula occurrences in a *sequent*, a multiset of formulae, possibly separated into antecedents and consequents (sometimes a sequent is taken to be a list or even a set; throughout the thesis, it will be a multiset). When two consecutive inferences are applied to different formulae in a sequent, their order may often be exchanged; that is, they may be *permuted*. Permutations are pervasive in sequent calculi, and occur even in a sequent calculus presentation for conjunction–implication intuitionistic logic; a simple example is given below.

$$\dfrac{\dfrac{A,B\vdash C}{A,A\wedge B\vdash C}}{A\wedge B,A\wedge B\vdash C} \qquad \dfrac{\dfrac{A,B\vdash C}{A\wedge B,B\vdash C}}{A\wedge B,A\wedge B\vdash C}$$

An important observation is that, for this fragment of intuitionistic logic, permutations are factored out by the translation from sequent calculus into natural deduction. This was a main inspiration for Girard's idea of *proof nets* [41], further explored in Section 1.3. The idea of eliminating bureaucracy, and in particular the permutations of the sequent calculus, by moving to alternative, graphical representations of proof, is a central theme of this dissertation.

Generally, cut-elimination in the sequent calculus is non-confluent. Because this means that proofs have multiple normal forms, the idea of proof identity by normality does not apply directly. If the normal forms of proofs differ only by permutations, as is the case for example for multiplicative linear logic, then non-confluence is not a problem: a notion of proof identity can be based on equivalence classes of normal proofs under permutations. However, the picture is not always that clear: the normal forms of a proof may differ in other ways than by permutations, and different cut-elimination methods may produce different classes of normal forms. In such a case, it can be an interesting challenge to identify which equations between proofs are bureaucracy, and which constitute genuine differences.

The next two sections discuss the proof theory of two logics that are naturally expressed in the sequent calculus: linear logic, in Section 1.3, and classical logic, in Section 1.4.

1.3 Linear logic and proof nets

Linear logic was introduced by Jean-Yves Girard in the seminal [41]. It originated in an analysis of coherence spaces (see e.g. [44]), developed by Girard as a semantics of function evaluation in the lambda calculus. Linear logic is a refinement of both classical and intuitionistic logic, in the sense that both logics can be interpreted in linear logic by interpreting single classical or intuitionistic connectives as one or more linear connectives.

Syntactically, linear logic is naturally expressed in the sequent calculus, as displayed in Figure 1.1. The logic is divided into three fragments, called *additive*, *multiplicative* and *exponential*. The multiplicative connectives $(\otimes, \parr)$ are each other's dual under negation, $(-)^{\perp}$, as are the two *neutrals* $(\mathbf{1}, \perp)$, which are the *units* for the two connectives. Similarly, the additive connectives $(\&, \oplus)$ and their units $(\top, \mathbf{0})$ are duals, as are the two exponential *modalities* $(!, ?)$. (That, for example, $\mathbf{1}$ is a *unit* of the tensor $(\otimes)$ means that any formula A is canonically isomorphic to $\mathbf{1} \otimes A$ and to $A \otimes \mathbf{1}$.)

Linear logic has been a transformative influence in theoretical computer science, by being a rich source of ideas in general, and by bringing the following two important concepts within the domain of logic in particular.

Resource-consciousness In a proof of a linear implication $A \multimap B$ (or $A^{\perp} \parr B$) in linear logic, the assumption A must be used exactly once; this in contrast to the classical or intuitionistic implication $(A \to B)$ where the assumption may be used arbitrarily many times. In this and similar ways, linear logic is a logic of *resources*, where classical and intuitionistic logics describe *truth*.

	Conjunction	**Disjunction**
Multiplicatives	**Tensor ($\otimes$), One ($\mathbf{1}$)** $\dfrac{\vdash \Gamma, A \quad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}$ $\quad \dfrac{}{\vdash \mathbf{1}}$	**Par ($\parr$), Bot ($\perp$)** $\dfrac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}$ $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma, \perp}$
Additives	**With ($\&$), Top ($\top$)** $\dfrac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}$ $\quad \dfrac{}{\vdash \Gamma, \top}$	**Plus ($\oplus$), Zero (0)** $\dfrac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}$ $\quad \dfrac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}$
Exponentials	**Of course ($!$), Why not ($?$)** $\dfrac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}$ $\quad \dfrac{\vdash \Gamma, A}{\vdash \Gamma, ?A}$ $\quad \dfrac{\vdash \Gamma}{\vdash \Gamma, ?A}$ $\quad \dfrac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}$	
Axiom, Cut	$\dfrac{}{\vdash A, A^{\perp}}$ $\quad \dfrac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}$	

Figure 1.1: Linear logic as a one-sided sequent calculus

Concurrent computation Like classical logic, linear logic has an *involutive* (i.e. self-inverse) negation, which is handled in the sequent calculus presentation by allowing multiple conclusions in a sequent—intuitionistic sequent calculus, in the formulation by Gentzen [40], allows only one. Computationally, the presence of several conclusions may be interpreted as multiple computations that are processed simultaneously, and that may interact. This way, at least in theory, linear logic provides an account of *concurrent* or *parallel* computation. Explorations of the connection between linear logic and concurrency are found, among others, in [1] and [12], and also the recent [22]; an overview is given in [21].

One branch of research on linear logic, and of that inspired by it, has focused on exploring these computational aspects. In particular the resource-consciousness of linear logic was quickly adopted by the functional programming community, in the form of *linear types* [95]. Recently, the intuitionistic variant of linear logic, which allows only single-conclusion sequents, thereby emphasising resource-consciousness over concurrency, has been used to enrich the lambda calculus with a refined theory of computational effects [35].

Of the research into linear logic itself, and its semantics, there are three main threads that are relevant to the present discussion. One is that of the categorical semantics of linear logic, which will be briefly touched on below. A second is that into game-theoretic semantics, which may be seen as investigating the computational side of linear logic via an alternate, more semantically oriented route than the sequent calculus. The other direction is the search for proof nets: canonical, geometric proof formalisms, intended as an alternative syntax to the sequent calculus.

Categorical semantics of linear logic

Soon after linear logic was introduced, it was noted by Robert Seely in [86] that a natural categorical semantics for linear logic is as follows: the multiplicative fragment is modelled by $*$-autonomous categories (see also [10]), in which the additives correspond to products and coproducts, and the exponentials form a (co)monad structure with additional properties (the modern formulation [14] requires a *monoidal* (co)monad). An alternative formulation of $*$-autonomous categories was the result of an investigation into a reasonable notion of *linearity* in categories by Robin Cockett and Robert Seely [24].

These categorical models identify proofs under cut-elimination, providing a notion of proof identity in the tradition of proof identity by normality. They also identify proofs under permutations, and other, similar equations—many of which are forced by the identification of proofs under cut-elimination. In these models the following are essential concepts.

Composition via cut-elimination Composition of morphisms is an essential, basic operation in category theory, producing a morphism $g \circ f : A \to C$ from morphisms $f : A \to B$ and $g : B \to C$. To similarly compose two proofs in the sequent calculus, a cut may be used.

$$\frac{A \vdash B \qquad B \vdash C}{A \vdash C}\,\text{Cut}$$

If a categorical model identifies proofs under cut-elimination, it is natural to use only normal (i.e. cut-free) proofs as representations of morphisms. Then the cut used to compose two proofs must be eliminated; this is the idea of *composition via cut-elimination*.

Associative composition The basic laws of category theory are that composition is associative and has identity morphisms as (left and right) units. For a category where morphisms are represented by the normal forms of proofs, and composition is implemented as cut-elimination, associativity of composition is implied by confluence of cut-elimination. This is easily seen: the two ways of applying two compositions correspond to the two ways in which two cuts may be eliminated in order; by confluence, these must yield the same result. (However, confluence is not a necessary condition for associativity of composition to hold.)

Free categorical models If a logic has categorical models with a certain structure, a *term model* may be constructed by taking as objects the formulae in the logic, and as morphisms the equivalence classes of proofs under the laws associated with the categorical structure. In such a categorical model, the given categorical structure occurs *freely*. (A relevant example is how additive linear logic forms a category with free finite products and coproducts, discussed in Chapter 2.)

Full completeness A categorical model of a logic is *fully complete* if every morphism is the denotation of some proof. This is equivalent to the functor from the free category of the logic, into the model, being full (surjective on morphisms). The concept of full completeness—the term was coined in [3]—is a natural strengthening of the traditional proof-theoretic notion of completeness, which requires that if a formula is true in the model, it must have a proof in the syntax.

The semantics of a logic is usually a main reason for which the logic is studied. The categorical models of linear logic have structure that is basic, and common throughout mathematics—and even physics. One branch of research into linear logic is the search for natural models of linear logic, that are as close as possible to the free model. Full completeness is one measure of how close a model is—crudely, a fully complete model is a quotient of the free model.

One relevant series of investigations into characterising, and finding natural examples of, the categorical semantics of linear logic, are the works of André Joyal and Hongde Hu in the late 1990s. Building on a modification of Girard's coherence spaces, the original semantics of linear logic, by Thomas Ehrhard in [36], and following up on the work by Joyal on free bicompletions [63], categories with free limits and colimits, they connect the categorical approach and coherence space semantics, in [55] and [54]. This led to a coherence space model of the additive fragment, without the units, that is equivalent to the free categorical model, by Hongde Hu in [53]. A fully complete model of the multiplicative fragment, also without units, is presented in [31]. Finally, a fully complete coherence space model for the combined multiplicative–additive fragment is given by Richard Blute, Masahiro Hamano and Philip Scott, in [18].

Another route towards categorical models for linear logic is via game theory. This will be discussed next.

Game semantics of linear logic

A rich branch of investigation into the computational content of linear logic is that into its game-theoretic semantics, initiated by Andreas Blass [15] and Yves Lafont and Thomas Streicher [66]. In an informal view of the game interpretation, a formula describes a game between two players, Player and Opponent, while a proof is a winning strategy for Player. The additive connectives are interpreted as a binary choice for the Player (for the coproduct) or the Opponent (for the product). The multiplicatives encode two games played in parallel, where either Player (in the coproduct) or Opponent (in the product) may switch between the two games (*schedule*), while the other is forced to continue play in the currently active game. The four neutrals are winning positions, the additive units of a global kind, and the multiplicative units of a local kind. The exponential modalities (?) and (!) allow Player and Opponent, respectively, to *backtrack*: to return to an earlier position to make a new choice, in addition to the earlier one.

In the early and mid-1990s, research into formalising these ideas led to solutions to the long-standing problem of finding a good semantics for PCF, the *Programming language of Computable Functions*. These results were obtained independently by two traditions of linear logic games, each building on their respective formulation of games for the multiplicave fragment [3],[61]. One tradition is that of Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria [4] (see also [8]), the other that of Martin Hyland and Luke Ong [62]—while ideas similar to those of the latter tradition were independently put forward in the work of Hanno Nickau [81].

The above games are all *sequential*: strategies prescribe a fixed order of moves. This is fine for the multiplicative and exponential fragments, but as is discussed in [2], for the additive fragment sequential games suffer from much the same problem as the sequent calculus: composition is not associative. One possible way around this problem is to incorporate concurrency in games, as pioneered by Samson Abramsky and Paul-André Melliès in [5], where a fully complete games model for multiplicative–additive linear logic is presented. This line of research was continued by Paul-André Melliès in [77] and [75], eventually leading to a fully complete games model for full propositional linear logic in [76]. These games are *alternating*, meaning that Player's and Opponent's turns alternate. This allows an *interleaving* approach to concurrency, which represents a concurrent computation by the collection of its possible execution orders. A remaining challenge in game semantics for linear logic is to move away from alternating games, towards a game-semantic treatment in the spirit of *true concurrency*, where concurrency is inherent to the formalism [78], [37].

Proof nets

Proof nets, graphical representations of linear logic proofs, were introduced by Girard alongside linear logic, in [41]. These original proof nets, now known as MLL-nets, were canonical for the multiplicative fragment without units, factoring out permutations. But the potential of the idea was clear: proof nets would be a geometric description of morphisms in the free categorical model of linear logic, combining the best properties of syntax—e.g. the ability to do computation—and semantics—being directly amenable to mathematical analysis of its structure. (That the natural idea of finding proof nets to eliminate bureaucracy, coincided with a finding a syntactic description of the free categorical model, was pointed out by Richard Blute in [16].)

An example MLL-net is displayed in Figure 1.2, along with two sequent proofs that it is a translation of—and that are identical up to permutations. Of the structure of a sequent proof, a MLL-net retains just the axioms, as *axiom links*, connections between

$$\dfrac{\dfrac{\dfrac{}{\vdash B^\perp,B} \qquad \dfrac{\dfrac{}{\vdash A,A^\perp} \quad \dfrac{}{\vdash C^\perp,C}}{\vdash A,C^\perp,C\otimes A^\perp}}{\vdash A\otimes B^\perp,B,C^\perp,C\otimes A^\perp}}{\vdash A\otimes B^\perp,B\parr C^\perp,C\otimes A^\perp} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash A,A^\perp} \quad \dfrac{}{\vdash B^\perp,B}}{\vdash A\otimes B^\perp,B,A^\perp} \qquad \dfrac{}{\vdash C^\perp,C}}{\vdash A\otimes B^\perp,B,C^\perp,C\otimes A^\perp}}{\vdash A\otimes B^\perp,B\parr C^\perp,C\otimes A^\perp}$$

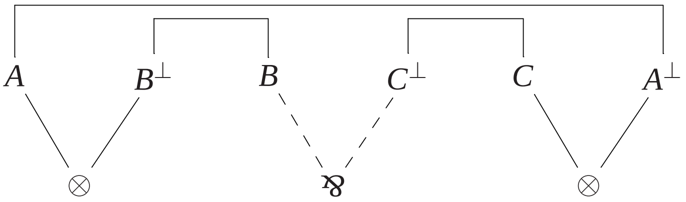


Figure 1.2: An example MLL-net

the leaves of the formula trees of the conclusion sequent. Not every configuration of formula trees connected by axiom links, called a *proof structure*, corresponds to a sequent proof. The following are therefore central components to the theory of MLL-nets—and any other notion of proof net.

Correctness criteria A *correctness criterion* is a property that distinguishes the proof nets from the proof structures. By their nature, different correctness criteria for a notion of proof net must be equivalent. Nevertheless, different formulations are useful in different ways, and for a notion of proof net to have multiple correctness criteria, as is the case with MLL-nets, can be instructive. A correctness criterion is generally expected to be intrinsic to the formalism, i.e. defined on the structure of the proof net itself. Thus the property of being the translation of a sequent proof is not usually considered a reasonable correctness criterion.

Sequentialisation *Sequentialisation* is the term for the reverse translation from proof nets to sequent proofs; it may be used to indicate the translation algorithm itself, or the property that one exists. While the translation from proofs to proof nets is usually a straightforward induction on the structure of a proof, the property of sequentialisation is closely related to correctness criteria, and requires a deep analysis of the structure of the proof nets. Commonly, sequentialisation is formalised as an algorithm on proof structures, that produces a sequent proof if the structure is a proof net, and fails otherwise—in that way constituting a correctness criterion.

Correctness criteria and sequentialisation for MLL-nets have been a subject of study in their own right. The most well-known correctness criterion for MLL-nets is that of Vincent Danos and Laurent Regnier [29]. It states that a proof structure is a proof net if and only if for every *switching*, which is a choice of deleting exactly one of the two (dashed) links of every par-vertex ($\parr$), the remaining graph is a tree (acyclic and connected). Although the time complexity of this algorithm is exponential, correctness of MLL-nets can be decided in linear time [46]. The paper [13] introduced the notion of *kingdom*, a notion of subnet corresponding directly to subproofs in the sequent calculus—to be precise: corresponding to smallest subproofs under permutations. A recent study, [32], presented an approach to sequentialisation using *jumps*, a relation on the structure of a proof net that, wholly or partially, reflects the ordering of inferences in a sequent proof translation of the net.

The amount of effort it has taken to reach the current level of understanding of MLL-nets underlines how proof nets are not an easy subject, and to extend MLL-nets to larger fragments of linear logic has proven exceedingly difficult. Successive proposals for a good syntax for the full multiplicative fragment, including the multiplicative units, are [17] and [65] in the late 1990s, and more recently [90] and [57]. These approaches all have good properties, but none is truly canonical, in the sense that none provides a geometric description of the free categorical models of multiplicative linear logic, free $*$-autonomous categories.

In another direction, several notions of proof net have been suggested for the combined multiplicative–additive fragment, without the units. After partial results in [43] a notion of proof net was presented by Dominic Hughes and Rob van Glabbeek, in [59], that is canonical for the categorical semantics for the multiplicative–additive fragment: *semi $*$-autonomous* categories with binary products and coproducts.

Proof nets for additive linear logic

In Part I of this dissertation a new notion of proof net is presented, for additive linear logic, the fragment of sequents $A \vdash B$ where A and B are additive formulae, constructed from atomic propositions, the additive connectives $(\&, \oplus)$, and their units $(\mathbf{0}, \top)$. The categorical semantics of additive linear logic is that of categories with finite products and coproducts—hence the logic is also known as sum–product logic. The proof nets are canonical for this semantics.

First, in Chapter 2, existing nets for additive linear logic without units, a fragment of the multiplicative–additive nets in [59], are adapted to incorporate the units in a way that is simple, but not canonical, forming a notion of *sum–product nets*. The categorical equations over the units force an equational theory over sum–product nets, which is then decided by rewriting to canonical forms called *saturated nets*, using a simple rewrite relation called *saturation*, presented in Chapter 3. To complete the theory of saturated nets, it is shown how they form a syntactic characterisation of the categorical term models of additive linear logic, namely categories with free, finite products and coproducts. The results include a direct notion of composition for saturated nets and, importantly, a correctness criterion and a sequentialisation algorithm.

A main technical contribution of this work is the proof, in Chapter 4, that the saturation relation is correct, i.e. that saturated nets are indeed canonical. Of the several issues confronted in this proof, an important example is that in Figure 4.5 on page 94.

1.4 Classical logic

For classical logic there are fundamental obstacles to finding both computational meaning and decent notions of proof identity. The discussion will first cover the situation for propositional classical logic, and consider first-order logic later.

A first problem for finding a good notion of proof identity for propositional classical logic is that cut-elimination in the sequent calculus, the traditional home of classical proof, is highly non-confluent. In particular the so-called Lafont example (see [44, Appendix B]), in Figure 1.3, shows that (under mild assumptions) a cut on two weakened formulae forces any two proofs of the same sequent to be identified. A further obstacle is what is sometimes called Joyal's theorem—or even Joyal's paradox, more for its undesirability than for any mathematical paradoxicality—the observation that a Cartesian closed category with an involutive negation collapses into a preorder (see e.g. [69, Section 1.8] or [42, Appendix B]). What this means is that if intuitionistic proof, whose semantics is that of cartesian closed categories, is equipped with a classical, involutive negation in the form of an isomorphism $A \cong \neg\neg A$, then any two proofs of the same formula are identified.

Irrespective of these problems, there are several consistent proposals for what constitutes proof identity in classical logic. However, the overall picture is one of multiple competing notions of proof identity. Below, an overview will be given of several prominent such proposals. Each of these approaches to categorical semantics is based on relaxing some of the assumptions leading to Joyal's theorem; that is, dropping one part of the structure of Cartesian closed categories with involutive negation.

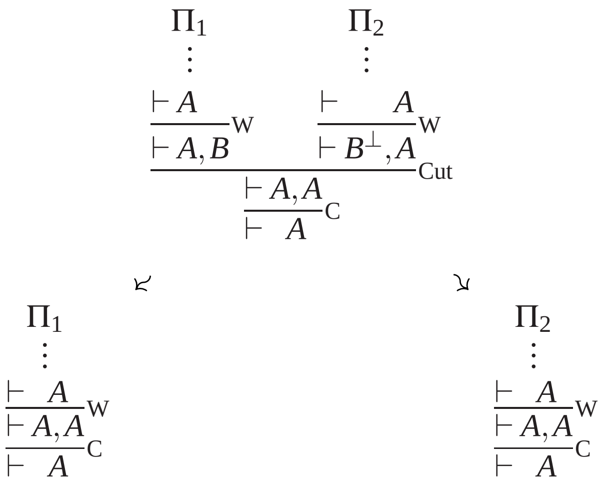


Figure 1.3: The Lafont example

Relax involutive negation The formulation of classical proof in natural deduction allows good computational interpretations of classical logic. This is exemplified by Michel Parigot's $\lambda\mu$-calculus [82], which has a categorical semantics in Peter Selinger's control categories [87]. In classical natural deduction negation is not involutive: the classical principle $\neg\neg A \Rightarrow A$, which may or may not appear directly as an inference rule, is not an isomorphism. Formulations of this principle as a proof construct have a computational interpretation as control operators for continuations [45], which allows a computational semantics in the form of an abstract machine [91]. A related approach to the use of classical natural deduction is the interpretation of classical logic in intuitionistic logic, by a *double negation translation* (corresponding, computationally, to a translation into continuation-passing style). Since the early formalisations of intuitionistic logic, different such translations have been found by Kolmogorov, Gödel, Gentzen, Kuroda, and Krivine, among others (for a comparison and further references, see [38]). This is also the route taken by Girard's LC [42].

Relax bi-Cartesian structure Decent categorical models of classical logic can be obtained starting from $*$-autonomous categories, the categorical semantics of multiplicative linear logic, rather than Cartesian closed categories. Several such approaches are outlined below, that differ in the precise choice of categorical identities extending the $*$-autonomous structure. What most have in common, is that negation is involutive, but conjunction and disjunction are modelled by (dual) monoidal products, rather than by Cartesian products and coproducts. A consequence of relaxing the Cartesian structure is that models are no longer Cartesian closed. One approach along these lines are the Boolean categories by François Lamarche and Lutz Straßburger in [67], continued by Straßburger in [88] and [89]. Also, non-trivial categorical models of classical proof are obtained by Carsten Führmann and David Pym in [39] by taking sequent calculus proofs as morphisms, on which cut-elimination imposes an ordering on proofs, rather than forcing their identification. This model was extended (from propositional logic) to first-order logic in Richard McKinley's Ph.D. thesis [72]. Further approaches are Martin Hyland's categorical proof invariants based on compact closed categories, in [60], and the categorical and polycategorical models in [11].

Relax Cartesian closure A third approach to categorical models of classical proof maintains the bi-Cartesian structure of conjunction and disjunction, as well as the involutive negation, but relaxes Cartesian closure. This is the approach taken in [34], where a notion of proof identity is proposed based on bi-Cartesian categories with additional structure. These categories are also models for additive linear logic, and the syntax underlying these categories is are proof nets for additive linear logic without units [33]. (These nets are the unit-free fragment of the proof nets presented in Part I of the dissertation.)

Syntactic approaches

In addition to the semantic, categorical approach, there is a rich and inventive field of syntactic approaches to classical logic. Firstly, cut-elimination for (variants of) the classical sequent calculus, and in particular reduction relations that are strongly normalising, are of significant computational interest and continue to be studied (see e.g. [9], [7], [93], and [51]). Secondly, there is the proof formalism called *deep inference*, which allows proof transformations on subformulae in a style reminiscent of term rewriting, and which has interesting normalisation properties (see e.g. [19] and [47]). Thirdly, several graphical representations of proof have been proposed for classical logic. Proof nets in the style of Girard's MLL-nets are discussed in [85] and [73], which treat contraction as a connective, duplicating parts of a formula tree; and in [68], which explores proof nets that consist solely of formula trees and axiom links. A different graphical approach is the celebrated [58] by Dominic Hughes, presenting a notion of proof that consists purely of functions between graphs.

Classical proof forests

In the above it was discussed how propositional classical proof has no non-trivial, generally agreed upon semantics; and that finding a good syntax for it is not an easy task. For first-order classical logic, these issues may be expected to be worse. In addition to the propositional fragment, it includes the first-order proof content associated with quantifiers: *eigenvariables* to instantiate universally quantified variables, and the assignment of *witnessing terms* for existentially quantified variables.

However, it is possible to give an account of first-order classical proof that simply ignores propositional proof. This is a consequence of Herbrand's Theorem [50], which separates first-order and propositional proof content, plus the fact that propositional classical logic is decidable. An idea for a semantics of first-order classical proof is then as follows: taking first-order proof content as primary, the meaning of a proof is found in the assignment of witnessing information to quantified variables, while propositional content is ignored (not unreasonably given decidability). The proposal offers the possibility of a non-trivial semantics of first-order classical proof (even though the restriction to the propositional fragment would be trivial).

Part II of this dissertation attempts to carry out this programme.[2] It investigates a representation of first-order classical proof called *classical proof forests*, introduced in Chapter 5. A proof forest is a proof for a sequent of first-order formulae (for simplicity) in prenex-normal form. It consists of a forest structure, with a tree for each formula, that records witness assignments to universally and existentially quantified variables. The trees branch out only at vertices representing existential quantifiers; propositional formulae are represented by the leaves, which are evaluated by a tautology check. A partial order called the *dependency* records when a choice of witnesses depends on a witness assignment elsewhere in the proof forest. By allowing this dependency to be a partial order, classical proof forests factor out the permutations of the sequent calculus, whose inferences are arranged in a tree-ordering. In that way, classical proof forests are canonical for first-order classical proof.

A similar formalism to classical proof forests has been considered before by Dale Miller [79], called *expansion tree proofs*, as an economic representation of higher-order classical proof. Also, classical proof forests admit a natural game-theoretic interpre-

---

[2] The idea of carrying out such a programme has apparently occurred independently to several people. The technical ideas in the form pursued in this thesis were first investigated in by Alex Simpson in the early 2000's. Martin Hyland has told us that he has also looked at very similar ideas himself. Also, Richard McKinley independently began a closely related programme of investigation, which is discussed in more detail below.

tation, in the style of the game semantics for classical arithmetic of Thierry Coquand [26]. In this interpretation, a proof forest is a strategy for ∃loise in a *two-player backtracking game* against her opponent ∀belard. The witness assignments to quantifiers in a proof forest represent the moves by both players, who take turns selecting values from a given domain. Branching on existential quantifiers represents backtracking by ∃loise. Different from Coquand's games, which are sequential, a proof forest does not necessarily prescribe a fixed order of moves; rather, the strategy supports any order of play that respects the dependency ordering.

The present treatment of classical proof forests is an investigation into composition via cut-elimination. The economic structure of proof forests, its natural game-theoretic semantics, and the fact that they are canonical for the sequent calculus, raised the hope that cut-reduction might be well-behaved. Unfortunately, or perhaps interestingly, this has not turned out to be the case, at least not initially. While the design of the cut-reduction steps, in Chapter 6, follows naturally from the structure of the proof forests, reductions are very badly behaved. Starting from a perfectly acceptable configuration dubbed the 'universal counterexample', displayed in Figure 6.3 on page 167, reductions produce unnaturally configured cuts that are impossible to reduce, and exhibit cyclic reduction traces. However, partially inspired by the game semantics, solutions are found to both problems. For a modified reduction relation that implements these solutions, weak normalisation is proven, and strong normalisation is conjectured.

The treatment of classical proof forests is continued, in Chapter 7, by an exploration of the differences between reduction in proof forests and in the sequent calculus. By avoiding reduction steps that leave the image of the translation from the sequent calculus, the original reduction relation on proof forests is shown to be weakly normalising, too. Several further, interesting modifications to the reduction relations are discussed informally, including a comparison with a closely related formalism called Herbrand nets, by Richard McKinley [74]—see below. Finally, while reduction in proof forests is weakly normalising, and plausibly even strongly so, it is not confluent. An evaluation of non-confluence in the different reduction relations and strategies—where, again, the universal counterexample is central—concludes the exposition on proof forests.

Herbrand nets

The research on classical proof forests was conducted concurrently with, and initially independently of, a similar investigation by Richard McKinley, originating in his investigation of order-enriched categorical models of first-order classical proof [72]. After becoming aware of each other's work, a fruitful exchange of ideas and results followed, leading to many possible directions for continuing research. The investigation into classical proof forests was influenced mainly by the game semantics, viewing the divergence with the sequent calculus as an interesting opportunity. The direction taken by McKinley was to place additional structure on proof forests in order to obtain a closer correspondence with the sequent calculus, resulting in the *Herbrand nets* presented in [74]. The main structural difference between Herbrand nets and classical proof forests is that unlike the latter, Herbrand nets have a form of *axiom links* corresponding to the axiom rule of the sequent calculus, and are in that way more closely related to proof nets for MLL with quantifiers (see e.g. [13]). However, in a detailed comparison of the two formalisms, in Section 7.3, it will emerge that the differences between classical proof forests and Herbrand nets are quite superficial. At the same time, there is a strong common theme, in the form of the basic forest structure with a dependency ordering that is shared by classical proof forests and Herbrand nets. Indeed, it is perhaps more accurate to view the two approaches as variants of essentially the same approach to first-order classical proof, than as completely distinct formalisms.

Throughout Part II of this dissertation contributions by McKinley are carefully identified and attributed.

1.5 Synopsis

As discussed, this thesis contributes to two separate, but connected investigations into canonical proof. The structure of the dissertation is as follows.

Part I treats proof nets for additive linear logic. In this part, Chapter 2 introduces additive linear logic, its semantics of sum–product categories, and its sequent calculus presentation, and presents the (non-canonical) notion of sum–product nets and their equational theory. Chapter 3 presents the saturation procedure and the (canonical) saturated nets, discusses identity and composition in the category of saturated nets, and describes the correctness condition for saturated nets. Chapter 4 covers the proof that the decision procedure for term equality in free sum–product categories based on saturation is sound.

Part II treats classical proof forests. They are presented in Chapter 5, which includes a game-theoretic interpretation and a comparison to the sequent calculus. Chapter 6 introduces a cut-reduction procedure, illustrates how it is badly behaved, and suggests modifications, resulting in a weak normalisation theorem (and a strong normalisation conjecture) for the modified reduction relation. Chapter 7 gives a weak normalisation result for the original reduction relation, discusses other variations on it, and illustrates how different variants of proof forest reduction are non-confluent.

Chapter 8 summarises the results in the thesis and suggests angles for future work. Technically, this chapter does not belong to Part II; however, this is obscured by the fact that the LaTeX command `\end{part}` has no visible effect.

Part I

Proof nets for additive linear logic

Chapter 2

Sum–product nets

2.1 Introduction

Chapters 2, 3, and 4 will present a notion of proof nets for additive linear logic, the fragment of linear logic consisting of linear implication between strictly additive formulae. As the principal account of semantics for this fragment is given by categories with finite products and coproducts it is also known as sum–product logic. The proof nets presented here are canonical for this semantics: there is a one-to-one correspondence between proof nets and morphisms in a free sum–product category.

The motivation for investigating proof nets for this logic is threefold. Firstly, additive linear logic is of independent interest because of its categorical semantics. A free sum–product category is the free completion with products and coproducts of a base category $\mathcal{C}$. As such, free sum–product categories are a restriction of Joyal's free bicomplete categories [63], which are completions with all limits and colimits, to the (finite) discrete case. Also the game-theoretic semantics of additive linear logic, explored in [64] and [2] among others, makes it an interesting subject of study; but this will not be investigated further here.

A second, more specific motivation is that additive linear logic is a fragment of the Enriched Effect Calculus by Jeff Egger, Rasmus Møgelberg and Alex Simpson [35], a type theory for computation with effects, based on intuitionistic linear logic. It was suggested by Alex Simpson that the free sum–product completion of the empty category is a model for this calculus, and may possibly be a complete model—this question, however, has not yet been resolved.

Thirdly, while additive linear logic is a relatively simple fragment of linear logic, the treatment of the units, or neutral elements, in proof nets for linear logic is notoriously difficult. In addition, the the fragment includes much of the complexity of the full multiplicative–additive fragment, since the multiplicative connectives are present in a restricted form at the meta-level: as linear implication and composition (or cut). A notion of proof nets for this fragment is thus an important contribution to investigations into the proof-net problem for larger fragments of linear logic. The relatively simple nature of additive linear logic, and the simplicity of its proof nets in the absence of the units, make it an ideal setting for exploring the properties of the additive units, which have thus far not appeared in proof nets. To quote Girard, in [43, Appendix A.3]:

> There is still no satisfactory approach to additive neutrals [. . .].[1] The only way of handling $\top$ is by means of a box or, if one prefers, by means of a second order translation: on this Kamtchatka of linear logic, the old problems of sequent calculus are not fixed. The absence of a satisfactory treatment of $\top$ calls for another notion of proof-net. . .

Another quote is from Dominic Hughes in [56, Section 1], where he presents additive proof nets without the units:

> Work in progress aims to extend the approach presented here to units (i.e., initial and final objects), and to an arbitrary base category (rather than a set of atoms, i.e., discrete category). The former, if at all feasible, appears to be quite involved. This is evidenced by the fact that, when empty products and sums are present, there is no obvious confluent and terminating rewrite system for the cut-free proofs (or proof terms) of Cockett and Seely's deductive system.[2] If such a rewrite system can be found, it might provide useful clues towards extending the approach presented in this paper to the initial and final objects, yielding a canonical graphical syntax for finite products and sums.

The last sentence of the above quote describes what is presented in these chapters: a canonical graphical syntax for finite categorical products and coproducts. After discussing background material, below, first sum–product categories and additive linear logic will be discussed, in Section 2.2. In Sections 2.3 and 2.4 a notion of proof nets, based on existing nets without units [59], will be described. These nets are not canonical for sum–product categories; in Section 2.5 an equational theory over nets is defined, that equates nets that represent the same categorical morphism.

The next chapter, Chapter 3 will present a simple rewriting algorithm called *saturation*, that, from sum–product nets, obtains canonical normal forms called *saturated*

---

[1] The original text reads, "...which are fortunately extremely uninteresting in practice." One can only guess at the reasons for questioning the significance of the additive units; after all, they are an integral part of linear logic, and in the opinion of the author, and presumably in that of others who have worked on them, pose a demanding challenge with interesting technical consequences.

[2] This refers to [25].

nets. Chapter 4 will be devoted to the proofs underlying the canonicity result. Most of the results in this part of the dissertation appeared in [49] (included as an appendix). A new result, not presented in that paper, is the correctness condition for saturated nets, in Section 3.4. Also the soundness proof, in Chapter 4, has not yet appeared in print (though it has accompanied [49] as an appendix in the peer review process).

2.2 Sum–product categories and additive linear logic

First, recall the definitions of categorical products and coproducts. The *(binary) product* $A \times B$ of two objects A and B comes with *projections* $\pi_0 : A \times B \to A$ and $\pi_1 : A \times B \to B$, and for every $f : X \to A$ and $g : X \to B$ a unique *product map* or *pairing* $\langle f,g\rangle : X \to A \times B$ such that $\pi_0 \circ \langle f,g\rangle = f$ and $\pi_1 \circ \langle f,g\rangle = g$. Dually, the *(binary) coproduct* $A+B$ of objects A and B has two *injections* $\iota_0 : A \to A+B$ and $\iota_1 : B \to A+B$, and for every two maps $f : A \to X$ and $g : B \to X$ a unique *coproduct map* or *co-pairing* $[f,g] : A+B \to X$ such that $[f,g] \circ \iota_0 = f$ and $[f,g] \circ \iota_1 = g$. The equations in the above definitions are expressed by the following commuting diagrams.

$$\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, dashed, "\langle f,g\rangle"'] \arrow[dr, "g"] & \\
A & A \times B \arrow[l, "\pi_0"] \arrow[r, "\pi_1"'] & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "\iota_0"] \arrow[dr, "f"'] & A+B \arrow[d, dashed, "{[f,g]}"'] & B \arrow[l, "\iota_1"'] \arrow[dl, "g"] \\
 & X &
\end{tikzcd}$$

Equivalently, the uniqueness requirement for pairing and copairing may be replaced by the following equations, for maps $f : X \to A \times B$ and $g : A+B \to X$.

$$f = \langle \pi_0 \circ f,\ \pi_1 \circ f\rangle \qquad\qquad g = [g \circ \iota_0,\ g \circ \iota_1]$$

The *terminal object* or *nullary product* $\mathbf{1}$ has a unique *terminal map* $!_X : X \to \mathbf{1}$ out of every object X, while the *initial object* or *nullary product* $\mathbf{0}$ has a unique *initial map* $?_X : \mathbf{0} \to X$ into every object X.

A *sum–product category* or *bi-cartesian category* is a category that has all finite products and coproducts, presented as binary and nullary products and coproducts. A *free* sum–product category is a category that is the *free sum–product completion* $\Sigma\Pi(\mathcal{C})$, the free completion with binary and nullary products and coproducts, of a base category $\mathcal{C}$. Formally, for products and coproducts to occur freely means that there is a functor $i : \mathcal{C} \to \Sigma\Pi(\mathcal{C})$ such that every functor F from $\mathcal{C}$ to a sum–product category $\mathcal{D}$ factors uniquely (up to natural isomorphism) as $F' \circ i$, where $F' : \Sigma\Pi(\mathcal{C}) \to \mathcal{D}$

preserves products and coproducts.

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "i"] \arrow[dr, "F"'] & \Sigma\Pi(\mathcal{C}) \arrow[d, dashed, "F'"] \\
 & \mathcal{D}
\end{tikzcd}$$

Objects $i(A)$ and morphisms $i(a)$ in $\Sigma\Pi(\mathcal{C})$, in the codomain of the functor i, are called *atomic*. For the remainder, let the base category $\mathcal{C}$ be fixed.

Free sum–product completions are a restriction to finite, discrete limits and colimits of the *bicompletions*, completions with all limits and colimits, studied by André Joyal in [63]. This work was inspired by Whitman's Theorem, from the 1940s, which characterises the free lattice completion of partially ordered sets by a property closely related to the subformula property. Generalising Whitman's Theorem, Joyal gave a characterisation of free bicomplete categories by a property called *softness*, plus several *atomicity* properties for atomic objects; from this perspective, free lattices are the special case of free bicomplete categories that are partial orders.

For the present case of free sum–product categories, softness is expressed in the following pushout diagram in the category of sets, where the arrows are the natural compositions with the appropriate projections and injections—e.g. the top arrow maps $f : X_i \to Y_j$ onto $f \circ \pi_i$.

$$\begin{tikzcd}
\coprod_{i,j} \hom(X_i, Y_j) \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & \coprod_j \hom(\prod_i X_i, Y_j) \arrow[d] \\
\coprod_i \hom(X_i, \coprod_j Y_j) \arrow[r] & \hom(\prod_i X_i, \coprod_j Y_j)
\end{tikzcd}$$

For binary products and coproducts it states that a morphism $f : X_0 \times X_1 \to Y_0 + Y_1$ factors through one of the projections or injections, i.e. arises as one of the following compositions, for some g or h,

$$X_0 \times X_1 \xrightarrow{\pi_i} X_i \xrightarrow{g} Y_0 + Y_1 \qquad\qquad X_0 \times X_1 \xrightarrow{h} Y_j \xrightarrow{\iota_j} Y_0 + Y_1$$

and if it factors through both a projection and an injection it does so via a common morphism $k : X_i \to Y_j$ (for some i and j), as follows.

$$\begin{tikzcd}
X_0 \times X_1 \arrow[r, "\pi_i"'] \arrow[rr, bend left, "h"] & X_i \arrow[r, "k"] \arrow[rr, bend right, "g"'] & Y_j \arrow[r, "\iota_j"] & Y_0 + Y_1
\end{tikzcd}$$

For the initial and terminal object, the diagram states that a morphism $f : X_0 \times X_1 \to \mathbf{0}$ factors through a projection π_i, that a morphism $g : \mathbf{1} \to Y_0 + Y_1$ factors through an injection ι_j, and that there is no map from $\mathbf{1}$ to $\mathbf{0}$. The atomicity properties for atomic objects $i(A)$ in $\Sigma\Pi(\mathcal{C})$, part of Joyal's characterisation in [63], state the following: maps $X_0 \times X_1 \to i(A)$ and $i(A) \to Y_0 + Y_1$ factor through a π_i and ι_j respectively, and a map $i(A) \to i(B)$ must be an atomic map $i(a)$, with $a \in \mathcal{C}(A,B)$. Since the objects in the category $\Sigma\Pi(\mathcal{C})$ are those generated over the atomic objects by taking finite products and coproducts, what the above amounts to is that any map $f : X \to Y$ can be constructed by a combination of pairing, copairing, and composition, from injections, projections, initial maps, terminal maps, and $\mathcal{C}$-maps, while passing only through objects that are components of X and Y.

Sum–product logic

One motivation for Joyal's work was the connection between categorical products and coproducts and the additives of linear logic [86]. Additive linear logic, or sum–product logic, provides a term calculus for sums and products, and a syntactic description of free sum–product categories. Following the categorical notation, and using the objects of $\mathcal{C}$ as the atomic formulae, the formulae of additive linear logic are generated by the grammar below.

$$X \quad := \quad A \in \mathcal{C} \;\mid\; \mathbf{0} \;\mid\; \mathbf{1} \;\mid\; X + X \;\mid\; X \times X$$

To recover Girard's notation for linear logic, read $\oplus$ for $+$, read & for $\times$, and read $\top$ for $\mathbf{1}$. The sequent calculus for sum–product logic, with maps from the category $\mathcal{C}$ as axioms, is displayed in Figure 2.1. The proof terms, which will be called $\Sigma\Pi(\mathcal{C})$-*terms*, are suggestive of the interpretation of proofs as categorical morphisms in $\Sigma\Pi(\mathcal{C})$; note that the overloading of the composition symbol $(\circ)$ is harmless, since π and ι will not occur in isolation.

Softness of $\Sigma\Pi(\mathcal{C})$ is related to the subformula property for sum–product logic, and to cut-elimination. This was the subject of investigations by Robin Cockett and Robert Seely in [25]. The equations in Figure 2.2, read from left to right, form a cut-elimination procedure for additive linear logic—note that the first case, which equates

$$\frac{a \in \mathcal{C}(A,B)}{A \xrightarrow{a} B} \qquad \frac{X \xrightarrow{s} Y_0 \qquad X \xrightarrow{t} Y_1}{X \xrightarrow{\langle s,t\rangle} Y_0 \times Y_1} \qquad \frac{X_i \xrightarrow{t} Y}{X_0 \times X_1 \xrightarrow{t\circ\pi_i} Y}$$

$$\frac{}{\mathbf{0} \xrightarrow{?} X} \qquad \frac{X \xrightarrow{t} Y_i}{X \xrightarrow{\iota_i \circ t} Y_0 + Y_1} \qquad \frac{X_0 \xrightarrow{s} Y \qquad X_1 \xrightarrow{t} Y}{X_0 + X_1 \xrightarrow{[s,t]} Y}$$

$$\frac{}{X \xrightarrow{!} \mathbf{1}}$$

$$\frac{}{X \xrightarrow{id_X} X}\ \text{Id} \qquad \frac{X \xrightarrow{t} Y \qquad Y \xrightarrow{s} Z}{X \xrightarrow{s\circ t} Z}\ \text{Cut}$$

Figure 2.1: Sum–product logic

composition in $\mathcal{C}$ and in $\Sigma\Pi(\mathcal{C})$, would read $b \circ a = b \circ a$ without the context of a proof (see also Table 2 in [25]). Using the equations in Figure 2.3 also the identity rule may be eliminated. Additional equations are given in Figure 2.4 (see also [25, Table 2] and [23, Figure 2]). Many of these equations, in all three figures, are the traditional permutations of the sequent calculus; for example, the top left equation of Figure 2.4, illustrated below as a permutation on sequent proofs.

$$\cfrac{\cfrac{X_1 \xrightarrow{t} Y_0}{X_0 \times X_1 \xrightarrow{t\circ\pi_1} Y_0}}{X_0 \times X_1 \xrightarrow{\iota_0\circ(t\circ\pi_1)} Y_0 + Y_1} = \cfrac{\cfrac{X_1 \xrightarrow{t} Y_i}{X_1 \xrightarrow{\iota_i\circ t} Y_0 + Y_1}}{X_0 \times X_1 \xrightarrow{(\iota_0\circ t)\circ\pi_1} Y_0 + Y_1}$$

The equations of the three figures together form an equational theory over proofs.

Definition 2.2.1. Two $\Sigma\Pi(\mathcal{C})$-terms s and t are *equal*, $\Sigma\Pi(\mathcal{C}) \models s = t$, if they are equated by the congruence over the equations in Figures 2.2, 2.3, and 2.4.

That equality over terms is a congruence means that it commutes with the term constructors

$$- \circ \pi_i \qquad \iota_j \circ - \qquad \langle -,- \rangle \qquad [-,-] \qquad - \circ -$$

or in other words, that the following equations hold, if $\Sigma\Pi(\mathcal{C}) \models t = t'$.

$$\begin{array}{llll} t \circ \pi_i = t' \circ \pi_i & \langle t,s \rangle = \langle t',s \rangle & [t,s] = [t',s] & t \circ s = t' \circ s \\ \iota_j \circ t = \iota_j \circ t' & \langle s,t \rangle = \langle s,t' \rangle & [s,t] = [s,t'] & s \circ t = s \circ t' \end{array}$$

Two main results in Cockett and Seely's paper, slightly paraphrased, are as follows.

$$\dfrac{\dfrac{a \in \mathcal{C}(A,B)}{A \xrightarrow{a} B} \qquad \dfrac{b \in \mathcal{C}(B,C)}{B \xrightarrow{b} C}}{A \xrightarrow{b\circ a} C}\,\text{Cut} \quad = \quad \dfrac{b \circ a \in \mathcal{C}(A,C)}{A \xrightarrow{b\circ a} C}$$

$$\begin{array}{rcl@{\qquad}rcl}
id \circ t & = & t & t \circ id & = & t \\
! \circ t & = & ! & t \circ ? & = & ? \\
(t \circ \pi_i) \circ \langle s_0, s_1 \rangle & = & t \circ s_i & [t_0, t_1] \circ (\iota_j \circ s) & = & t_j \circ s
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\langle t_0, t_1 \rangle \circ s & = & \langle t_0 \circ s,\ t_1 \circ s \rangle & t \circ (s \circ \pi_i) & = & (t \circ s) \circ \pi_i \\
(\iota_j \circ t) \circ s & = & \iota_j \circ (t \circ s) & t \circ [s_0, s_1] & = & [t \circ s_0,\ t \circ s_1]
\end{array}$$

Figure 2.2: Cut-elimination in sum–product logic

$$\dfrac{}{A \xrightarrow{id} A}\,\text{Id} \quad = \quad \dfrac{id \in \mathcal{C}(A,A)}{A \xrightarrow{id} A}$$

$$\begin{array}{rcl@{\qquad}rcl}
id_0 & = & ?_0 & id_{X+Y} & = & [\iota_0 \circ id_X,\ \iota_1 \circ id_Y] \\
id_1 & = & !_1 & id_{X\times Y} & = & \langle id_X \circ \pi_0,\ id_Y \circ \pi_1 \rangle
\end{array}$$

Figure 2.3: Identity-elimination in sum–product logic

$$\iota_i \circ (t \circ \pi_j) = (\iota_i \circ t) \circ \pi_j \qquad ! = ! \circ \pi_i$$

$$! = [!, !]$$

$$\iota_i \circ [t, s] = [\iota_i \circ t, \iota_i \circ s]$$

$$\iota_i \circ ? = ?$$

$$\langle t \circ \pi_i, s \circ \pi_i \rangle = \langle t, s \rangle \circ \pi_i$$

$$\langle ?, ? \rangle = ?$$

$$\langle [t_0, t_1], [s_0, s_1] \rangle = [\langle t_0, s_0 \rangle, \langle t_1, s_1 \rangle] \qquad !_{\mathbf{0}} = ?_{\mathbf{1}}$$

Figure 2.4: Equations in sum–product logic

Proposition 2.2.2 ([25, Proposition 4.6]). *The free sum–product completion* $\Sigma\Pi(\mathcal{C})$ *is characterised by sum–product logic, by taking as objects the formulae and as morphisms the equivalence classes of proofs under equality.*

Proposition 2.2.3 ([25, Proposition 2.9]). *For cut-free, identity-free proof terms* s *and* t*, if* $\Sigma\Pi(\mathcal{C}) \models s = t$ *then* s *and* t *are equated by the congruence over the equations in Figure 2.4.*

The statement of this second proposition implies that morphisms in $\Sigma\Pi(\mathcal{C})$ are represented by equivalence classes of cut-free proofs under the equations of Figure 2.4 alone. The following three facts then immediately imply that the *word problem* for $\Sigma\Pi(\mathcal{C})$, the problem of whether two proof terms denote the same morphism, is decidable if the word problem for $\mathcal{C}$ is decidable:

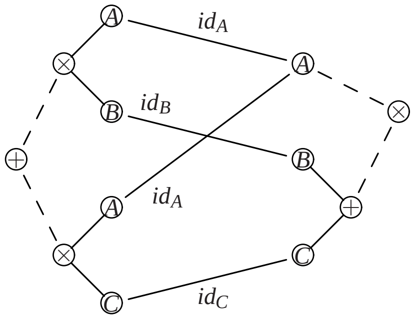
- every proof term is equal to a cut-free one, by cut elimination;
- up to the choice of $\mathcal{C}$-axioms, there are only finitely many proofs for a given conclusion sequent;
- to decide whether two cut-free terms are equated by the congruence of Figure 2.4 is straightforward.

Following up on the work in [25], in [23] Robin Cockett and Luigi Santocanale developed an intricate decision procedure for this decision problem (the word problem for $\Sigma\Pi(\mathcal{C})$), which runs in polynomial time.

2.3 Sum–product nets

Proof nets for additive linear logic, without units, were described in an unpublished report by Dominic Hughes in [56], while a similar approach appeared, in the same year, in [33]. They are also a fragment of the proof nets for multiplicative–additive linear logic without units by Dominic Hughes and Rob van Glabbeek (see [59, Section 4.10]). (An alternative graphical formalism, based on a different axiomatisation of sum–product categories, can be found in [6].) In this section proof nets in the style of Hughes and Van Glabbeek will be adapted to include the units, but not canonically. This notion of proof net will be called *sum–product nets*, and coincides with that of Hughes and Van Glabbeek on the fragment of additive linear logic without units.

A sum–product net representing a morphism $X \to Y$ consist of the two syntax trees plus a collection of *links*, connecting leaves in the syntax tree of X to leaves in that of Y. The object trees will be drawn facing each other with their leaves, their roots pointing outward. An example is drawn in Figure 2.5, together with the term it represents.



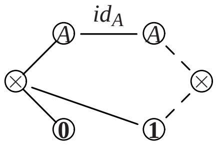
$$[\,\langle id_A \circ \pi_0,\ \iota_0 \circ id_B \circ \pi_1\rangle\ ,\ \langle id_A \circ \pi_0,\ \iota_1 \circ id_C \circ \pi_1\rangle\,]$$
$$:$$
$$(A \times B) + (A \times C) \longrightarrow A \times (B + C)$$

Figure 2.5: An example net

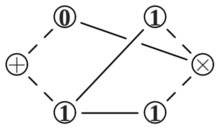
Nets are read from left to right, and correspond to cut-free proof terms in a simple way. Links correspond to axioms, and are labelled with the morphisms in the base category $\mathcal{C}$ which they represent. They are drawn slightly detached from vertices to distinguish them from the solid lines in the object trees, which represent projections and injections. Unlike the solid lines representing $\mathcal{C}$-morphisms, injections, and projections, dashed lines are not immediately interpreted as morphisms—as injections and projections they would run in the wrong direction, from right to left. Instead, a pair of dashed lines on a coproduct vertex in the source tree may be seen as corresponding to copairing $[-,-]$, and a pair of dashed lines on a product vertex in the target tree, to pairing $\langle -,- \rangle$.

To identify the actual nets among arbitrary collections of links, there is the following correctness criterion, called the *switching condition*. A *switching* is a choice selecting exactly one of the dashed edges of each coproduct vertex in the source tree and each product vertex in the target tree. A switching *switches off* the vertices in the branches it does not select, and *switches on* all other vertices in the tree. The switching condition states that, for any switching, in the remaining graph there must be exactly one path connecting both root nodes; or equivalently, for every switching there is exactly one link whose vertices are both switched on by the switching.

The nets so described are canonical for the unit-free fragment: they uniquely describe morphisms in categories with free, finite, non-empty products and coproducts (see also [56]). These nets may be extended to include the units in a straightforward manner: by adding (unlabelled) links that represent initial and terminal maps, as in the following examples.



$\langle id_A \circ \pi_0,\ !\rangle$



$[?, \langle !, !\rangle]$

The main technical difference is that these initial links and terminal links may connect to vertices that are not leaves; in particular, the switching condition is unaffected. Nets of this kind will be called *sum–product nets*. They are not canonical for additive linear logic with units—how to obtain canonical nets, using sum–products nets as a basis, will be the subject of the remainder of this part of the dissertation. A quick note: the feature that links may connect to non-leaf nodes is natural from the perspective of the sequent calculus, but it is not a strict necessity. It is quite possible to restrict all links to connect only to leaf nodes, but though this would simplify composition (see Section 3.3), it would needlessly complicate everything else.

Definitions

The *vertices* (or *positions*) in the syntax tree of an object X are given as binary words, elements of $\{0,1\}^*$, with the empty word denoted by ε, as follows. The set of positions of an object X is defined as follows.

$$\mathrm{pos}(A \in \mathcal{C}) \;=\; \mathrm{pos}(\mathbf{0}) \;=\; \mathrm{pos}(\mathbf{1}) \;=\; \{\varepsilon\}$$

$$\mathrm{pos}(X \times Y) \;=\; \mathrm{pos}(X+Y) \;=\; \{\varepsilon\} \,\cup\, \{0v \mid v \in \mathrm{pos}(X)\} \,\cup\, \{1v \mid v \in \mathrm{pos}(Y)\}$$

Variables $v, w, \ldots, z$ are used for vertices, while i and j range over $\{0,1\}$. The positions in $\mathrm{pos}(X)$ are ordered by the standard prefix ordering ($\leq$). The subformula of an object X at a vertex v is denoted X_v, defined as follows.

$$X_\varepsilon \;=\; X \qquad\qquad (X_0 \times X_1)_{iv} \;=\; (X_0 + X_1)_{iv} \;=\; (X_i)_v$$

When X is understood, the phrase ‘v is Y’ will mean $X_v = Y$. In this definition, a position v has children $v0$ and $v1$ if it is a product or a coproduct, and none otherwise.

Definition 2.3.1 (Prenets). A $\Sigma\Pi(\mathcal{C})$-*prenet* $(X,Y,\mathcal{R})$ consists of a *source* object X, a *target* object Y, and a *linking*, a relation

$$\mathcal{R} \quad \subseteq \quad \mathrm{pos}(X) \times \big(\mathrm{hom}(\mathcal{C}) \cup \{*\}\big) \times \mathrm{pos}(Y)$$

(where $* \notin \mathrm{hom}(\mathcal{C})$), such that for any $\langle v,l,w\rangle \in \mathcal{R}$, if $l = *$ then $X_v = \mathbf{0}$ or $Y_w = \mathbf{1}$; and otherwise X_v and Y_w are objects in $\mathcal{C}$, and $l \in \mathcal{C}(X_v,Y_w)$.

Variables f, g, h and k are used for prenets. The *links* in a prenet are the elements $\langle v,l,w\rangle$ of the linking $\mathcal{R}$, and may be rendered $\langle v,w\rangle$ when the label l is understood or irrelevant. A prenet is called *empty* when $\mathcal{R} = \varnothing$. A link $\langle v,*,w\rangle$, whose label $(*)$ will be omitted from diagrams, is a *unit* link; if v is $\mathbf{0}$ it is an *initial* link, and if w is $\mathbf{1}$ it is a *terminal* link. A link labelled with a $\mathcal{C}$-morphism is *atomic*.

A *switching* ς of an object X is a partial function on $\mathrm{pos}(X)$, that chooses one branch of each vertex that is a product: $\varsigma(v) \in \{0,1\}$ if X_v is a product, while otherwise $\varsigma(v)$ is undefined. The dual notion of a *co-switching* is a partial function choosing branches of the coproduct vertices in a syntax tree. A vertex w is *switched on* by a [co-]switching ς, written $\varsigma \circlearrowleft w$, if for any ancestor (i.e. prefix) of w that is a [co]product, ς selects the branch containing w:

$$\varsigma \circlearrowleft w \quad \overset{\Delta}{\iff} \quad \big(vi \leq w \wedge v \in \mathrm{dom}(\varsigma)\big) \Rightarrow \varsigma(v) = i\ .$$

Here, $\mathrm{dom}(\varsigma)$ indicates the domain of ς as a function, i.e. the vertices on which ς is defined. A switching for a prenet $(X,Y,\mathcal{R})$ is a pair (ς,τ) of a co-switching ς of X and a switching τ of Y. A link $\langle v,w\rangle$ is *switched on* by (ς,τ) if $\varsigma \circlearrowleft v$ and $\tau \circlearrowleft w$.

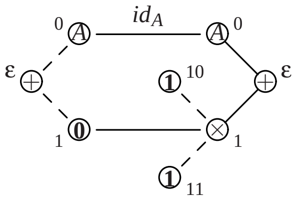
Definition 2.3.2 (Nets). A $\Sigma\Pi(\mathcal{C})$-*net* is a prenet f that satisfies the following correctness criterion (the *switching condition*).

- Every switching (ς,τ) for f switches on precisely one link.

Let NET denote the set of all $\Sigma\Pi(\mathcal{C})$-nets.

In the unit-free case $\Sigma\Pi(\mathcal{C})$-nets coincide with the proof nets in [56] and the additive fragment of the proof nets in [59].

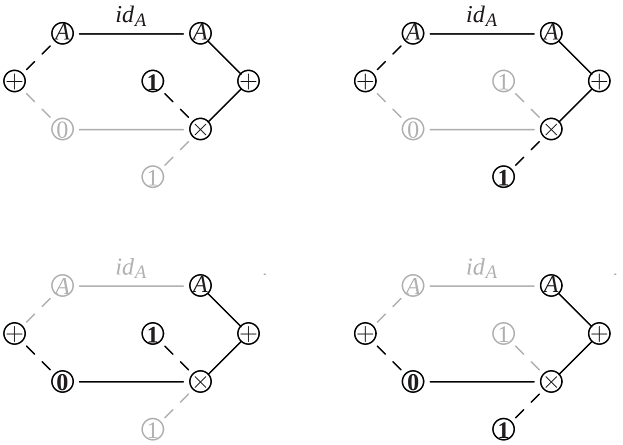
The example in Figure 2.6 illustrates a net, with its positions indicated, together with its formal definition. The dashed edges in the diagrams are those of nodes subject to switchings and co-switchings (in the switching condition). The net in Figure 2.6 has



$$(A+\mathbf{0},\ A+(\mathbf{1}\times\mathbf{1}),\ \{\,\langle 0, id_A, 0\rangle, \langle 1, *, 1\rangle\,\}\ )$$

Figure 2.6: Another example net

four switchings, shown below, each switching on exactly one link; vertices and links that are switched off are drawn in grey.



It is easily observed that, in any syntax tree, any vertex is switched on by at least one switching, and at least co-switching; and that consequently also each link in a prenet is switched on by at least one switching. When two links are switched on simultaneously by some switching, they are said to be *incompatible*; by the switching condition, a net may not contain incompatible links. Figure 2.7 shows examples of incompatible links. This notion is formalised below.

Definition 2.3.3 (Incompatibility)**.** In a prenet $(X, Y, \mathcal{R})$ vertices x, x' in $\mathrm{pos}(X)$, vertices y, y' in $\mathrm{pos}(Y)$, or links $\langle v, w\rangle, \langle v', w'\rangle$ in $\mathcal{R}$ are (pairwise) *incompatible*,

$$x \mathbin{\#} x' \qquad\qquad y \mathbin{\#} y' \qquad \text{or} \qquad \langle v, w\rangle \mathbin{\#} \langle v', w'\rangle$$

if there is a switching (ς, τ) for $(X, Y, \mathcal{R})$ such that

$$\varsigma \mathbin{\text{⏻}} x, x' \qquad\qquad \tau \mathbin{\text{⏻}} y, y' \qquad \text{or} \qquad (\varsigma, \tau) \mathbin{\text{⏻}} \langle v, w\rangle,\ \langle v', w'\rangle$$

respectively.

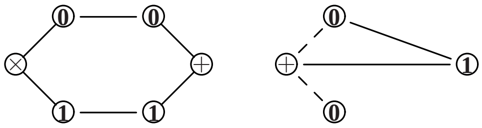


Figure 2.7: prenets with incompatible links

Thus, the vertices x and x' in a source object X are incompatible if there is a co-switching ς that switches them on simultaneously. As is easily seen, this occurs precisely when one (strictly) dominates the other ($x < x'$ or $x' < x$) or their greatest common ancestor is a product,

$$\exists v\, i\, j.\ \ vi \leq x,\ \ vj \leq x',\ \ i \neq j,\ \ \text{and } X_v \text{ is a product.}$$

Vertices y and y' in the target object Y are incompatible if a switching switches on both simultaneously, or equivalently if neither dominates the other and their greatest common ancestor is a product. Two links $\langle v, w\rangle$ and $\langle v', w'\rangle$ in $\mathcal{R}$ are incompatible precisely when $v \# v'$ and $w \# w'$.

The switching condition has an at–least component, which will be called the *connectedness condition*, and an at–most component, the *compatibility condition* The following are technically useful classes of pre-nets.

Definition 2.3.4 (Connected prenets)**.** A pre-net is *connected* if every switching for it switches on at least one link.

Definition 2.3.5 (Partial nets)**.** A pre-net is a *partial net* if it satisfies the *compatibility condition*, the condition that any switching for it switches on at most one link. Let PNET denote the set of partial nets.

The compatibility condition is so named because it is equivalent to the statement that a prenet may not contain incompatible links.

2.4 Connecting nets and terms

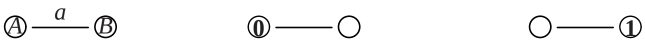
The connection between sum–product nets and the (cut-free) proof terms of sum–product logic will be made via an inductive construction method for nets. It will consist of *basic* nets, corresponding to axioms, and *net constructors*, corresponding to inference rules. These will give rise to a translation procedure from terms to nets.

Showing that all nets are so constructed will give an interpretation of nets as terms, or *sequentialisation*.

Basic nets are those consisting of a single link connecting the root vertices of both objects. Define the following abbreviation for basic nets.

$$(X,Y,l) \quad \stackrel{\Delta}{=} \quad (X,Y,\{\,\langle\varepsilon,l,\varepsilon\rangle\,\})$$

Below, basic nets are illustrated, and additional notation $(?,!)$ is introduced for nets consisting of a single initial or terminal link. Here, a is a morphism in $\mathcal{C}(A,B)$, and X and Y are $\Sigma\Pi(\mathcal{C})$-objects; note that the unlabelled nodes in the diagrams stand for subtrees, not just leaf nodes.



$$(A,B,a) \qquad ?_Y \stackrel{\Delta}{=} (\mathbf{0},Y,*) \qquad !_X \stackrel{\Delta}{=} (X,\mathbf{1},*)$$

The *constructors* are the following, for $\Sigma\Pi(\mathcal{C})$-objects X and Y, and $i,j \in \{0,1\}$.

$$(\pi_i(X\times Y);-) \qquad [-,-] \qquad \langle -,-\rangle \qquad (-;\iota_j(X+Y))$$

The annotation with objects $X\times Y$ and $X+Y$, in the first and last constructor above, will mostly be omitted. The constructors are illustrated in Figure 2.8; the dotted lines labelled f and g denote the pre-nets to which the constructors are applied, while the unlabelled vertices abbreviate syntax trees of arbitrary objects. The notation for terms and for nets is distinguished by the use of different alphabets (s,t and $\mathrm{f},\mathrm{g},\mathrm{h},\ldots$ respectively), the use of italics for terms and an upright font for nets, and different notation for composition with projections and injections. (The distinct notation is introduced to help avoid confusion.) Using the following operations,

$$u\cdot\mathcal{R} \quad \stackrel{\Delta}{=} \quad \{\langle uv,l,w\rangle \mid \langle v,l,w\rangle\in\mathcal{R}\} \qquad \mathcal{R}\cdot u \quad \stackrel{\Delta}{=} \quad \{\langle v,l,uw\rangle \mid \langle v,l,w\rangle\in\mathcal{R}\}\ ,$$

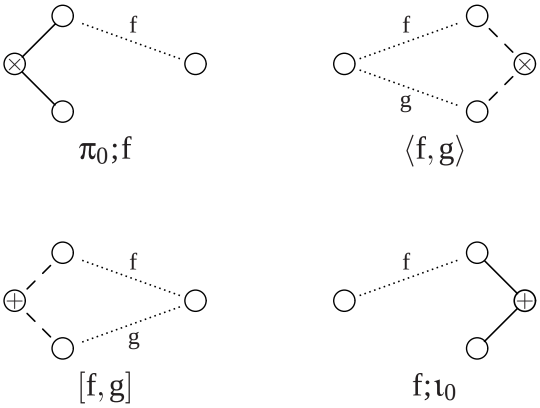
the constructors are defined, on pre-nets, below; note that like the term constructors, they are subject to well-typedness conditions.

$$\pi_i(X_0\times X_1);(X_i,Y,\mathcal{R}) \stackrel{\Delta}{=} (X_0\times X_1, Y,\ i\cdot\mathcal{R})$$

$$[(X,Z,\mathcal{R}),(Y,Z,\mathcal{S})] \stackrel{\Delta}{=} (X+Y,Z,(0\cdot\mathcal{R})\cup(1\cdot\mathcal{S}))$$

$$\langle(X,Y,\mathcal{R}),(X,Z,\mathcal{S})\rangle \stackrel{\Delta}{=} (X,Y\times Z,(\mathcal{R}\cdot 0)\cup(\mathcal{S}\cdot 1))$$

$$(X,Y_i,\mathcal{R});\iota_i(Y_0+Y_1) \stackrel{\Delta}{=} (X,Y_0+Y_1,\ \mathcal{R}\cdot 0)$$



$\pi_0;\mathrm{f}$ $\langle \mathrm{f}, \mathrm{g} \rangle$

$[\mathrm{f}, \mathrm{g}]$ $\mathrm{f};\iota_0$

Figure 2.8: Net constructors

The translation from (cut-free) proof terms to nets, implicit in the naming of constructors, is made explicit as $\llbracket - \rrbracket$ below.

Definition 2.4.1. The translation function $\llbracket - \rrbracket$ from $\Sigma\Pi(\mathcal{C})$-terms to $\Sigma\Pi(\mathcal{C})$-nets is defined as follows.

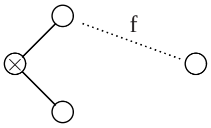
$$\llbracket ?_Y \rrbracket = ?_Y \qquad \llbracket !_X \rrbracket = !_X \qquad \llbracket t \circ \pi_i \rrbracket = \pi_i;\llbracket t \rrbracket \qquad \llbracket \langle t, s \rangle \rrbracket = \langle \llbracket t \rrbracket, \llbracket s \rrbracket \rangle$$

$$\llbracket a : A \to B \rrbracket = (A, B, a) \qquad \llbracket [t, s] \rrbracket = [\,\llbracket t \rrbracket, \llbracket s \rrbracket\,] \qquad \llbracket \iota_j \circ t \rrbracket = \llbracket t \rrbracket;\iota_j$$

Applying a constructor is called *construction*. The reverse notion, *deconstruction*, is the extraction of a pre-net f or g from one $\pi_i;\mathrm{f}$, $\langle \mathrm{f}, \mathrm{g} \rangle$, $[\mathrm{f}, \mathrm{g}]$, or $\mathrm{f};\iota_j$. Both construction and deconstruction preserve the switching condition, and moreover, the connectedness and compatibility conditions, individually, as well.

Lemma 2.4.2. *Construction and deconstruction preserve the connectedness and compatibility conditions.*

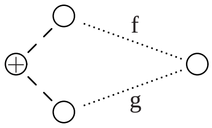
Proof. There are four cases, one for each of the constructors. For the first case let $\mathrm{f} = (X_0, Y, \mathcal{R})$, so that $(\pi_0(X);\mathrm{f})$ is $(X, Y, 0 \cdot \mathcal{R})$, depicted below.



For any co-switching ς of X there is a co-switching ς_0 of X_0 defined by $\varsigma_0(v) = \varsigma(0v)$, and this mapping is surjective: any switching of X_0 is a switching ς_0 for some ς. It follows that $\varsigma \mathbin{\text{⏻}} 0v$ if and only if $\varsigma_0 \mathbin{\text{⏻}} v$, while all links in $\pi_0;\mathrm{f}$ are of the form $\langle 0v, w \rangle$. Clearly, any switching (ς, τ) for $\pi_0;\mathrm{f}$ switches on precisely as many links as does (ς_0, τ)

for f. Then π_0;f is connected, respectively a partial net, if and only if f is. The case for π_1;f is symmetric.

Next, let $\mathrm{f} = (X_0, Y, \mathcal{R}_0)$ and $\mathrm{g} = (X_1, Y, \mathcal{R}_1)$, so that $[\mathrm{f},\mathrm{g}]$ is $(X, Y, 0\cdot\mathcal{R}_0 \cup 1\cdot\mathcal{R}_1)$, illustrated below.



Given a co-switching ς let ς_0 and ς_1 be co-switchings on X_0 and X_1 respectively, defined by $\varsigma_i(v) = \varsigma(iv)$. Conversely, every pair of co-switchings ς_0 for X_0 and ς_1 for X_1 defines two co-switchings ς and ς' for X, by letting $\varsigma(iv) = \varsigma'(iv) = \varsigma_i(v)$, while $\varsigma(\varepsilon) = 0$ and $\varsigma'(\varepsilon) = 1$. For any co-switching ς for X it follows that $\varsigma \circlearrowleft iv$ if and only if $\varsigma(\varepsilon) = i$ and $\varsigma_i \circlearrowleft v$. Then a switching (ς, τ) for $[\mathrm{f},\mathrm{g}]$ switches on precisely as many links as does (ς_0, τ) for f if $\varsigma(\varepsilon) = 0$, and as many as does (ς_1, τ) for g if $\varsigma(\varepsilon) = 1$. It follows that $[\mathrm{f},\mathrm{g}]$ is connected resp. compatible if and only if both f and g are.

The third and fourth case, for $\langle -,- \rangle$ and $(-;\iota_j)$, are dual to the above. ☐

From the above lemma, and the fact that basic nets are nets, it is immediate that the translation $[\![t]\!]$ of a term is a net. It remains to show that all nets arise as the translation of some term. Call a prenet *left-constructible* if it is of the form π_i;f or $[\mathrm{f},\mathrm{g}]$, and *right-constructible* if it is of the form $\langle \mathrm{f},\mathrm{g} \rangle$ or f;ι_i. Call a pre-net *constructible* if it is left-constructible or right-constructible, and *bi-constructible* if it is both. Recall that a partial net is a pre-net satisfying the compatibility condition.

Lemma 2.4.3. *A partial net is empty, basic, or constructible.*

Proof. Let $\mathrm{f} = (X, Y, \mathcal{R})$ be a partial net. It will be assumed that f is neither left- nor right-constructible, nor empty, to show that f is basic or to arrive at a contradiction. The assumption of f non-empty and not left-constructible gives two possibilities:

1) $\mathcal{R}$ contains some link $\langle \varepsilon, w \rangle$,

2) X is a product, and $\mathcal{R}$ contains some links $\langle 0v, w \rangle$ and $\langle 1v', w' \rangle$.

These options are exhaustive: if X is an atom or unit, the links in $\mathcal{R}$ all have ε as their source; if X is a coproduct, then f is left-constructible if and only if no links in $\mathcal{R}$ have source ε; if X is a product, f is left-constructible if and only if, for some $i \in \{0,1\}$, all links in $\mathcal{R}$ are of the form $\langle iv, w \rangle$. Dually, assuming $\mathcal{R}$ non-empty and not right-constructible gives two options,

$a)$ $\mathcal{R}$ contains some link $\langle v, \varepsilon \rangle$,

$b)$ Y is a coproduct, and $\mathcal{R}$ contains some links $\langle x, 0y \rangle$ and $\langle x', 1y' \rangle$.

This leaves four combinations to be verified.

$1a)$ If the link in $1)$ and that in $a)$ are distinct, $\langle \varepsilon, w \rangle \neq \langle v, \varepsilon \rangle$, the compatibility condition is violated, since $\varepsilon \# v$ and $w \# \varepsilon$ (recall that # denotes incompatibility, the relation that vertices are switched on simultaneously by some (co-)switching). Otherwise, $\mathcal{R}$ is the singleton $\{\langle \varepsilon, \varepsilon \rangle\}$: the presence of any other link $\langle v, w \rangle$ would violate the compatibility condition. Then f must be basic.

$1b)$ Given $\langle \varepsilon, w \rangle$, $\langle x, 0y \rangle$ and $\langle x', 1y' \rangle$, since $\varepsilon \# x$ and $\varepsilon \# x'$ the compatibility condition demands that neither $w \# 0y$ nor $w \# 1y'$. But since Y is a coproduct, if $w = 1w'$ then $1w' \# 0y$, if $w = 0w'$ then $0w' \# 1y'$, and if $w = \varepsilon$ then both $\varepsilon \# 0y$ and $\varepsilon \# 1y'$, a contradiction.

$2a)$ This case is dual to $1b)$ above.

$2b)$ The links given by $2)$ are $\langle 0v, w \rangle$ and $\langle 1v', w' \rangle$. Because $0v \# 1v'$, by the compatibility condition it cannot be that $w \# w'$. This means that w and w' must reside in the same branch of the coproduct Y, that is, $i \leq w$ and $i \leq w'$ for some $i \in \{0, 1\}$. Without loss of generality, assume that $0 \leq w$ and $0 \leq w'$. Dually, the links of $b)$ are $\langle x, 0y \rangle$ and $\langle x', 1y' \rangle$, and (without loss of generality) assume that $0 \leq x$ and $0 \leq x'$. Then $x' \# 1v'$ and $1y' \# w'$, violating the compatibility condition because of the links $\langle x', 1y' \rangle$ and $\langle 1v', w' \rangle$.

□

The above lemmata are used to show, firstly, that partial nets are precisely the pre-nets constructed over basic nets and empty pre-nets, and secondly, that nets are precisely the pre-nets constructed over basic nets.

Proposition 2.4.4. PNET *is the smallest set containing all empty pre-nets and basic nets, closed under construction.* NET *is the smallest set containing all basic nets, closed under construction.*

Proof. Both statements will be proved simultaneously. In one direction, it is immediate that empty pre-nets are partial nets, and basic nets are nets; and by Lemma 2.4.2 construction preserves connectedness and compatibility.

For the other direction, let f be a partial net. It will be shown that f is constructed over empty pre-nets and basic nets, or only basic nets if f is a net, by induction on the source and target object of f. By Lemma 2.4.3 the partial net f is empty, basic or constructible. In the first two cases, the statements are immediate (f is non-empty if it is a net). In the third case, f is of one of the four forms below.

$$\pi_i;\mathrm{g} \quad \langle \mathrm{g},\mathrm{h}\rangle \quad [\mathrm{g},\mathrm{h}] \quad \mathrm{g};\iota_j$$

By Lemma 2.4.2, since f is compatible so are the components g and h, and if f is connected, so are g and h. Moreover, either the source or the target objects of g and h are strictly smaller than that of f, while the other remains identical to that of f. Then if f is a partial net, so are g and h; by the induction hypothesis these are constructed over empty pre-nets and basic nets, and hence so is f. Similarly, if f is a net, g and h are nets that, by the induction hypotheses, are constructed over basic nets, and hence f is also a net. ☐

Sequentialisation is then an immediate corollary.

Corollary 2.4.5 (Sequentialisation)**.** *Every sum–product net* f *is the translation of some term* t, $\mathrm{f} = [\![t]\!]$.

Proof. Immediate from Proposition 2.4.4. ☐

The present proof of sequentialisation is similar to that in [56] (see Proposition 3 and Subsection 4.2.3 in that paper). There, the absence of units slightly simplifies the argument, because links connect only to leaves; however, the combinatorial reasoning is very similar in both cases. The proof in [59] is not directly comparable, due to the complicated issues arising from the presence of the multiplicative connectives.

2.5 An equational theory over nets

Sum–product nets factor out some, but not all of the equations over sum–product logic displayed in Figure 2.4. It will be shown how the remaining equations form an equational theory over nets, whose equivalence classes represent the morphisms of the free sum–product category $\Sigma\Pi(\mathcal{C})$.

Firstly, bi-constructible pre-nets—those that are both left-constructible and right-constructible—come in four kinds, illustrated in Figure 2.9. They are governed by the following equations.

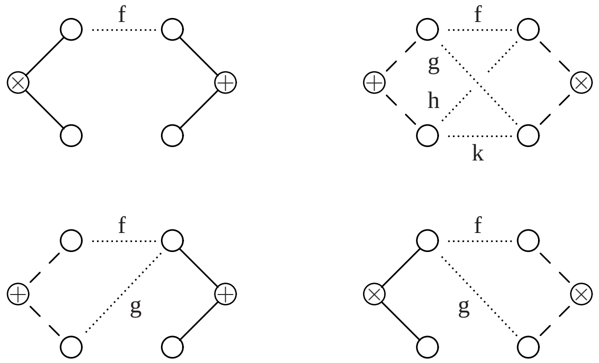


Figure 2.9: Bi-constructible pre-nets

Proposition 2.5.1. *Sum–product nets satisfy*

$$(\pi_i;f);\iota_j \;=\; \pi_i;(f;\iota_j) \qquad \langle[f,h],[g,k]\rangle \;=\; [\langle f,g\rangle,\langle h,k\rangle]$$
$$[f,g];\iota_j \;=\; [(f;\iota_j),(g;\iota_j)] \qquad \langle(\pi_i;f),(\pi_i;g)\rangle \;=\; \pi_i;\langle f,g\rangle\ .$$

Proof. Immediate from the definition of the constructors. ☐

The corresponding equations over terms are the four not involving the units, i.e.

$$\iota_i \circ (t \circ \pi_j) \;=\; (\iota_i \circ t) \circ \pi_j \qquad \langle[t_0,t_1],[s_0,s_1]\rangle \;=\; [\langle t_0,s_0\rangle,\langle t_1,s_1\rangle]$$
$$\iota_i \circ [t,s] \;=\; [\iota_i \circ t, \iota_i \circ s] \qquad \langle t \circ \pi_i, s \circ \pi_i\rangle \;=\; \langle t,s\rangle \circ \pi_i\ .$$

Because initial and terminal links are labelled uniformly nets satisfy $[\![!_{\mathbf{0}}]\!] = [\![?_{\mathbf{1}}]\!] = (\mathbf{0},\mathbf{1},*)$, absorbing the additional equation $!_{\mathbf{0}} = ?_{\mathbf{1}}$. That nets do not accidentally equate too many proof terms is established by the following proposition.

Proposition 2.5.2. *For cut-free $\Sigma\Pi(\mathcal{C})$-terms s and t, if $[\![s]\!] = [\![t]\!]$ then $\Sigma\Pi(\mathcal{C}) \models s = t$.*

Proof. By induction on the construction of a net f it will be shown that all terms s such that $[\![s]\!] = \mathrm{f}$, of which there is at least one by Corollary 2.4.5, are equated in $\Sigma\Pi(\mathcal{C})$. The base case concerns basic nets, and the induction step constructible nets; it is immediate from the definitions that a net cannot be both basic and constructible.

For basic nets, if $\mathrm{f} = (A,B,a)$ then s can only be $a \in \mathcal{C}(A,B)$, by the definition of the translation function $[\![-]\!]$. Next, if $\mathrm{f} = (\mathbf{0},\mathbf{1},*)$ then s is either $?_{\mathbf{1}}$ or $!_{\mathbf{0}}$, while if f is some other net $(\mathbf{0},Y,*)$ or $(X,\mathbf{1},*)$ then s can only be $?_Y$ and $!_X$ respectively.

For constructible nets, f can be of the form

$$\pi_i;g \qquad [g_0,g_1] \qquad \langle h_0,h_1\rangle \quad \text{or} \quad h;\iota_j\ ,$$

of which the two leftmost are mutually exclusive, as are the two rightmost. Without loss of generality let $\mathrm{f} = \pi_i;\mathrm{g}$. From the induction hypothesis it is immediate that all terms $t \circ \pi_i$ translating to $\pi_i;\mathrm{g}$ are equated. If f is only left-constructible, there are no other terms translating to f. Otherwise, f is bi-constructible; then let f be of the form $\mathrm{h};\iota_j$ (the case for $\mathrm{f} = \langle \mathrm{h}_0, \mathrm{h}_1 \rangle$ is similar). It follows from the definition of the constructors that $\mathrm{g} = \mathrm{k};\iota_j$ and $\mathrm{h} = \pi_i;\mathrm{k}$ for some net k, and as in Proposition 2.5.1,

$$\mathrm{f} \quad = \quad \pi_i;(\mathrm{k};\iota_j) \quad = \quad (\pi_i;\mathrm{k});\iota_j \ .$$

Let t' be a term such that $[\![t']\!] = \mathrm{k}$. Then for any terms $s \circ \pi_i$ and $\iota_j \circ t$ translating to f,

$$[\![s]\!] \ = \ [\![\iota_j \circ t']\!] \quad \text{and} \quad [\![t]\!] \ = \ [\![t' \circ \pi_i]\!] \ .$$

The induction hypothesis and the sum–product equations then give

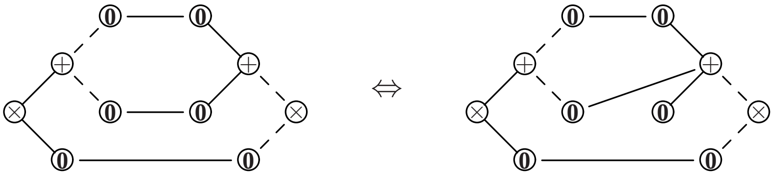
$$\Sigma\Pi(\mathcal{C}) \models \quad s \circ \pi_i \quad = \quad (\iota_j \circ t') \circ \pi_i \quad = \quad \iota_j \circ (t' \circ \pi_i) \quad = \quad \iota_j \circ t \ .$$

□

The four remaining equations over sum–product logic, below, will impose an equational theory over nets, *equivalence* ($\Leftrightarrow$), illustrated in Figure 2.10.

$$! \ = \ ! \circ \pi_0 \qquad ! \ = \ [!, !] \qquad ? \ = \ \iota_0 \circ ? \qquad ? \ = \ \langle ?, ? \rangle$$

Equivalence ($\Leftrightarrow$) over nets must reflect that the term equations above form a congruence. For example, the following nets must be equivalent, as they are the translations of equated terms.



$$\langle [\iota_0 \circ ?, \iota_1 \circ ?] \circ \pi_0 \, , \, ? \circ \pi_1 \rangle \qquad = \qquad \langle [\iota_0 \circ ?, ?] \circ \pi_0 \, , \, ? \circ \pi_1 \rangle$$

The natural way of defining equivalence of nets is via graph-rewriting, by interpreting the equivalences in Figure 2.10 as replacing one *subnet* with another, leaving the context intact. In the remainder of this section it will be shown that ($\Leftrightarrow$), defined as a rewrite relation on nets, naturally corresponds to equality of $\Sigma\Pi(\mathcal{C})$-terms.

Firstly, to define ($\Leftrightarrow$), a notion of subnet is needed. A *subprenet* of $(X, Y, \mathcal{R})$ will be a prenet between subformulae of X and Y, with a subcollection of the links between

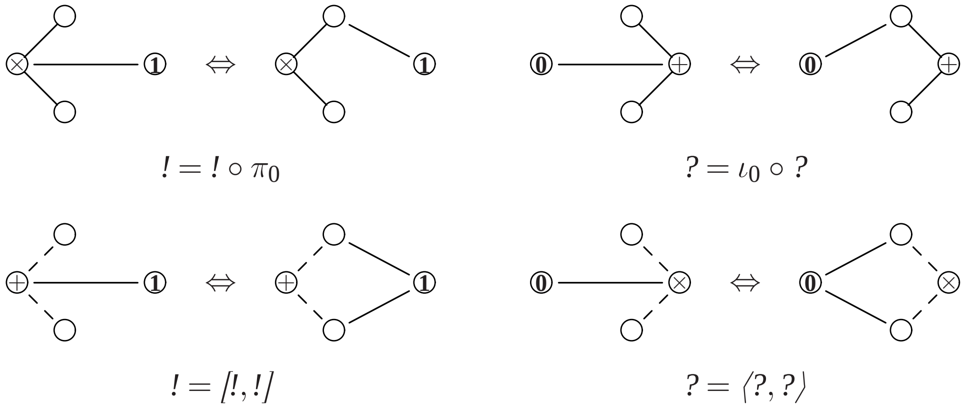


Figure 2.10: The unit laws force an equational theory over nets

them: a prenet $(X_v, Y_w, \mathcal{S})$ such that $v \cdot \mathcal{S} \cdot w \subseteq \mathcal{R}$. Call two prenets *parallel* if they have identical source objects and identical target objects, and define, on parallel prenets,

$$(X, Y, \mathcal{S}) \subseteq (X, Y, \mathcal{R}) \quad \overset{\Delta}{\Longleftrightarrow} \quad \mathcal{S} \subseteq \mathcal{R} \ .$$

Define, for a prenet $\mathrm{f} = (X, Y, \mathcal{R})$,

$$\begin{aligned} \mathrm{f}_{v,w} &\overset{\Delta}{=} (X_v, Y_w,\ \mathcal{R}_{v,w}) \\ \mathcal{R}_{v,w} &\overset{\Delta}{=} \{\langle v', l, w' \rangle \mid \langle vv', l, ww' \rangle \in \mathcal{R}\} \ . \end{aligned}$$

Definition 2.5.3 (Subnets)**.** A *subprenet* of a prenet f is a prenet $\mathrm{g} \subseteq \mathrm{f}_{v,w}$. If $\mathrm{g} \subseteq \mathrm{f}$ then g is *wide*, if $\mathrm{g} = \mathrm{f}_{v,w}$ then g is *full*, and g is a *subnet* if it is a net. The set of subnets of a prenet f is denoted by $\textsc{sub}(\mathrm{f})$.

The notation $\mathrm{f}\{\mathrm{g}\}_{v,w}$ denotes a pre-net f with the sub-prenet $\mathrm{f}_{v,w}$ replaced by a parallel prenet g. Formally, for prenets $\mathrm{f} = (X, Y, \mathcal{R})$ and $\mathrm{g} = (X_v, Y_w, \mathcal{S})$, define

$$\begin{aligned} \mathrm{f}\{\mathrm{g}\}_{v,w} &\overset{\Delta}{=} (X, Y,\ \mathcal{R}\{\mathcal{S}\}_{v,w}) \\ \mathcal{R}\{\mathcal{S}\}_{v,w} &\overset{\Delta}{=} \{\langle v', l, w' \rangle \in \mathcal{R} \mid v \not\leq v' \ \vee\ w \not\leq w'\} \cup (v \cdot \mathcal{S} \cdot w) \end{aligned}$$

The general form of rewriting in context is given by the following relation.

$$\mathrm{f}\{\mathrm{g}\}_{v,w} \ \Rightarrow\!\![\mathrm{g} \,|\, \mathrm{h}]\!\!\Rightarrow_{v,w}\ \mathrm{f}\{\mathrm{h}\}_{v,w}$$

The relation $\Rightarrow\!\![\mathrm{g} \,|\, \mathrm{h}]\!\!\Rightarrow_{v,w}$ replaces the prenet between vertices v and w, which is required to be g, with the parallel pre-net h, leaving the context intact. An equivalent formulation would be $\mathrm{f} =\!\!\![\mathrm{f}_{v,w} \,|\, \mathrm{h}]\!\!\Rightarrow_{v,w} \mathrm{f}\{\mathrm{h}\}_{v,w}$. Dropping the subscript v,w indicates the union over all v and w, and a single application of a rewrite relation $=\!\!\![\mathrm{g} \,|\, \mathrm{h}]\!\!\Rightarrow$ (i.e. for some v and w) will be called a *rewrite step*.

Definition 2.5.4 (Equivalence)**.** The equational theory $\Leftrightarrow$ (*equivalence*) on $\Sigma\Pi$-nets is the equivalence relation generated by the following four relations.

$$=\!\!\![\,!\,|\,\pi_i;!\,]\!\!\Rightarrow \qquad =\!\!\![\,!\,|\,[!,!]\,]\!\!\Rightarrow \qquad =\!\!\![\,?\,|\,\langle ?,?\rangle\,]\!\!\Rightarrow \qquad =\!\!\![\,?\,|\,?;\iota_j\,]\!\!\Rightarrow$$

The four rewrite rules in the above definition are the equivalences illustrated in Figure 2.10, interpreted as rewrite steps from left to right, on subnets; naturally, in the equational theory $\Leftrightarrow$, they are applied in both directions. From the illustration it is easily observed that they preserve the switching condition. Note that there are no side-conditions to the application of these equations—unlike the rewriting in multiplicative proof nets with units, where rewrites only apply on the condition that they preserve the correctness criterion for multiplicative proof nets (see [17] and [57]). It remains to show that $\Leftrightarrow$ reflects precisely the equational theory over sum–product terms. The first step will be to show that subnets of sum–product nets are analogous to subterms in sum–product logic. To make this more precise: for any subnet g of a net f, there is a term t with subterm s such that $\mathrm{f} = [\![t]\!]$ and $\mathrm{g} = [\![s]\!]$. This is established below.

Lemma 2.5.5. *For a net* f *the set* $\textsc{sub}(\mathrm{f})$ *of subnets of* f *is the union of* $\{\mathrm{f}\}$, $\textsc{lsub}(\mathrm{f})$ *and* $\textsc{rsub}(\mathrm{f})$, *where:*

$$\textsc{lsub}(\mathrm{f}) = \begin{cases} \textsc{sub}(\mathrm{g}) \cup \textsc{sub}(\mathrm{h}) & \text{if } \mathrm{f} = [\mathrm{g},\mathrm{h}] \\ \textsc{sub}(\mathrm{g}) & \text{if } \mathrm{f} = \pi_i;\mathrm{g} \\ \varnothing & \text{otherwise} \end{cases}$$

$$\textsc{rsub}(\mathrm{f}) = \begin{cases} \textsc{sub}(\mathrm{g}) & \text{if } \mathrm{f} = \mathrm{g};\iota_i \\ \textsc{sub}(\mathrm{g}) \cup \textsc{sub}(\mathrm{h}) & \text{if } \mathrm{f} = \langle \mathrm{g},\mathrm{h}\rangle \\ \varnothing & \text{otherwise} \end{cases}$$

Proof. One direction is immediate: $\textsc{lsub}(\mathrm{f}) \subset \textsc{sub}(\mathrm{f})$ and $\textsc{rsub}(\mathrm{f}) \subset \textsc{sub}(\mathrm{f})$. For the other it must be shown that $\textsc{sub}(\mathrm{f}) \subseteq \{\mathrm{f}\} \cup \textsc{lsub}(\mathrm{f}) \cup \textsc{rsub}(\mathrm{f})$.

Firstly, if $\mathrm{gf}_{v,w}$ is a net, $\mathrm{g} = \mathrm{f}_{v,w}$: if $\mathrm{f}_{v,w}$ violates the compatibility condition (i.e. has a switching that switches on more than one link), so does f, since there is always a switching for f that switches on v and w. Then consider the subnet $\mathrm{f}_{v,w}$ of the net f. By Lemma 2.4.3 f is basic, or left- or right-constructible. If f is basic then $\mathrm{f}_{v,w}$ is empty unless $v = w = \varepsilon$, which means that the only subnet of f is f itself.

For left-constructible f, the only case that is not immediate is $v = \varepsilon$, by the following reasoning. Firstly, if $\mathrm{f} = \pi_i;\mathrm{g}$ and $i \leq v$ then $\mathrm{f}_{v,w}$ is a subnet of g, and hence in $\textsc{lsub}(\mathrm{f})$; if on the other hand v resides in the branch opposite i, i.e. $(1-i) \leq v$, then $\mathrm{f}_{v,w}$ is empty, and not a subnet. Secondly, if $\mathrm{f} = [\mathrm{g},\mathrm{h}]$ then $\mathrm{f}_{v,w}$ is a subnet of g or h unless $v = \varepsilon$. Thus for left-constructible f, unless $v = \varepsilon$ the statement is immediate. Dually, for right-constructible f only the case $w = \varepsilon$ is not immediate. For bi-constructible f this leaves only the case $v = w = \varepsilon$, which is again immediate.

Of the two remaining cases, consider the one where f is left-constructible, but not right-constructible, and $v = \varepsilon$ but $w \neq \varepsilon$; the other case is dual. Since $\textsc{rsub}(\mathrm{f})$ is empty, and $\mathrm{f}_{\varepsilon,w}$ is not in $\textsc{lsub}(\mathrm{f})$, it must be shown that $\mathrm{f}_{\varepsilon,w}$ is not a net. Let $\mathrm{f} = (X,Y,\mathcal{R})$. Because f is not right-constructible, either some $\langle x,\varepsilon\rangle \in \mathcal{R}$, or Y is a coproduct and some $\langle x,0y\rangle, \langle x',1y'\rangle \in \mathcal{R}$. If $\langle x,\varepsilon\rangle \in \mathcal{R}$, let ς be a co-switching on X such that $\varsigma \mathrel{\text{⏻}} x$. Since ε is switched on by any switching on Y, there can be no other links $\langle x',y'\rangle$ in $\mathcal{R}$ such that $\varsigma \mathrel{\text{⏻}} x'$. Then in $\mathrm{f}_{\varepsilon,w}$ there are no links switched on by (ς,τ), for any switching τ on Y, violating the connectedness condition.

In the remaining case Y is a coproduct and $\langle x,0y\rangle, \langle x',1y'\rangle \in \mathcal{R}$. Because $w \neq \varepsilon$ either $0 \leq w$ or $1 \leq w$; without loss of generality let $1 \leq w$, as the other case is symmetric. Fix a co-switching ς of X and a switching τ of Y such that $\langle x,0y\rangle$ is the only link switched on, while simultaneously $\tau \mathrel{\text{⏻}} w$ (since Y is a coproduct, such a switching exists). Then in f no link $\langle v',w'\rangle$ such that $w \leq w'$ is switched on by ς and τ. Let τ' be the switching of Y_w that agrees with τ, in the sense that $\tau'(u) = \tau(wu)$. In $\mathrm{f}_{\varepsilon,w}$ no link is switched on by ς and t', violating the connectedness requirement, so it is not a net. □

The proposition below establishes that equivalence over sum–product nets is sound and complete for term equality in $\Sigma\Pi(\mathcal{C})$.

Proposition 2.5.6. *For cut-free proof terms s and t of sum–product logic,*

$$\Sigma\Pi(\mathcal{C}) \models s = t \quad \iff \quad [\![s]\!] \Leftrightarrow [\![t]\!] \,.$$

Proof. From left to right, the argument is by induction on the derivation of term equality. If $\Sigma\Pi(\mathcal{C}) \models s = t$ is an instance of one of the equations

$$\iota_i \circ (t \circ \pi_j) \;=\; (\iota_i \circ t) \circ \pi_j \qquad \langle [t_0,t_1], [s_0,s_1] \rangle \;=\; [\langle t_0,s_0\rangle, \langle t_1,s_1\rangle]$$

$$\iota_i \circ [t,s] \;=\; [\iota_i \circ t, \iota_i \circ s] \qquad \langle t \circ \pi_i, s \circ \pi_i \rangle \;=\; \langle t,s\rangle \circ \pi_i$$

$$?_{\mathbf{1}} = !_{\mathbf{0}}$$

then $\llbracket s \rrbracket = \llbracket t \rrbracket$ (see Proposition 2.5.1). Secondly, if $\Sigma\Pi(\mathcal{C}) \models s = t$ is an instance of one of the equations

$$! \;=\; ! \circ \pi_0 \qquad ! \;=\; [!,!] \qquad ? \;=\; \iota_0 \circ ? \qquad ? \;=\; \langle ?,? \rangle$$

then $s \Leftrightarrow t$ follows from an application of one of the rewrite steps below (in either direction).

$$\dashv\![\,!\mid \pi_i;!\,]\!\Rightarrow_{\varepsilon,\varepsilon} \qquad \dashv\![\,!\mid [!,!]\,]\!\Rightarrow_{\varepsilon,\varepsilon} \qquad \dashv\![\,?\mid \langle ?,? \rangle\,]\!\Rightarrow_{\varepsilon,\varepsilon} \qquad \dashv\![\,?\mid ?;\iota_j\,]\!\Rightarrow_{\varepsilon,\varepsilon}$$

Thirdly, if $\Sigma\Pi(\mathcal{C}) \models s = t$ by a series of equations, then $\llbracket s \rrbracket \Leftrightarrow \llbracket t \rrbracket$ follows by transitivity of $\Leftrightarrow$. Finally, let $\Sigma\Pi(\mathcal{C}) \models s = t$ be an instance of one of the following equations, while $\Sigma\Pi(\mathcal{C}) \models s' = t'$ and $\Sigma\Pi(\mathcal{C}) \models s'' = t''$.

$$\begin{array}{ll} s' \circ \pi_i = t' \circ \pi_i & \qquad \langle s', s'' \rangle = \langle t', t'' \rangle \\ \iota_j \circ s' = \iota_j \circ t' & \qquad [s', s''] = [t', t''] \end{array}$$

The case where $s = s' \circ \pi_i$ and $t = t' \circ \pi_i$ is treated explicitly; the others are similar. The induction hypothesis gives $\llbracket s' \rrbracket \Leftrightarrow \llbracket t' \rrbracket$. This equivalence consists of a series of rewrite steps $\dashv\![\,\mathrm{g} \mid \mathrm{h}\,]\!\Rightarrow_{v,w}$. But if $\mathrm{g} = \mathrm{f}_{v,w}$ then also $\mathrm{g} = (\pi_i;\mathrm{f})_{iv,w}$. Then by taking the rewrite step $\dashv\![\,\mathrm{g} \mid \mathrm{h}\,]\!\Rightarrow_{iv,w}$ for each step above, $\llbracket s' \circ \pi_i \rrbracket \Leftrightarrow \llbracket t' \circ \pi_i \rrbracket$.

From right to left, firstly, if $\llbracket s \rrbracket = \llbracket t \rrbracket$ then by Proposition 2.5.2 $\Sigma\Pi(\mathcal{C}) \models s = t$. Otherwise, let the equivalence $\llbracket s \rrbracket \Leftrightarrow \llbracket t \rrbracket$ consist of a single rewrite step

$$\llbracket s \rrbracket \quad \dashv\![\,!\mid \pi_0;!\,]\!\Rightarrow_{v,w} \quad \llbracket t \rrbracket\ .$$

The other cases are similar, and the general case, for multiple steps, follows by transitivity. The present case is shown by induction on (the lengths of) v and w. Since $\mathrm{g} = \llbracket s \rrbracket_{v,w}$, by Lemma 2.5.5 one of the following five cases holds:

1. $v = w = \varepsilon$, and $\llbracket s \rrbracket =!$ and $\llbracket t \rrbracket = (\pi_0;!)$
2. $\llbracket s \rrbracket = \pi_i;\mathrm{f}$, the vertex v is iu, and $! = \mathrm{f}_{u,w}$
3. $\llbracket s \rrbracket = [\mathrm{f}_0, \mathrm{f}_1]$, the vertex v is iu, and $! = (\mathrm{f}_i)_{u,w}$
4. $\llbracket s \rrbracket = \mathrm{f};\iota_j$, the vertex w is ju, and $! = \mathrm{f}_{v,u}$
5. $\llbracket s \rrbracket = \langle \mathrm{f}_0, \mathrm{f}_1 \rangle$, the vertex w is ju, and $! = (\mathrm{f}_i)_{v,u}$

The first case is the base case of the induction. In this case, s must be $!$ or possibly $?_{\mathbf{1}}$, and likewise t is $! \circ \pi_0$ or $?_{\mathbf{1}} \circ \pi_0$; it follows immediately that $\Sigma\Pi(\mathcal{C}) \models s = t$. In the remaining cases neither $\llbracket s \rrbracket$ nor $\llbracket t \rrbracket$ is basic, and by Lemma 2.4.3 both must be constructible. Let $\llbracket s \rrbracket$ be of the form $\langle \mathrm{f}_0, \mathrm{f}_1 \rangle$; the other three cases are similar. It is easily inferred that the rewrite step $=\!\![\,!\,|\,\pi_0;!\,]\!\!\Rrightarrow_{v,w}$ does not affect right-constructibility (for this particular case, it is sufficient that it does not add a rooted link $\langle x, \varepsilon \rangle$). Then $\llbracket t \rrbracket$ is of the form $\langle \mathrm{g}_0, \mathrm{g}_1 \rangle$. Without loss of generality, let $w = 1u$; the rewrite step under consideration is then

$$\langle \mathrm{f}_0, \mathrm{f}_1 \rangle \quad =\!\![\,!\,|\,\pi_0;!\,]\!\!\Rrightarrow_{v,1u} \quad \langle \mathrm{g}_0, \mathrm{g}_1 \rangle \ .$$

It follows that $\mathrm{f}_0 = \mathrm{g}_0$ and

$$\mathrm{f}_1 \quad =\!\![\,!\,|\,\pi_0;!\,]\!\!\Rrightarrow_{v,u} \quad \mathrm{g}_1 \ .$$

Let s_0, s_1 and t_1 be terms translating to f_0, f_1 and g_1 respectively. By the induction hypothesis $\Sigma\Pi(\mathcal{C}) \models s_1 = t_1$. The remaining equations below follow by Proposition 2.5.2, from $\llbracket s \rrbracket = \llbracket \langle s_0, s_1 \rangle \rrbracket$ and $\llbracket t \rrbracket = \llbracket \langle s_0, t_1 \rangle \rrbracket$.

$$\Sigma\Pi(\mathcal{C}) \models \quad s \quad = \quad \langle s_0, s_1 \rangle \quad = \quad \langle s_0, t_1 \rangle \quad = \quad t$$

$\square$

Chapter 3

Saturated nets

3.1 Introduction

In the previous chapter sum–product nets were introduced, and it was shown that equivalence classes of sum–product nets under the equational theory $(\Leftrightarrow)$ are in one–to–one correspondence with morphisms in free sum–product categories. The current chapter will present a simple rewrite relation called *saturation*, in Section 3.2, that rewrites sum–product nets to a canonical form called *saturated nets*. The description of saturated nets, which are a canonical representation of free sum–product categories, is a central contribution of this part of the dissertation.

The category of saturated nets is described in more detail in Section 3.3, which includes a treatment of identity and composition in the category of saturated nets. A second main contribution, a correctness criterion for saturated nets, is discussed in Section 3.4. The final section of the chapter, Section 3.5, looks at the time complexity of saturation as a decision procedure.

3.2 Deciding equivalence of nets

The equivalence relation $(\Leftrightarrow)$ over nets will be decided by rewriting equivalent nets to a common canonical form. A natural first question is whether a suitable, confluent rewrite relation can be obtained by orientating the equivalence rewrites, i.e. by restricting them to one direction. Two straightforward candidates are to rewrite towards the leaves or towards the the roots of the trees. A first, concrete example illustrating that, in fact, equivalence rewrites need to be employed in both directions, is given by the example equivalence chain in Figure 3.1.

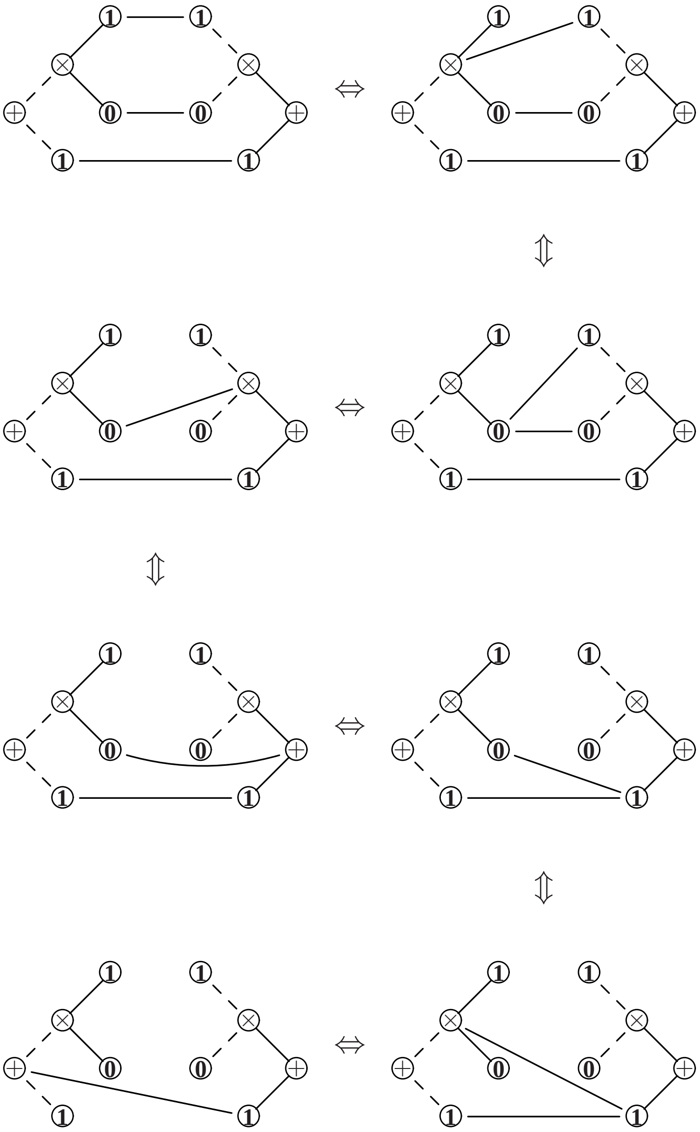


Figure 3.1: The equivalence rewrites in action

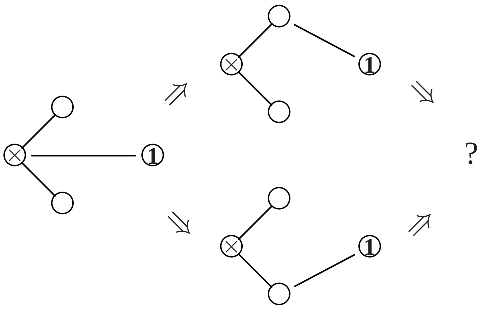
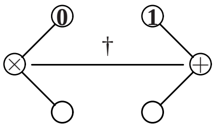


Figure 3.2: Rewriting towards the leaves is non-confluent

A more precise analysis will show that neither direction of rewriting is confluent. For rewriting towards the leaves, an example of non-confluence is illustrated in Figure 3.2. For the other direction, rewriting towards the roots, the situation is more delicate. To solve the non-confluence of the example in Figure 3.3, definitions can be adapted to allow the following ‘net’.



† this is not a legal link

To permit this simple construction merely requires an additional type of link, which it is possible to define coherently, while no modification to the correctness criterion for nets, the switching condition, is needed. However, the non-confluence of the example in Figure 3.4 has no solution along these lines.

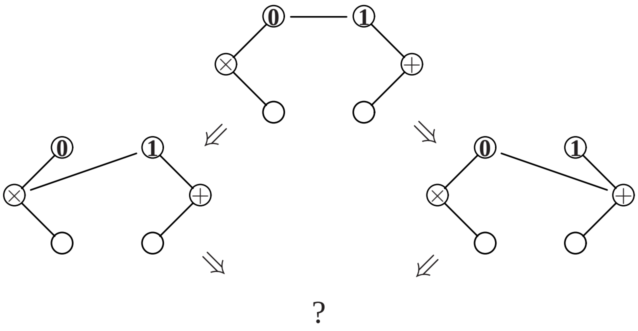


Figure 3.3: Rewriting towards the roots is non-confluent (1)

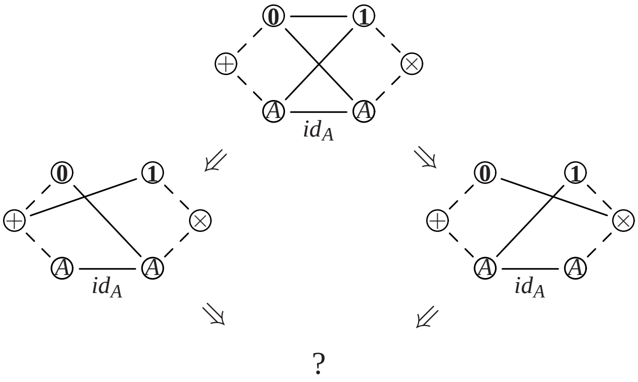


Figure 3.4: Rewriting towards the roots is non-confluent (2)

Since confluent rewriting seems impossible without breaking the switching condition, the obvious next step is to break it. Then when two nets rewrite into each other, the easiest way to obtain confluence is to combine the links of both, as in the example of Figure 3.5. This gives a simple rewrite relation, that will be called *saturation*.

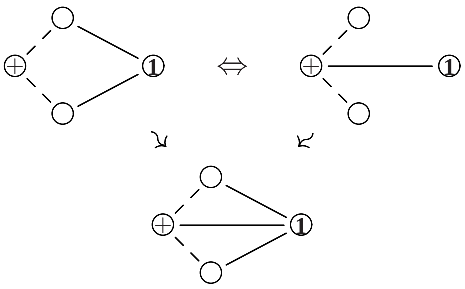


Figure 3.5: Saturation

To formally define the saturation relation a different form of rewriting is required, whereby links are added to a net, rather than replaced. Let the *union* of two parallel pre-nets be the union of their collections of links,

$$(X,Y,\mathcal{R})\cup(X,Y,\mathcal{S}) \;\triangleq\; (X,Y,\mathcal{R}\cup\mathcal{S})\;.$$

Define a second template for specifying rewrites as follows.

$$\mathrm{f}\quad \sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow_{v,w}\quad \mathrm{f}\{\mathrm{f}_{v,w}\cup\mathrm{h}\}_{v,w}\qquad\qquad \text{if}\quad \mathrm{g}\subseteq\mathrm{f}_{v,w}$$

Informally, if the pre-net f contains the subnet $\mathrm{g}\subseteq\mathrm{f}_{v,w}$, add the links of the pre-net h, parallel to g. The difference with the first rewrite template $=\![\mathrm{g}\,|\,\mathrm{h}]\!\Rightarrow_{v,w}$, used in Section 2.5 to define equivalence over nets $(\Leftrightarrow)$ (Definition 2.5.4), is that in $\sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow_{v,w}$ the subprenet $\mathrm{f}_{v,w}$ may contain other links than those in g, and the links of h are added to those of h, instead of replacing them. Dropping the subscript, the rewrite relation $\sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow$ includes all rewrite steps $\sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow_{v,w}$ for some v and w.

Definition 3.2.1. The *saturation* relation $\rightsquigarrow$ on pre-nets is the union of the following eight relations.

$$\begin{array}{cccc}
\sim\!(\pi_i;!\,|\,!)\!\rightsquigarrow & \sim\!([!,!]\,|\,!)\!\rightsquigarrow & \sim\!(\langle ?,?\rangle\,|\,?)\!\rightsquigarrow & \sim\!(?;\iota_j\,|\,?)\!\rightsquigarrow \\
\sim\!(!\,|\,\pi_i;!)\!\rightsquigarrow & \sim\!(!\,|\,[!,!])\!\rightsquigarrow & \sim\!(?\,|\,\langle ?,?\rangle)\!\rightsquigarrow & \sim\!(?\,|\,?;\iota_j)\!\rightsquigarrow
\end{array}$$

The relation $\rightsquigarrow^{-}$ is the irreflexive restriction of $\rightsquigarrow$.

The eight *saturation steps* in Definition 3.2.1 are illustrated in Figure 3.6. Note that for each saturation step $\sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow$ there is a corresponding equivalence $=\![\mathrm{g}\,|\,\mathrm{h}]\!\Rightarrow$: although Definition 2.5.4 lists only four equivalence steps, $(\Leftrightarrow)$ is symmetric. The main differences between saturation $(\rightsquigarrow)$ and equivalence $(\Leftrightarrow)$ are: one, saturation is a directed, single-step rewrite relation, where $(\Leftrightarrow)$ is an equivalence relation; two, $(\rightsquigarrow)$

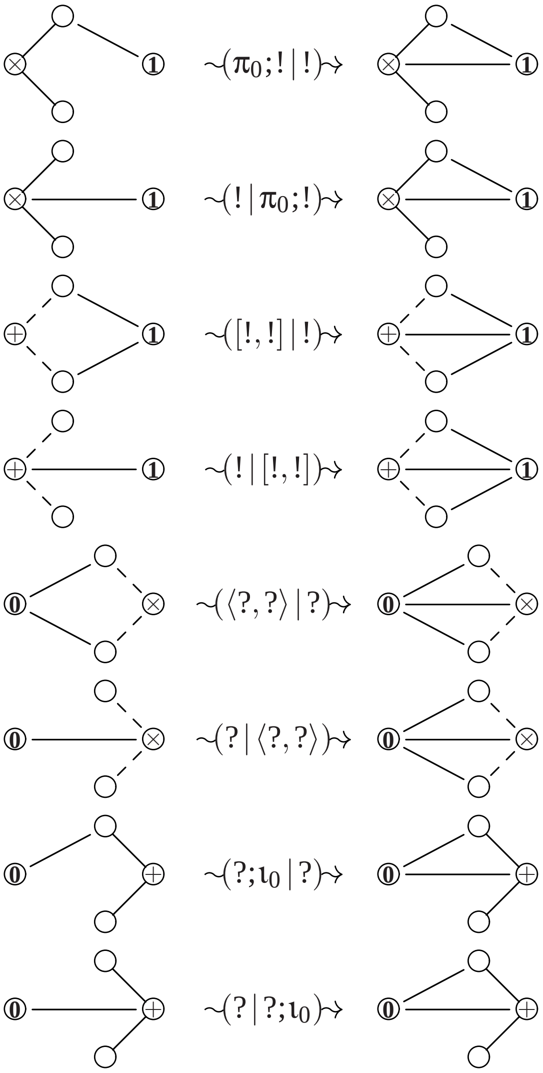

$\sim\!(\pi_0;!\,|\,!)\!\rightsquigarrow$
$\sim\!(!\,|\,\pi_0;!)\!\rightsquigarrow$
$\sim\!([!,!]\,|\,!)\!\rightsquigarrow$
$\sim\!(!\,|\,[!,!])\!\rightsquigarrow$
$\sim\!(\langle ?,?\rangle\,|\,?)\!\rightsquigarrow$
$\sim\!(?\,|\,\langle ?,?\rangle)\!\rightsquigarrow$
$\sim\!(?;\iota_0\,|\,?)\!\rightsquigarrow$
$\sim\!(?\,|\,?;\iota_0)\!\rightsquigarrow$

Figure 3.6: Saturation steps

is defined on prenets, where $(\Leftrightarrow)$ is defined only on nets; three, $(\rightsquigarrow)$ only adds links to a prenet, where $(\Leftrightarrow)$ as a rewrite relation replaces links with others. In general, the relation $\sim\!(\mathrm{g}\,|\,\mathrm{h})\!\rightsquigarrow_{v,w}$ is reflexive for nets that already have h (and g) as a subnet between vertices v and w. In order to provide saturation with a standard notion of termination, the irreflexive variant $\rightsquigarrow^{-}$ is defined. Both $\rightsquigarrow$ and $\rightsquigarrow^{-}$ will be referred to as saturation, with the distinction only made when necessary. Figure 3.7 shows an example net being saturated; the first image, top left, shows the original net, the last, bottom left, its saturation. In between, for each saturation step the links that trigger it and the links that it introduces are displayed in black, for emphasis, while other links are shaded grey; an equals sign indicates when two nets differ only in shading.

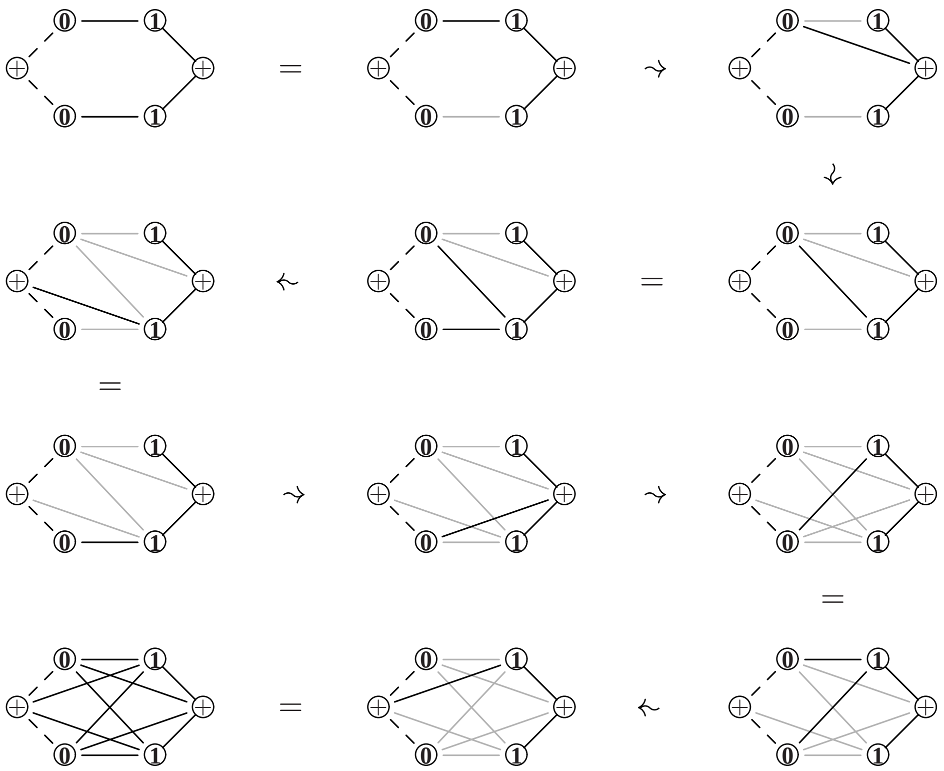


Figure 3.7: Saturating a net

Proposition 3.2.2. *The saturation relation $(\rightsquigarrow^{-})$ is confluent and strongly normalising.*

Proof. For strong normalisation it is sufficient to observe that each step in $\rightsquigarrow^{-}$ adds one or two unit links to a pre-net, while the number of unit links in a pre-net $(X,Y,\mathcal{R})$ is bounded by the size of $\mathrm{pos}(X)\times\mathrm{pos}(Y)$.

For confluence, let $\mathrm{f} = (X, Y, \mathcal{R})$, let $\mathrm{g}' = (X_v, Y_w, \mathcal{S})$, and let $\mathrm{h}' = (X_x, Y_y, \mathcal{T})$. Observe that the result of applying a saturation step $\rightsquigarrow\!(\mathrm{g}\,|\,\mathrm{g}')\!\rightsquigarrow_{v,w}$ to f is just

$$\mathrm{f}\{\mathrm{f}_{v,w} \cup \mathrm{g}'\}_{v,w} \quad = \quad (X, Y,\ \mathcal{R}\ \cup\ v \cdot \mathcal{S} \cdot w)\ .$$

The following diagram shows local confluence for $\rightsquigarrow$.

$$\begin{tikzcd}
 & & (X,Y,\mathcal{R}) \arrow[dl, squiggly, no head] \arrow[dr, squiggly, no head] & & \\
 & (\mathrm{g}\,|\,\mathrm{g}')_{v,w} \arrow[dl, squiggly] & & (\mathrm{h}\,|\,\mathrm{h}')_{x,y} \arrow[dr, squiggly] & \\
(X,Y,\mathcal{R}\ \cup\ v \cdot \mathcal{S} \cdot w) \arrow[dr, squiggly, no head] & & & & (X,Y,\mathcal{R}\ \cup\ x \cdot \mathcal{T} \cdot y) \arrow[dl, squiggly, no head] \\
 & (\mathrm{h}\,|\,\mathrm{h}')_{x,y} \arrow[dr, squiggly] & & (\mathrm{g}\,|\,\mathrm{g}')_{v,w} \arrow[dl, squiggly] & \\
 & & (X,Y,\mathcal{R}\ \cup\ v \cdot \mathcal{S} \cdot w \cup\ x \cdot \mathcal{T} \cdot y) & &
\end{tikzcd}$$

Then also $\rightsquigarrow^{-}$ is locally confluent, and in the context of strong normalisation this implies $\rightsquigarrow^{-}$ is confluent. ☐

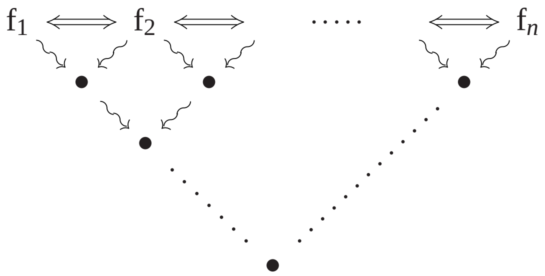
The normal form of a pre-net f with respect to $\rightsquigarrow^{-}$ is denoted $\sigma\mathrm{f}$, and, if f is a net, is called a *saturated net*. The idea is that saturation provides a decision procedure by comparing saturated nets, i.e. $\mathrm{f} \Leftrightarrow \mathrm{g}$ if and only if $\sigma\mathrm{f} = \sigma\mathrm{g}$. The left–to–right direction, $\mathrm{f} \Leftrightarrow \mathrm{g} \ \Rightarrow\ \sigma\mathrm{f} = \sigma\mathrm{g}$, states that comparing saturated nets is complete for deciding equivalence, i.e. it makes all the identifications that $(\Leftrightarrow)$ makes. From right to left, $\sigma\mathrm{f} = \sigma\mathrm{g} \ \Rightarrow\ \mathrm{f} \Leftrightarrow \mathrm{g}$ states the soundness direction, that comparing saturated nets makes only the identifications that $(\Leftrightarrow)$ makes.

Theorem 3.2.3 (Completeness)**.** *For nets* f *and* g, *if* $\mathrm{f} \Leftrightarrow \mathrm{g}$ *then* $\sigma\mathrm{f} = \sigma\mathrm{g}$.

Proof. If $\mathrm{f} \Leftrightarrow \mathrm{f}'$ is witnessed by a single step $\mathrm{f} \Relbar[\mathrm{g}\,|\,\mathrm{h}]\Rightarrow_{v,w} \mathrm{f}'$ in the equivalence relation, then there is a common pre-net f'' such that $\mathrm{f} \rightsquigarrow \mathrm{f}''$ and $\mathrm{f}' \rightsquigarrow \mathrm{f}''$, as illustrated below.

$$\begin{tikzcd}
\mathrm{f}\{\mathrm{g}\}_{v,w} \arrow[rr, Rightarrow, no head] \arrow[dr, squiggly, no head] & & {[\mathrm{g}\,|\,\mathrm{h}]_{v,w}} \arrow[rr, Rightarrow] & & \mathrm{f}\{\mathrm{h}\}_{v,w} \arrow[dl, squiggly, no head] \\
 & (\mathrm{g}\,|\,\mathrm{h})_{v,w} \arrow[dr, squiggly] & & (\mathrm{h}\,|\,\mathrm{g})_{v,w} \arrow[dl, squiggly] & \\
 & & \mathrm{f}\{\mathrm{g} \cup \mathrm{h}\}_{v,w} & &
\end{tikzcd}$$

Any equivalence $\mathrm{f} \Leftrightarrow \mathrm{g}$ can be decomposed as a series of single-step equivalences $\mathrm{f} = \mathrm{f}_1 \Leftrightarrow \mathrm{f}_2 \Leftrightarrow \ldots \Leftrightarrow \mathrm{f}_n = \mathrm{g}$. Confluence then completes the following triangle diagram (note that the tip of the triangle need not be the normal form yet, as further saturation steps may be possible).



$\square$

The soundness theorem is stated below; its elaborate proof will be the subject of the next chapter.

Theorem 3.2.4 (Soundness)**.** *For* $\Sigma\Pi(\mathcal{C})$*-nets* f *and* g, *if* $\sigma\mathrm{f} = \sigma\mathrm{g}$ *then* $\mathrm{f} \Leftrightarrow \mathrm{g}$.

3.3 The category of saturated nets

An immediate consequence of the soundness and completeness theorems of the previous section, Theorem 3.2.4 and Theorem 3.2.3, is that saturated nets uniquely describe the morphisms in the category $\Sigma\Pi(\mathcal{C})$.

Theorem 3.3.1. *For cut-free, identity-free* $\Sigma\Pi(C)$*-terms s and t,*

$$\Sigma\Pi(\mathcal{C}) \models s = t \quad \Longleftrightarrow \quad \sigma[\![s]\!] = \sigma[\![t]\!] \ .$$

Proof. Immediate by Proposition 2.5.6, Theorem 3.2.3, and Theorem 3.2.4. $\square$

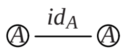
This section will give a more complete picture of the category of saturated nets. Firstly, an alternative characterisation of saturated nets will be provided. This will be used to provide a direct account of identities and composition for saturated nets, describing the category of saturated nets independently of the translation to and from sum–product logic.

Firstly, the following proposition asserts that the saturation of a net contains precisely the combined links of all equivalent nets.

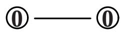
Proposition 3.3.2. *The saturation of a net* f *is* $\bigcup\{\mathrm{g} \mid \mathrm{f} \Leftrightarrow \mathrm{g}\}$.

The saturation process gives an intuition why this might hold, and it is immediate from the completeness theorem that a saturated net contains at least the links of all equivalent nets. Nevertheless, proving the proposition is not straightforward, and will be postponed until Section 4.8 in the next chapter, when the accumulated lemmata will have brought a proof within easy reach.

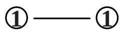
Composition and identity in the category of saturated nets are, naturally, fully determined by the translation from sum–product logic. Translating and then saturating identity proofs in sum–product logic gives the saturated identities $\sigma(\mathrm{id}_X)$ for each object X, where the net id_X is defined as follows.



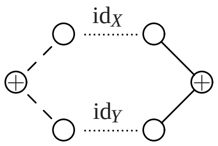
$$\mathrm{id}_A \triangleq (A, A, id_A)$$



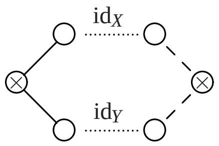
$$\mathrm{id}_{\mathbf{0}} \triangleq (\mathbf{0}, \mathbf{0}, *)$$



$$\mathrm{id}_{\mathbf{1}} \triangleq (\mathbf{1}, \mathbf{1}, *)$$



$$\mathrm{id}_{X+Y} \triangleq [\,(\mathrm{id}_X;\iota_0),(\mathrm{id}_Y;\iota_1)\,]$$



$$\mathrm{id}_{X\times Y} \triangleq \langle\,(\pi_0;\mathrm{id}_X),(\pi_1;\mathrm{id}_Y)\,\rangle$$

From the above it is easily deduced that in an identity net $\mathrm{id}_X = (X, X, \mathcal{R})$, before saturation, the linking $\mathcal{R}$ is the identity relation on the leaves in X, labelled appropriately.

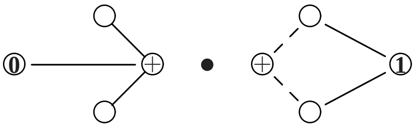
Composition of nets

Before turning to composition of saturated nets, first composition for nets will be discussed. An indirect account of composition is via cut elimination in the term calculus: to compose two nets f and g,

- find terms s and t such that $\llbracket s \rrbracket = \mathrm{f}$ and $\llbracket t \rrbracket = \mathrm{g}$;
- compose the two terms with a cut to form $t \circ s$;
- apply cut elimination to $t \circ s$, yielding a term r;
- and then translate r to a net $\llbracket r \rrbracket$.

All operations above preserve the denotation of terms and nets as categorical morphisms. Thus, while composition need not be associative, because cut elimination in the term calculus is non-confluent, it is associative up to equivalence.

For unit-free nets it was established by Hughes and Van Glabbeek that composition is the relational composition of linkings (see [56] or [59]). In the presence of the units, this does not work immediately: the following composition would be empty.



As is illustrated by this simple example, the problem is caused by links connecting to arbitrary nodes, whereas in the unit-free case, all links connect to the leaves. Because all nets have equivalent nets whose links connect only to leaves, reached simply by applying rewrites towards the leaves exhaustively, this problem will not be hard to solve. First, some terminology will be introduced.

Definition 3.3.3. A pair of prenets $(X,Y,\mathcal{R})$ and $(Y',Z,\mathcal{S})$ is *composable* if $Y = Y'$. The *relational composition* $(\bullet)$ of composable prenets is defined as

$$(X,Y,\mathcal{R}) \;\bullet\; (Y,Z,\mathcal{S}) \quad \overset{\Delta}{=} \quad (X,\, Z,\, \{\, \langle u, l \bullet k, w\rangle \mid \langle u,l,v\rangle \in \mathcal{R},\; \langle v,k,w\rangle \in \mathcal{S}\,\}\,)\;,$$

where the composition of labels is given by

$$(* \;\bullet\; l) \overset{\Delta}{=} * \qquad\qquad (l \;\bullet\; *) \overset{\Delta}{=} * \qquad\qquad (a \;\bullet\; b) \overset{\Delta}{=} (b \circ a)\;.$$

A pair of composable nets f and g is *matching* if $\mathrm{f} \bullet \mathrm{g}$ is a net.

Note that like the notion of composability, the property of being matching is not symmetric. Also, note that relational composition is defined on all prenets, while matching describes the class of (pairs of) nets for which relational composition is well-defined. In the lemma below relational composition is shown to satisfy a series of equations, corresponding to elimination and permutation steps of the cut-elimination procedure for sum–product logic, given in Figure 2.2 in Section 2.2.

Lemma 3.3.4. *Relational composition of prenets satisfies the following equations: for basic nets,*

$$\begin{aligned}
(A,B,a) \bullet (B,C,b) &= (A,C,b \circ a) & \qquad b \circ a &= b \circ a \\
(\mathbf{0},Y,*) \bullet (Y,Z,l) &= (\mathbf{0},Z,*) & t \circ ? &= ? \\
(X,Y,l) \bullet (Y,\mathbf{1},*) &= (X,\mathbf{1},*) & !\circ s &= !\;,
\end{aligned}$$

for right-constructible prenets composed with left-constructible prenets,

$$\begin{aligned}
(\mathrm{f}';\iota_j) \bullet [\mathrm{g}_0,\mathrm{g}_1] &= \mathrm{f}' \bullet \mathrm{g}_j & \qquad [t_0,t_1] \circ (\iota_j \circ s') &= t_j \circ s' \\
\langle \mathrm{f}_0,\mathrm{f}_1\rangle \bullet (\pi_i;\mathrm{g}') &= \mathrm{f}_i \bullet \mathrm{g}' & (t' \circ \pi_i) \circ \langle s_0,s_1\rangle &= t' \circ s_i\;,
\end{aligned}$$

and for $\mathrm{f} \bullet \mathrm{g}$ *with a left-constructible prenet* f *or right-constructible prenet* g*,*

$$\begin{array}{rcl@{\qquad}rcl}
[\mathrm{f}_0, \mathrm{f}_1] \bullet \mathrm{g} & = & [\mathrm{f}_0 \bullet \mathrm{g}, \mathrm{f}_1 \bullet \mathrm{g}] & t \circ [s_0, s_1] & = & [t \circ s_0, t \circ s_1] \\
(\pi_i; \mathrm{f}') \bullet \mathrm{g} & = & \pi_i; (\mathrm{f}' \bullet \mathrm{g}) & t \circ (s' \circ \pi_i) & = & (t \circ s') \circ \pi_i \\
\mathrm{f} \bullet (\mathrm{g}'; \iota_j) & = & (\mathrm{f} \bullet \mathrm{g}'); \iota_j & (\iota_j \circ t') \circ s & = & \iota_j \circ (t' \circ s) \\
\mathrm{f} \bullet \langle \mathrm{g}_0, \mathrm{g}_1 \rangle & = & \langle \mathrm{f} \bullet \mathrm{g}_0, \mathrm{f} \bullet \mathrm{g}_1 \rangle & \langle t_0, t_1 \rangle \circ s & = & \langle t_0 \circ s, t_1 \circ s \rangle .
\end{array}$$

Proof. Immediate by unfolding the definitions. For example, for the fourth equation, links $\langle u, v \rangle$ in f and $\langle v, w \rangle$ in g_0 give rise to a link $\langle u, w \rangle$ in $\mathrm{f} \bullet \mathrm{g}_0$ if and only if $\langle u, v0 \rangle$ in $\mathrm{f}; \iota_0$ and $\langle v0, w \rangle$ in $[\mathrm{g}_0, \mathrm{g}_1]$ give rise to the same link $\langle u, w \rangle$ in the composition of the latter two prenets. ☐

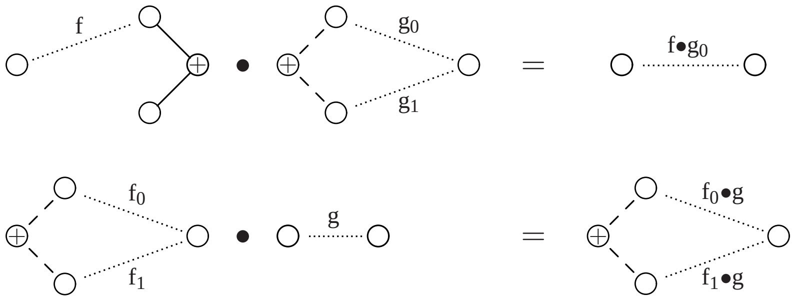


Figure 3.8: Composition via elimination and permutation steps

Two of the equations in Lemma 3.3.4 are illustrated in Figure 3.8. For matching nets, these equations give a complete description of composition, as is asserted by the lemma below. In addition, the lemma shows that for nets to be matching, it is sufficient that for the central, common object, links in both nets only connect to leaves. From this it is immediate that any composable nets f and g have equivalent nets f' and g' that are matching, by moving links towards the leaves. That the process of moving links towards the leaves is non-deterministic is not a problem, since the result of composing two nets need only be unique up to equivalence.

Lemma 3.3.5. *For composable nets* $\mathrm{f} = (X, Y, \mathcal{R})$ *and* $\mathrm{g} = (Y, Z, \mathcal{S})$*,*

1. *the equations of Lemma 3.3.4 characterise the relational composition* $\mathrm{f} \bullet \mathrm{g}$ *if and only if* f *and* g *are matching;*
2. *if all links connected to the central object* Y*, in particular all initial links in* f *and all terminal links in* g*, connect only to leaves of* Y*, then* f *and* g *are matching.*

Proof. For 1, from left to right is immediate: if the equations of Lemma 3.3.4 characterise $\mathsf{f} \bullet \mathsf{g}$, they do so by constructing it from basic nets. For the other direction, it is easily verified that since f and g are constructible (Proposition 2.4.4), the equations of Lemma 3.3.4 are exhaustive for all ways of constructing f and g if equations for the following two cases are added:

$$(\mathbf{0}, Y, *) \bullet \mathsf{g} \qquad\qquad \mathsf{f} \bullet (Y, \mathbf{1}, *)$$

where g is only left-constructible and f is only right-constructible. But these pairs are not matching: since g is left-constructible, it contains no links $\langle \varepsilon, w \rangle$, while $(\mathbf{0}, Y, *)$ contains only the link $\langle \varepsilon, \varepsilon \rangle$; then $(\mathbf{0}, Y, *) \bullet \mathsf{g}$ is empty, and not a net. The case for $\mathsf{f} \bullet (Y, \mathbf{1}, *)$ is similar. Furthermore, the last four equations in Lemma 3.3.4 preserve matching, in the following sense. For example for the equation

$$[\mathsf{f}_0, \mathsf{f}_1] \bullet \mathsf{g} \quad = \quad [\mathsf{f}_0 \bullet \mathsf{g}, \mathsf{f}_1 \bullet \mathsf{g}],$$

since $[\mathsf{f}_0, \mathsf{f}_1]$ and g are matching, both sides of the equation are nets; then also $\mathsf{f}_0 \bullet \mathsf{g}$ and $\mathsf{f}_1 \bullet \mathsf{g}$ are nets, which means that f_0 and g are matching, as are f_1 and g. For matching nets f and g it then follows by induction on their construction that the equations of Lemma 3.3.4 are exhaustive, which shows the remaining direction of 1 above.

For 2, it is easily observed that the two cases above,

$$(\mathbf{0}, Y, *) \bullet \mathsf{g} \qquad\qquad \mathsf{f} \bullet (Y, \mathbf{1}, *)$$

where g is only left-constructible and f is only right-constructible, cannot transpire, as follows. By assumption, since initial links in $(0, Y, *)$ only connect to leaves, Y is a leaf; but then g must be basic or right-constructible, a contradiction. The other case is symmetric. Then the equations of Lemma 3.3.4 characterise the composition of nets f and g with links only connecting to leaves in the central object Y, and 2 follows from 1. □

Next, it will be shown that, for matching nets, relational composition is the right notion of composition, in the sense that it commutes, up to equivalence, with composition via the term calculus, as outlined above.

Proposition 3.3.6. *For matching nets $[\![s]\!]$ and $[\![t]\!]$ translated from terms, the relational composition $[\![s]\!] \bullet [\![t]\!]$ is the translation $[\![r]\!]$ of a normal term r equal to $t \circ s$, the composition of s and t by a cut.*

Proof. Let $[\![s]\!] = \mathrm{f}$ and $[\![t]\!] = \mathrm{g}$. The statement is then shown by induction on the construction of f and g, following the equations of Lemma 3.3.4, which by Lemma 3.3.5 are exhaustive. Of the equations for basic nets, the second is treated, repeated below; the other two are similar.

$$(\mathbf{0},Y,*) \;\bullet\; (Y,Z,l) \;=\; (\mathbf{0},Z,*) \qquad\qquad t \circ ? \;=\; ?$$

In this case, s is a term equal to $?$ (since $[\![s]\!] = (\mathbf{0},Y,*)$), while t is a term such that $[\![t]\!]$ is a basic net. Let $r = ?$; the statement is then immediate from the following equations, plus the term equation above right.

$$[\![?]\!] = \mathrm{f} = (\mathbf{0},Y,*) \qquad [\![t]\!] = \mathrm{g} = (Y,Z,l) \qquad \mathrm{f} \bullet \mathrm{g} = (\mathbf{0},Z,*) = [\![?]\!]$$

Next, of the six equations for constructible nets, the first will be treated, repeated below; the others are similar.

$$(\mathrm{f}';\iota_j) \;\bullet\; [\mathrm{g}_0,\mathrm{g}_1] \;=\; \mathrm{f}' \bullet \mathrm{g}_j \qquad\qquad /t_0,t_1/ \circ (\iota_j \circ s') \;=\; t_j \circ s'$$

In this case, since $\mathrm{f} = \mathrm{f}';\iota_j$, there is a term s' such that $\mathrm{f} = [\![\iota_0 \circ s']\!]$. By Proposition 2.5.6 (soundness and completeness of $(\Leftrightarrow)$ for term equality under translation), from this and $[\![s]\!] = \mathrm{f}$ it follows that $\Sigma\Pi(\mathcal{C}) \models s = \iota_0 \circ s'$. Similarly, there are terms t_0 and t_1 such that $\mathrm{g} = [\![/t_0,t_1/]\!]$ and $\Sigma\Pi(\mathcal{C}) \models t = /t_0,t_1/$. Because f and g are matching, $\mathrm{f} \bullet \mathrm{g} = \mathrm{f}' \bullet \mathrm{g}_j$ is a net, which means it is immediate that f' and g_j are matching. Then the induction hypothesis can be applied, giving the following equations, for some normal term r.

$$[\![s]\!] \bullet [\![t]\!] \;=\; [\![s']\!] \bullet [\![t_j]\!] \;=\; [\![r]\!] \qquad\qquad \Sigma\Pi(\mathcal{C}) \models\; r \;=\; t_j \circ s'$$

By the equation $\Sigma\Pi(\mathcal{C}) \models /t_0,t_1/ \circ (\iota_j \circ s') = t_j \circ s'$ (one of the equations for cut elimination in Figure 2.2), it follows that $\Sigma\Pi(\mathcal{C}) \models r = t \circ s$, concluding the statement. ☐

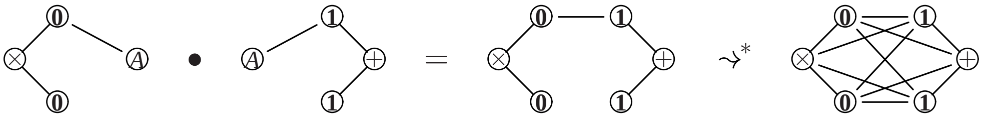
The following proposition is then immediate, from the above and the earlier result that translation between terms and nets commutes with term equality and net equivalence (Proposition 2.5.6).

Proposition 3.3.7. *For nets* $\mathrm{f} \Leftrightarrow \mathrm{f}'$ *and* $\mathrm{g} \Leftrightarrow \mathrm{g}'$, *if* f *and* g *are matching, and* f' *and* g' *are matching, then* $\mathrm{f} \bullet \mathrm{g} \Leftrightarrow \mathrm{f}' \bullet \mathrm{g}'$.

Proof. Immediate from Proposition 3.3.6 and Proposition 2.5.6. ☐

Composition of saturated nets

With composition for nets defined and shown to be correct, composition for saturated nets will be considered next. The simple example below illustrates that relational composition is not sufficient as a notion of composition for saturated nets: the first two nets, which are saturated, compose to form the third; however, this net is not saturated; its saturation is pictured fourth.



In the following, it will be shown that composition for saturated nets is relational composition followed by saturation.

Definition 3.3.8. The *composition* $\sigma g \circ \sigma f$ of composable saturated nets σf and σg is defined as relational composition followed by saturation, as follows.

$$\sigma g \circ \sigma f \quad = \quad \sigma(\sigma f \bullet \sigma g)$$

The main idea is as follows. Since saturation must commute with composition for nets and for saturated nets, what the composition of σf and σg should be is the following:

- the saturation $\sigma(h \bullet k)$ for any pair of matching nets $h \Leftrightarrow f$ and $k \Leftrightarrow g$ (note that by Proposition 3.3.7 above, for any choice of h and k the composition $h \bullet k$ is equivalent).

By Proposition 3.3.2 a saturated net is the union of an equivalence class of nets. This means that the relational composition of two saturated nets f and g is the union of the following:

- the compositions $h \bullet k$ of all pairs of *matching* nets $h \Leftrightarrow f$ and $k \Leftrightarrow g$;
- the compositions $h \bullet k$ of all pairs of *non-matching* nets $h \Leftrightarrow f$ and $k \Leftrightarrow g$;
- and nothing else, since every link in σf occurs in some $h \Leftrightarrow f$, and every link in σg occurs in some $k \Leftrightarrow g$.

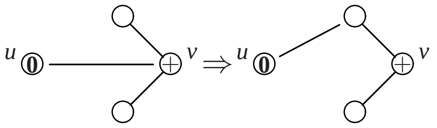
It is clear that the composition $\sigma g \circ \sigma f$ contains sufficient links, since the relational composition $\sigma f \bullet \sigma g$ contains at least one $h \bullet k$ for some matching pair h and k. To show that it does not contain too many links, it must be shown that the presence of prenets $h \bullet k$ for non-matching h and k is harmless. This is established below.

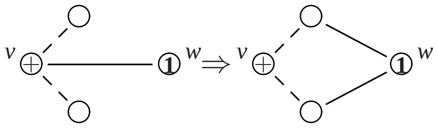
Lemma 3.3.9. *For composable nets* f *and* g *there are equivalent nets* $\mathrm{h} \Leftrightarrow \mathrm{f}$ *and* $\mathrm{k} \Leftrightarrow \mathrm{g}$ *that are matching, such that*

$$\mathrm{f}\bullet\mathrm{g} \quad \subseteq \quad \mathrm{h}\bullet\mathrm{k}\,.$$

Proof. Let Y be the target of f and source of g. The matching nets h and k will be generated by moving links towards the leaves in the central object Y. A measure of how close f and g are towards that goal is to consider, for all links $\langle u,v\rangle$ in f and $\langle v,w\rangle$ in g, the depth of the subtrees at v. The multiset of these depths, for all links in f and g combined, provides a convenient measure for induction (it should be noted that simpler measures are also possible).

The base case is where in f and g all links connect to leaves in Y (the measure is a multiset of zeroes). Otherwise, rewrite steps pushing a link down towards the leaves may be applied to f or g, or both simultaneously. Let v be a vertex in the target of f that is not an atom or unit; w.l.o.g. let v be a coproduct. To form nets f' and g', for any link $\langle u,v\rangle$ in f and $\langle v,w\rangle$ in g apply the following rewrite steps, replacing the former link by $\langle u,v0\rangle$ and the latter by $\langle v0,w\rangle$ and $\langle v1,w\rangle$.





If v is chosen such that there is at least one such link $\langle u,v\rangle$ or $\langle v,w\rangle$, then f' and g' are smaller, in the proposed measure, than f and g. The induction hypothesis gives nets h and k satisfying the following.

$$\mathrm{g} \Leftrightarrow \mathrm{f}' \Leftrightarrow \mathrm{f} \qquad\qquad \mathrm{k} \Leftrightarrow \mathrm{g}' \Leftrightarrow \mathrm{g} \qquad\qquad \mathrm{f}'\bullet\mathrm{g}' \subseteq \mathrm{h}\bullet\mathrm{k}\,.$$

To show that also $\mathrm{f}\bullet\mathrm{g} \subseteq \mathrm{f}'\bullet\mathrm{g}'$, let $\langle u,w\rangle$ be a link in $\mathrm{f}\bullet\mathrm{g}$ is due to links $\langle u,x\rangle$ and $\langle x,w\rangle$. If $x \neq v$ then, clearly, $\langle u,x\rangle$ and $\langle x,w\rangle$ are in f' and g' respectively, and $\langle u,w\rangle$ is in $\mathrm{f}'\bullet\mathrm{g}'$. Otherwise, if $x = v$, then the link $\langle u,x0\rangle$ is in f', and $\langle x0,w\rangle$ is in g'. Then, too, $\langle u,w\rangle$ is in $\mathrm{f}'\bullet\mathrm{g}'$. □

The following proposition then shows that this notion of composition is indeed the right one.

Proposition 3.3.10. *Composition of saturated nets satisfies*

$$\sigma[\![t]\!] \circ \sigma[\![s]\!] \;=\; \sigma[\![r]\!]$$

for some normal term r *equal to* $t \circ s$

Proof. Let $\mathrm{f} \Leftrightarrow [\![s]\!]$ and $\mathrm{g} \Leftrightarrow [\![t]\!]$ be matching nets. By Proposition 3.3.6 the equivalence below left holds, from which the equation below right follows by the completeness of saturation (Theorem 3.2.3).

$$\mathrm{f} \bullet \mathrm{g} \;\Leftrightarrow\; [\![r]\!] \qquad\qquad \sigma(\mathrm{f} \bullet \mathrm{g}) \;=\; \sigma[\![r]\!]$$

In addition, by the same theorem, $\sigma\mathrm{f} = \sigma[\![s]\!]$ and $\sigma\mathrm{g} = \sigma[\![t]\!]$. What remains to be shown is the following.

$$\sigma(\sigma\mathrm{f} \bullet \sigma\mathrm{g}) \quad = \quad \sigma(\mathrm{f} \bullet \mathrm{g})$$

One direction, $(\supseteq)$, follows because $\sigma\mathrm{f}$ contains f and $\sigma\mathrm{g}$ contains g, while both relational composition and saturation are monotone with respect to subset inclusion. For the other direction, it suffices to show the following.

$$\sigma\mathrm{f} \bullet \sigma\mathrm{g} \quad \subseteq \quad \sigma(\mathrm{f} \bullet \mathrm{g})$$

It will be shown that this inclusion follows from the fact that saturated nets are unions over equivalence classes (Proposition 3.3.2). Let $\langle u,w\rangle$ be a link in $\sigma\mathrm{f} \bullet \sigma\mathrm{g}$, originating in links $\langle u,v\rangle$ in $\sigma\mathrm{f}$ and $\langle v,w\rangle$ in $\sigma\mathrm{g}$. Then by Proposition 3.3.2 there are nets $\mathrm{f}' \Leftrightarrow \mathrm{f}$ and $\mathrm{g}' \Leftrightarrow \mathrm{g}$, respectively containing $\langle u,v\rangle$ and $\langle v,w\rangle$. For these nets, Lemma 3.3.9 gives equivalent, composable nets h and k such that $\mathrm{f}' \bullet \mathrm{g}' \subseteq \mathrm{h} \bullet \mathrm{k}$. Since $[\![s]\!] \Leftrightarrow \mathrm{h}$ and $[\![t]\!] \Leftrightarrow \mathrm{k}$, Proposition 3.3.6 gives a normal term r equal to $t \circ s$ such that $\mathrm{h} \bullet \mathrm{k} \;\Leftrightarrow\; [\![r]\!]$, and by completeness (Theorem 3.2.3) $\sigma(\mathrm{h} \bullet \mathrm{k}) = \sigma[\![r]\!]$. Combining the above, the following equation then shows that $\langle u,w\rangle$ is in $\sigma(\mathrm{f} \bullet \mathrm{g})$.

$$\langle u,w\rangle \quad \in \quad \mathrm{f}' \bullet \mathrm{g}' \quad \subseteq \quad \mathrm{h} \bullet \mathrm{k} \quad \subseteq \quad \sigma(\mathrm{h} \bullet \mathrm{k}) \quad = \quad \sigma[\![r]\!] \quad = \quad \sigma(\mathrm{f} \bullet \mathrm{g})$$

□

Corollary 3.3.11 (Characterising $\Sigma\Pi(\mathcal{C})$). *The category $\Sigma\Pi(\mathcal{C})$ is characterised by the following.*

- *Objects are given by the grammar*

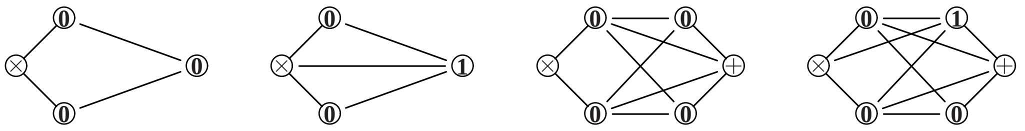
$$X \quad := \quad A \in \mathcal{C} \;\mid\; \mathbf{0} \;\mid\; \mathbf{1} \;\mid\; X + X \;\mid\; X \times X \;.$$

- *Morphisms are given by saturated nets.*
- *The identity morphism for an object X is the saturated net $\sigma(\mathrm{id}_X)$*
- *The composition of two composable saturated nets $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ is $\sigma\mathrm{g} \circ \sigma\mathrm{f}$.*

3.4 Correctness for saturated nets

A central part of any notion of proof net is a *correctness criterion*: a condition that identifies the proof nets among the proof structures (see also Section 1.3). Typically, such a condition is expected to be combinatorial, mainly to ensure that it is more informative, and possibly easier to verify, than a criterion provided by a translation procedure from proofs (i.e. the criterion that a structure is a net if and only if it is the translation of some proof). In the absence of the units, where sum–product nets are canonical, the correctness criterion is the switching condition, which determines whether a prenet is a net. For additive linear logic with units, the canonical proof objects are saturated nets. Here, a correctness criterion for saturated nets will be discussed, that separates the saturated nets from the arbitrary prenets.

Two conditions a saturated net must satisfy are immediately conspicuous: one, it must be connected, since it is obtained from a net by saturation; and two, it must be saturated. These two conditions are not sufficient: they are satisfied by all prenets that contain all possible links and are connected, which are not always saturated nets. For example, of the four (connected and saturated) prenets below, only the second and fourth are saturated nets.

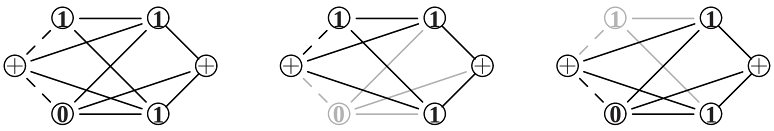


The problem is to distinguish a saturated net from a prenet formed by the union of several saturated nets. What separates these is that in a saturated net, all links can be obtained by saturation from a single net. The *neighbouring* relation, defined below, is used to verify whether one link may arise from another by saturation; informally, it relates links that occur together in the diagrams for the saturation steps.

Definition 3.4.1. The *neighbouring* relation $\frown$ over the links $\mathcal{R}$ in a prenet $(X,Y,\mathcal{R})$ is defined as the smallest symmetric relation satisfying

$$\langle vi, w\rangle \frown \langle v, w\rangle \qquad\qquad \langle v, wj\rangle \frown \langle v, w\rangle \ .$$

Since in a net a switching switches on exactly one link, a first attempt at a refined criterion for saturated nets would be to formalise the idea that if two links are switched on by the same switching, one must be introduced by saturation. This can be stated as follows: if two links are incompatible (Definition 2.3.3), $\langle v,w\rangle \# \langle x,y\rangle$, then they must be related in the reflexive–transitive closure of the neighbouring relation, $\langle v,w\rangle \frown^* \langle x,y\rangle$. However, this criterion does not suffice to characterise saturated nets: the prenet below left is not a saturated net, although it is connected, saturated, and all its links are related in $\frown^*$.



The first of the two switchings of the prenet above left, pictured to the right of it, suggests a refinement to the criterion. Although all links are (transitive) neighbours in the whole prenet, this no longer holds if the neighbouring relation is taken just over the links that are switched on. Thus, let $\frown_\varsigma$ denote the neighbouring relation $\frown$ restricted to links switched on by ς, and let $\frown_\varsigma^*$ be its reflexive–transitive closure.

Definition 3.4.2. A prenet is *close-knit* if for any switching ς

$$\varsigma \text{ ⏻ } \langle v,w\rangle \;\wedge\; \varsigma \text{ ⏻ } \langle x,y\rangle \quad\Rightarrow\quad \langle v,w\rangle \frown_\varsigma^* \langle x,y\rangle \;.$$

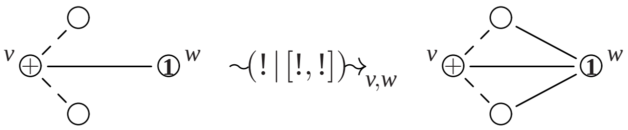
The correctness criterion will then be as follows.

Theorem 3.4.3 (Correctness of saturated nets)**.** *A prenet is a saturated net if it is connected, saturated, and close-knit.*

One direction of the proof is easily established.

Proposition 3.4.4. *A saturated net* $\sigma\mathsf{f}$ *is close-knit.*

Proof. Trivially, f is close-knit, since a switching ς for f switches on exactly one link. This is preserved in saturation, because any link added in a saturation step is a neighbour of an existing link. For example, in a saturation step $\mathsf{g} \sim\!(!\mid[!,!])\!\rightarrowtail_{v,w} \mathsf{g}'$, reproduced below, if $\varsigma \text{ ⏻ } \langle v0,w\rangle$ and $\varsigma \text{ ⏻ } \langle x,y\rangle$, also $\varsigma \text{ ⏻ } \langle v,w\rangle$, and if g is close-knit then $\langle v0,w\rangle \frown_\varsigma \langle v,w\rangle \frown_\varsigma^* \langle x,y\rangle$.



□

The other direction will be stated here, but not proved; the proof relies on the lemmata of the soundness proof for saturation, and will be completed in Section 4.9.

Proposition 3.4.5. *If a pre-net* h *is connected, saturated, and close-knit, it is a saturated net* σf.

For a connected, saturated, close-knit prenet h the proof of the proposition will give an actual net f such that $\sigma \mathrm{f} = \mathrm{h}$. This means it provides a 'de-saturation' algorithm that, together with the interpretation of a net as a term, constitutes a sequentialisation procedure—a method of translating a saturated net into a term that is inverse (up to term equality) to $\sigma[\![-]\!]$, translation followed by saturation. This is discussed in more detail in Section 4.9.

3.5 Complexity

In [23] Robin Cockett and Luigi Santocanale present an intricate decision procedure for the word problem of sum–product logic—the equational theory of Figure 2.4. The time complexity of this algorithm, in deciding equality of two cut-free terms $s,t : X \to Y$, is given in big-O notation as

$$\mathcal{O}\left((\mathit{hgt}(X) + \mathit{hgt}(Y)) \times |X| \times |Y|\right)$$

where $|X|$ denotes the size of the syntax tree of an object X, i.e. the number of vertices, and $\mathit{hgt}(X)$ denotes its height.

Here it will be argued that, with an appropriate implementation, the decision procedure provided by saturation slightly improves on this, having the following bound.

$$\mathcal{O}\left(|X| \times |Y|\right)$$

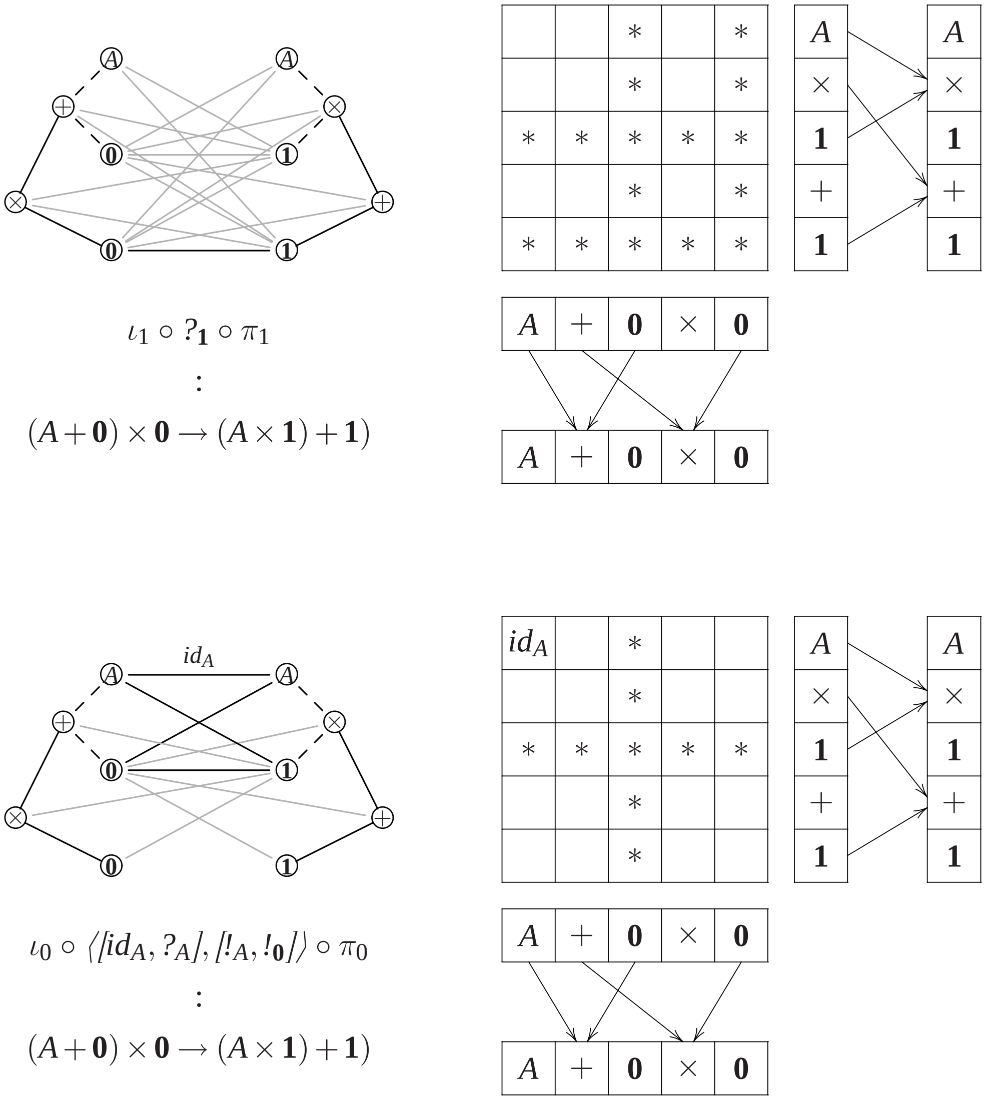
Starting with cut-free terms s and t of type $X \to Y$, the decision procedure would compute whether $\sigma[\![s]\!] = \sigma[\![t]\!]$ holds. This involves three steps: translating both terms to nets, saturating the nets, and comparing for equality.

An algorithm implementing these steps will be outlined. Firstly, for a net $(X,Y,\mathcal{R})$, the linking $\mathcal{R}$ is represented by a two-dimensional array of size $|X| \times |Y|$, whose entries are the labels of the links (i.e. strings representing $\mathcal{C}$-maps or $*$) or a null-value to describe the absence of links. The vertices in X and Y are the indices on the horizontal and vertical axes respectively, while the tree-structure of the objects is implemented by functions indicating parent vertices, children, and the type of a vertex (product, coproduct, initial, terminal, or an atom A). An impression of this representation is given in Figure 3.9. The illustration shows two nets, with their saturation added in grey, along with two corresponding terms, and an array-representation of the saturated nets, on the right; the arrows between the object arrays represent the *parent*-function.

The translation $[\![-]\!]$, from a cut-free term t into a net in this representation, can be implemented as follows. An easy induction on t shows that $|t|$, the size of t in the number of term constructors, is equal to or smaller than $|X|\times|Y|$ (this is not the case for proof terms t with cuts). The objects X and Y of t, if not explicitly present, are extracted by a simple walk over t. The vertices of a syntax tree for X can be indexed, and their parent-function and children-function extracted, in linear time in $|X|$, each by a simple walk over X. To translate t into a net can be done by a function walking over the term t, while simultaneously keeping track of the indices in $|X|$ and $|Y|$ (via the children and parent functions). The output of this function would be to update the corresponding entry in the linking array whenever a $\mathcal{C}$-map or unit map is encountered, and to add the positions $\mathtt{(x,y)}$ of unit links $\langle x,y\rangle$ to a stack $\mathtt{s}$. Each step in this algorithm consists of nothing more than a few array lookups and updates, plus a single stack push, and would thus be constant time. The time complexity of the algorithm as a whole is then linear in the size of the term $|t|$, and hence smaller than $|X|\times|Y|$.

Saturation steps can be implemented as follows. Popping an item $\mathtt{(x,y)}$ from the stack $\mathtt{s}$ gives the position in the matrix of a recently added link. The parent and children functions give the indices of links that may need to be added in the saturation step; since both have maximally two children, one parent, and one sibling (which must be inspected for rewrite steps of the kind $\sim\!(\langle ?,?\rangle \mid ?)\!\rightsquigarrow$ and $\sim\!([!,!] \mid !)\!\rightsquigarrow$), at most eight positions are accessed. The positions of newly added links are then pushed onto the stack. Consisting of a constant number of array lookups and updates, and stack pops and pushes, a saturation step is thus performed in constant time. For the complete saturation procedure, each link in the saturation appears on the stack only once, when it is added to the matrix. The complexity of saturation is then bounded, by a constant factor, by the number of entries in the linking array, $|X|\times|Y|$.

Finally, comparing the two saturated nets $\sigma[\![s]\!]$ and $\sigma[\![t]\!]$ for equality is done by a simple equality test of the two linking arrays. The complete process of translation, saturation, and equality testing, for cut-free terms, is thus performed in time bounded by $\mathcal{O}(|X|\times|Y|)$. (The complexity in the presence of cuts has not been evaluated.)



$$\iota_1 \circ ?_{\mathbf{1}} \circ \pi_1$$
$$:$$
$$(A+\mathbf{0})\times\mathbf{0} \to (A\times\mathbf{1})+\mathbf{1}$$

$$\iota_0 \circ \langle [id_A, ?_A], [!_A, !_{\mathbf{0}}]\rangle \circ \pi_0$$
$$:$$
$$(A+\mathbf{0})\times\mathbf{0} \to (A\times\mathbf{1})+\mathbf{1}$$

Figure 3.9: Two saturated nets in different representations

Chapter 4

The soundness proof

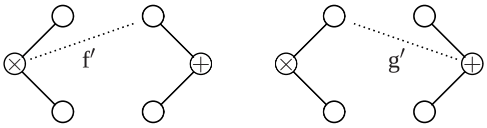
4.1 Introduction

This chapter will concern, mainly, the proof of Theorem 3.2.4, that saturation ($\rightsquigarrow$) is sound as a decision procedure for sum–product categories. The proof itself, presented in Section 4.7, will proceed by induction on the source and target object of a pair of parallel nets, and will rely on a body of lemmata, carefully constructed over the course of the chapter. At the end, the two further outstanding proofs will be completed: firstly, in Sections 4.8, the proof of Proposition 3.3.2, that saturated nets are unions of equivalence classes of nets; and secondly, in Section 4.9, the proof of Proposition 3.4.5, which describes the correctness criterion for saturated nets. In addition to the proofs, this chapter presents one new addition to the main body of results on saturated nets: a sequentialisation algorithm, also in Section 4.9.

The soundness proof will be outlined below. To be proven is that two nets f and g with the same saturation $\sigma\mathrm{f} = \sigma\mathrm{g} = (X, Y, \mathcal{R})$ are equivalent, that is $\mathrm{f} \Leftrightarrow \mathrm{g}$. The proof is by induction on X and Y, with the above statement as the induction hypothesis. As a first overview, there will be three cases:

- one of X and Y is an atom or unit,
- X is a coproduct or Y is a product, and
- X is a product and Y a coproduct.

The first two cases are relatively straightforward, and will be treated in Section 4.2. The main body of the proof is concerned with the third case, which is that of nets of the form $f = f';\iota_j$ and $g = \pi_i;g'$ as illustrated below.



For this third case, there are three primary obstacles to overcome, which will be outlined below.

Inductive saturation

To apply the induction hypothesis it must be possible to relate, e.g., a saturated net $\sigma(f;\iota_0)$, to the saturation of its component net, σf. This is addressed by providing an alternative characterisation of a saturated net σf, by induction on the construction of the net f. In Section 4.2, Lemma 4.2.3 presents the case for basic nets, and Lemma 4.2.5 that for nets of the form $\langle f, g\rangle$ and $[f, g]$. The case for nets $\pi_i;f$ and $f;\iota_j$, Lemma 4.4.1, will be the most involved. Section 4.3 will provide supporting material for this lemma, which will itself be presented and discussed in Section 4.4.

Nets over different projections and injections

The second obstacle is that nets constructed over different projections and injections, e.g. $f';\iota_0$ and $\pi_0;g'$, as illustrated above, but also $f';\iota_0$ and $h';\iota_1$, may have the same saturation. Naturally, in such a case the induction hypothesis cannot be applied to $\sigma f'$ and $\sigma g'$. This problem will be addressed in Section 4.5. It is shown that if $f';\iota_0$ and $\pi_0;g'$ have the same saturation, then this saturation must contain at least one initial link $\langle v, \varepsilon\rangle$ (and one terminal link $\langle \varepsilon, w\rangle$). Then Lemma 4.5.1 will show that since $\sigma(f';\iota_0)$ contains the link $\langle v, \varepsilon\rangle$, there must be a net f'' equivalent to $f';\iota_0$, also containing $\langle v, \varepsilon\rangle$. From the presence of this link it can then be deduced that f'' is left-constructible, and over which projection it is constructed. Since the saturation of $\pi_0;g'$ is the same as that of $f';\iota_0$, the same argument shows that $\pi_0;g'$ is equivalent to a net g'', containing the same link $\langle v, \varepsilon\rangle$, and constructed over the same projection as f''. Then the induction hypothesis can be applied to the deconstructions of f'' and g''.

Major reconstruction

The third obstacle is that nets constructed over the same projection or injection, e.g. $f;\iota_0$ and $g;\iota_0$, may have the same saturation, while their components, f and g, do not. An illustration of this is provided in Figure 4.5 on page 94. In Section 4.6 it will be shown how to transform the net $f;\iota_0$ into an equivalent net $h;\iota_0$ such that h does have the same saturation as g, so that the induction hypothesis can be applied to g and h. The formal details are recorded in Lemma 4.6.3.

Finale

The soundness proof is concluded in Section 4.7. Then in Section 4.8 and 4.9 the two remaining proofs from Chapter 3 are completed.

4.2 The first two cases

The first case of the soundness proof concerns parallel nets whose source or target is an atom or unit. For nets with source X and target Y, this gives six possibilities, that are pairwise dual. Four are immediate: if X is an atom or **1**, or dually if Y is an atom or **0**, illustrated below, it is easily observed that no rewrite or saturation steps apply.

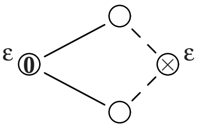
For such nets f and g, it follows that if $\sigma f = \sigma g$ then $f = g$.

For the remaining two cases, nets with source object **0** will be called *initial*, and with target **1**, *terminal*. The links in an initial net $(\mathbf{0}, Y, \mathcal{R})$ can move up and down the syntax tree of Y essentially without hindrance. From this, the lemma below follows—and the one after as well. In the next lemma, recall that two nets are *parallel* if they have the same source objects and the same target objects.

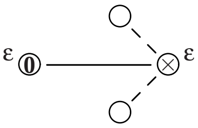
Lemma 4.2.1. *All parallel initial nets are equivalent, as are all parallel terminal nets.*

Proof. It will be shown, by induction on the construction of an initial net $f = (\mathbf{0}, Y, \mathcal{R})$, that f is equivalent to $?_Y$, from which the statement follows by transitivity.

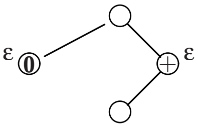
If f is basic, $f = ?_Y$. With **0** as source object f cannot be left-constructible. If f is right-constructible, for $f = \langle f_0, f_1\rangle$ the induction hypothesis gives $f_i \Leftrightarrow ?_{Y_i}$ for $i \in \{0,1\}$. Then $\langle f_0, f_1\rangle \Leftrightarrow \langle ?,?\rangle$, and by a single rewrite step, below, $\langle ?,?\rangle \Leftrightarrow ?$.



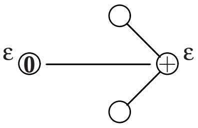
$=\!\![\langle ?,?\rangle \,|\, ?]\!\!\Rightarrow_{\varepsilon,\varepsilon}$



Next, if $f = f';\iota_j$ the induction hypothesis gives $f' \Leftrightarrow ?_{Y_j}$. Then by a single rewrite step, below, $f';\iota_j \Leftrightarrow ?;\iota_j \Leftrightarrow ?$.



$=\!\![?;\iota_0 \,|\, ?]\!\!\Rightarrow_{\varepsilon,\varepsilon}$



The case for terminal nets is dual. □

The above lemma confirms, syntactically, that **0** and **1** are initial and terminal objects, respectively, in the category of nets modulo equivalence, and that consequently any decision procedure for initial or terminal nets is sound: it is impossible to identify too many of them. That **0** and **1** are also initial and terminal in the category of saturated nets is a matter of completeness. It follows from Theorem 3.2.3 that all parallel initial or terminal nets have the same saturation. It will be useful to describe these saturated nets explicitly.

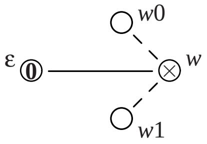
Definition 4.2.2. A prenet is *full* if it contains all possible unit links (but no atomic links), i.e. if it is of the form

$$(X, Y, \{\langle v, *, w\rangle \mid X_v = \mathbf{0} \text{ or } Y_w = \mathbf{1}\})\ .$$

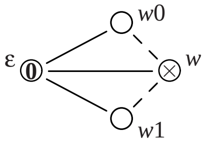
Clearly, for a given source and target object there is precisely one such prenet.

Lemma 4.2.3. *The saturation of initial and terminal nets is full.*

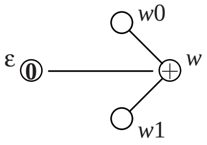
Proof. First, it will be shown that the saturation of a net $?_Y = (\mathbf{0}, Y, *)$ is full. Let $\sigma ?_Y = (\mathbf{0}, Y, \mathcal{R})$. It follows from the saturation steps that if a link $\langle \varepsilon, *, w\rangle \in \mathcal{R}$ connects to a vertex w with children $w0, w1 \in \mathrm{pos}(Y)$, then also $\langle \varepsilon, *, w0\rangle, \langle \varepsilon, *, w1\rangle \in \mathcal{R}$, as follows. If Y_w is a product,



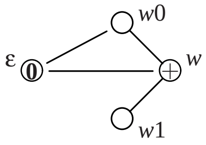
$\rightsquigarrow\!(?\,|\,\langle ?, ?\rangle)\!\rightsquigarrow_{\varepsilon,w}$



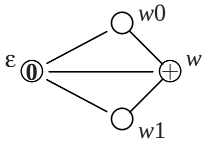
and if Y_w is a coproduct,



$\rightsquigarrow\!(?\,|\,?;\iota_0)\!\rightsquigarrow_{\varepsilon,w}$



$\rightsquigarrow\!(?\,|\,?;\iota_1)\!\rightsquigarrow_{\varepsilon,w}$

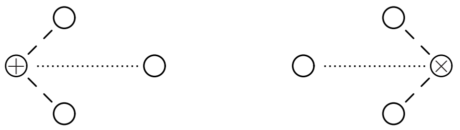


Then since $?_Y$ contains the link $\langle \varepsilon, *, \varepsilon\rangle$, its saturation is full:

$$\mathcal{R} = \{\langle \varepsilon, *, w\rangle \mid w \in \mathrm{pos}(Y)\}\ .$$

By Lemma 4.2.1 any initial net f with target Y is equivalent to $?_Y$. Then by completeness (Theorem 3.2.3) f and $?_Y$ have the same saturation, and so σf is full. The case for terminal nets is dual. □

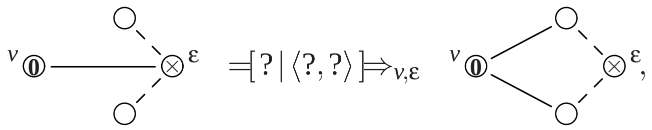
The second case of the soundness proof concerns nets whose source is a coproduct or whose target is a product; call these *coproduct nets* and *product nets*, respectively, illustrated below.



Product nets are not just the nets of the form $\langle \mathrm{f},\mathrm{g}\rangle$, since they need not be right-constructible. However, it is easily shown that they are equivalent to such nets, and that dually coproduct nets are equivalent to nets of the form $[\mathrm{f},\mathrm{g}]$.

Lemma 4.2.4. *A product net* g *is equivalent to a net* $\langle \mathrm{g}_0,\mathrm{g}_1\rangle$. *A coproduct net* f *is equivalent to a net* $[\mathrm{f}_0,\mathrm{f}_1]$.

Proof. Let $\mathrm{g} = (X, Y, \mathcal{R})$ be a product net, i.e. Y is a product. By the definition of the constructors, g is of the form $\langle \mathrm{g}_0,\mathrm{g}_1\rangle$ unless it contains initial links $\langle v, *, \varepsilon\rangle$ for some v, connecting to the root of Y. By applying the following rewrite step for any such v,

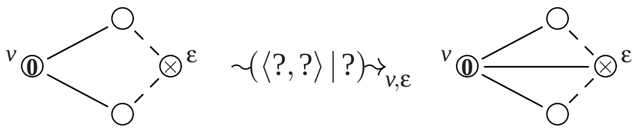


a net of the form $\langle \mathrm{g}_0,\mathrm{g}_1\rangle$ is obtained from g. The case for coproduct nets is dual. □

As equivalent nets have the same saturation, the above lemma means that a saturated product net $\sigma\mathrm{g}$ can always be described as the saturation of a net $\langle \mathrm{g}_0,\mathrm{g}_1\rangle$. Relating the latter saturation to those of its components, $\sigma\mathrm{g}_0$ and $\sigma\mathrm{g}_1$, will allow induction on saturated nets. To this end, consider a saturation path for $\langle \mathrm{g}_0,\mathrm{g}_1\rangle$ that first applies all possible saturation steps to g_0 and g_1 individually, as follows.

$$\langle \mathrm{g}_0,\mathrm{g}_1\rangle \rightsquigarrow \dots \rightsquigarrow \langle \sigma\mathrm{g}_0,\sigma\mathrm{g}_1\rangle \rightsquigarrow \dots \rightsquigarrow \sigma\langle \mathrm{g}_0,\mathrm{g}_1\rangle$$

The only saturation steps that can be applied to $\langle \sigma\mathrm{g}_0,\sigma\mathrm{g}_1\rangle$, in the irreflexive variant $\rightsquigarrow^-$, are those of the form below.



That the second part of the saturation path above contains only such steps follows from the observation that the newly added link $\langle v,\varepsilon\rangle$ does not trigger any new saturation steps: the only step that can be applied to it, is the reverse step to the one that introduced it. These observations are summarised by the lemma below.

Lemma 4.2.5. *If* $[\sigma \mathrm{f}_0, \sigma \mathrm{f}_1] = (X, Y, \mathcal{R})$ *then* $\sigma[\mathrm{f}_0, \mathrm{f}_1] = (X, Y, \mathcal{R} \cup \mathcal{S})$ *where*

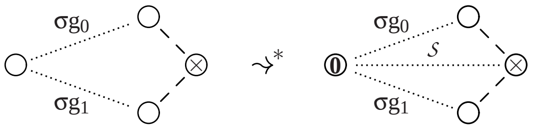
$$\mathcal{S} \;=\; \{\langle \varepsilon, *, w\rangle \mid Y_w = \mathbf{1},\ \langle 0, *, w\rangle \in \mathcal{R},\ \langle 1, *, w\rangle \in \mathcal{R}\}\ .$$

Dually, if $\langle \sigma \mathrm{g}_0, \sigma \mathrm{g}_1\rangle = (X, Y, \mathcal{R})$ *then* $\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle = (X, Y, \mathcal{R} \cup \mathcal{S})$ *where*

$$\mathcal{S} \;=\; \{\langle v, *, \varepsilon\rangle \mid X_v = \mathbf{0},\ \langle v, *, 0\rangle \in \mathcal{R},\ \langle v, *, 1\rangle \in \mathcal{R}\}\ .$$

Proof. It can be observed (following the above reasoning) that the saturation path from $\langle \sigma \mathrm{g}_0, \sigma \mathrm{g}_1\rangle$ to $\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle$ consists of the steps $\prec\!(\langle ?, ?\rangle \mid ?)\!\rightarrowtail_{v,\varepsilon}$ for those v such that both $\sigma \mathrm{g}_0$ and $\sigma \mathrm{g}_1$ have a link $\langle v, \varepsilon\rangle$. The case for $[\mathrm{f}_0, \mathrm{f}_1]$ is dual. □

Crucially, in the above lemma the links in $\mathcal{S}$ are all of the form $\langle v, \varepsilon\rangle$, and thus easily separated from those originally belonging to $\sigma \mathrm{g}_0$, which are all of the form $\langle v, 0w\rangle$, or those belonging to $\sigma \mathrm{g}_1$, which are of the form $\langle v, 1w\rangle$.



It follows that by simply restricting $\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle$ to the subprenet between ε and 0, or between ε and $\mathbf{1}$, the saturations of g_0 and g_1 can be recovered.

Lemma 4.2.6. *Saturation of product and coproduct nets satisfies:*

$$(\sigma[\mathrm{f}_0, \mathrm{f}_1])_{i,\varepsilon} \;=\; \sigma \mathrm{f}_i \qquad (\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle)_{\varepsilon,i} \;=\; \sigma \mathrm{g}_i\ .$$

Proof. By Lemma 4.2.5,

$$(\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle)_{\varepsilon,i} \;=\; \langle \sigma \mathrm{g}_0, \sigma \mathrm{g}_1\rangle_{\varepsilon,i}\ ,$$

and by the definition of the constructors,

$$\langle \sigma \mathrm{g}_0, \sigma \mathrm{g}_1\rangle_{\varepsilon,i} \;=\; \sigma \mathrm{g}_i\ .$$

The case for coproduct nets is dual. □

These two lemmata suffice to complete the case for product and coproduct nets in the soundness proof. For parallel product nets f and g with the same saturation, Lemma 4.2.4 gives equivalent nets $\langle \mathrm{f}_0, \mathrm{f}_1\rangle$ and $\langle \mathrm{g}_0, \mathrm{g}_1\rangle$ respectively. By Lemma 4.2.6

$$\sigma \mathrm{f}_i \;=\; (\sigma\langle \mathrm{f}_0, \mathrm{f}_1\rangle)_{\varepsilon,i} \;=\; (\sigma\langle \mathrm{g}_0, \mathrm{g}_1\rangle)_{\varepsilon,i} \;=\; \sigma \mathrm{g}_i$$

for $i \in \{0, 1\}$. The induction hypothesis of the soundness proof gives $\mathrm{f}_i \Leftrightarrow \mathrm{g}_i$, and the equivalences below follow.

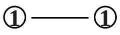
$$\mathrm{f} \;\Leftrightarrow\; \langle \mathrm{f}_0, \mathrm{f}_1\rangle \;\Leftrightarrow\; \langle \mathrm{g}_0, \mathrm{g}_1\rangle \;\Leftrightarrow\; \mathrm{g}$$

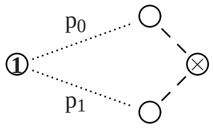
4.3 Pointed and copointed nets

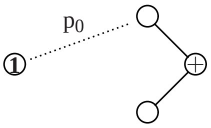
In a category, a *point* is a map out of a terminal object. Points are also known as *constants*, in particular in the category of sets. An object P that has a point $p : \mathbf{1} \to P$ will be called *pointed*. Note that this is non-standard: more commonly, a pointed object is taken to be a pair (P, p). However, for the present purpose it will mostly be relevant whether an object has a point, but not which one exactly; moreover, for a pointed object a point is easily reconstructed. In free sum–product categories, points and pointed objects are given by the following grammars, respectively.

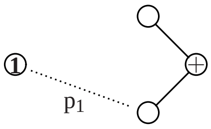
$$p \;:=\; ! \mid \langle p, p\rangle \mid \iota_j \circ p \qquad\qquad P \;:=\; \mathbf{1} \mid P \times P \mid P + X \mid X + P$$

Both are illustrated by the construction of nets with source object $\mathbf{1}$, below.







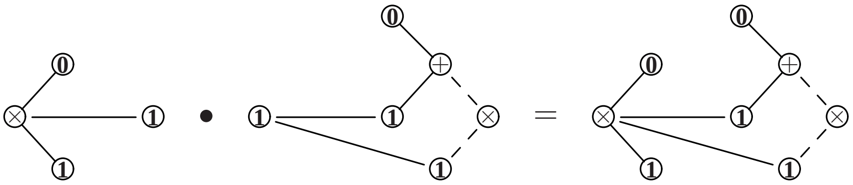


In the dual notions, a *copoint* is a map into $\mathbf{0}$, and an object that has a copoint is *copointed*. These are given by the following grammars.

$$q \;:=\; ? \mid [q, q] \mid q \circ \pi_i \qquad\qquad Q \;:=\; \mathbf{0} \mid Q + Q \mid Q \times X \mid X \times Q$$

Note that a pointed object may have more than one point, and similarly for copointed objects, but that an object is never both pointed and copointed. Another useful observation is that pointed objects are precisely those for which every switching switches on at least one terminal node. Dually, copointed objects are those whose co-switchings switch on at least one initial node. Also, in $\Sigma\Pi(\varnothing)$, the free sum–product completion of the empty category, where atoms are absent, these grammars are similar to the evaluation of truth or falsity in boolean expressions; in this category every object is either pointed or copointed.

A point p into a pointed object P composes with terminal maps to form a *pointed* map $p \circ !_X$ from any object X into P (thus, in the present non-standard definition, pointed objects are precisely the weakly terminal ones). For nets, the (relational) composition of a terminal map with a point gives a net with only terminal links connecting to the left root, as in the example below. (Note that since their common object is a single leaf, such nets are always matching—see also Section 3.3.)



Call initial links of the form $\langle v,*,\varepsilon\rangle$ and terminal links of the form $\langle\varepsilon,*,w\rangle$ *rooted*.

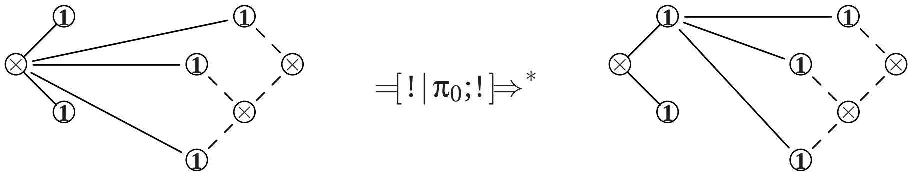
Definition 4.3.1. A prenet is *pointed* if it contains only rooted terminal links, and *copointed* if it contains only rooted initial links.

By this definition, pointed nets with a given source object X, and copointed nets with target Y, are described by the following grammars over constructors.

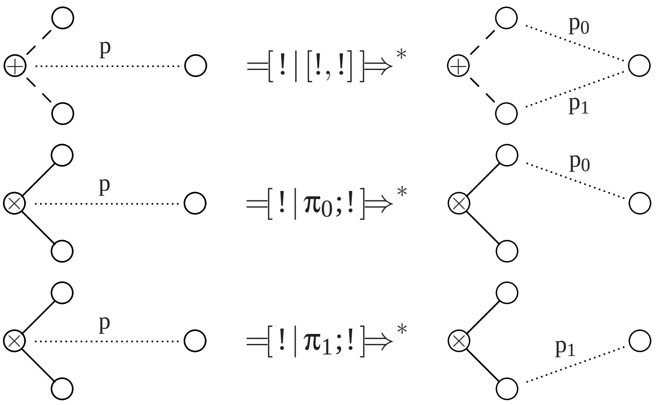
$$\mathrm{p} \;:=\; (X,\mathbf{1},*)\mid \mathrm{p};\iota_j \mid \langle \mathrm{p},\mathrm{p}\rangle \qquad\qquad \mathrm{q} \;:=\; (\mathbf{0},Y,*)\mid [\mathrm{q},\mathrm{q}]\mid \pi_i;\mathrm{q}$$

The definition restricts pointed and copointed nets to a convenient syntactic form, but other, equivalent nets may also correspond to pointed morphisms. In other words, every pointed map is described by some pointed net, but not every net that describes this map is pointed.

Since pointed nets consist of terminal links with a common source, these can be moved around in tandem, for example as follows.



This way, for example, a pointed net p with a coproduct source is equivalent to a net $[\mathrm{p}_0,\mathrm{p}_1]$. More generally, this can be applied to a partial pointed net p as well, if it is a sub-prenet of a net g, i.e. $\mathrm{p}\subseteq\mathrm{g}$. For example, if the source of g is a coproduct, it is equivalent to a net g' with a sub-prenet $[\mathrm{p}_0,\mathrm{p}_1]$ (this was used in the proof of Lemma 4.2.4). Here, if p is the prenet $(X,P,\mathcal{R})$ and $X=X_0+X_1$, then $[\mathrm{p}_0,\mathrm{p}_1]$ is the pre-net $(X,P,0\cdot\mathcal{R}\;\cup\;1\cdot\mathcal{R}\,)$. The pre-nets p_0 and p_1 are $(X_0,P,\mathcal{R})$ and $(X_1,P,\mathcal{R}\,)$



p
$=\!\!|[\,!\,|\,[!,!]\,]\!\!\Rightarrow^*$
p0
p1
p
$=\!\!|[\,!\,|\,\pi_0;!\,]\!\!\Rightarrow^*$
p0
p
$=\!\!|[\,!\,|\,\pi_1;!\,]\!\!\Rightarrow^*$
p1

Figure 4.1: Synchronised equivalence steps

respectively; as an artefact of the way vertices are addressed, the nets p, p_0, and p_1 share the same linking $\mathcal{R}$. Applying an equivalence step to all links in a partial pointed or copointed net will be called a *synchronised equivalence step*, illustrated in Figure 4.1; informally, this will also be referred to as *moving* pointed and copointed nets up and down a syntax tree.

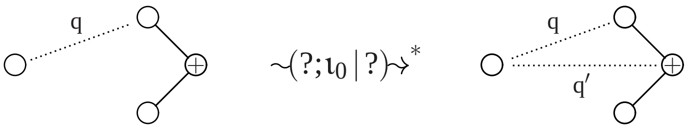


Figure 4.2: A synchronised saturation step

A similar notion will be that of *synchronised saturation steps*: the application of a saturation step to all links in a pointed or copointed sub-prenet, as illustrated in Figure 4.2. In the illustration, several saturation steps of the form $\sim\!(?;\iota_0\,|\,?)\!\!\rightarrowtail_{v,\varepsilon}$ are grouped together. It is then easily seen that if a saturated net σf has a sub-pre-net $(q;\iota_0) \subseteq \sigma f$ with q copointed, it must also have the copointed sub-pre-net $q' \subseteq \sigma f$—and vice versa. For easy reference, there is the following lemma.

Lemma 4.3.2. *The saturation of a copointed prenet* $q = (Q, Y, \mathcal{R})$ *contains a sub-pre-net* $(Q, Y, \mathcal{R} \cdot w) \subseteq \sigma q$ *for any vertex w in Y. Dually, for a pointed pre-net* $p = (X, P, \mathcal{R})$, *for any v in X there is a sub-prenet* $(X, P, v \cdot \mathcal{R}) \subseteq \sigma p$.

Proof. The copointed pre-net q is a collection of initial links $\langle v, \varepsilon \rangle$. For each such link, by Lemma 4.2.3 the saturation of the initial subnet between v and ε in σq is full, and contains an initial link $\langle v, w \rangle$ for any w in Y. It follows that $\mathcal{R} \cdot w$ is a subset of the linking of σq. The case for p follows by duality. ☐

A categorical morphism can be both pointed and copointed; such maps will be called *bipointed* here. Bipointed maps feature heavily in the decision procedure of Cockett and Santocanale [23]—where they are called *disconnects*—because of the following property: there is precisely one bipointed map from a copointed object Q to a pointed object P, and none between other objects. The uniqueness property is easily observed from the fact that in the diagram below the copoint q and the point p are arbitrary.

$$\begin{tikzcd}
Q \arrow[r, "q"'] \arrow[rr, bend left, "!"] & \mathbf{0} \arrow[r, "!"] \arrow[r, "?"'] \arrow[rr, bend right, "?"'] & \mathbf{1} \arrow[r, "p"] & P
\end{tikzcd}$$

The corresponding notion for nets will again be restricted to a syntactically useful form.

Definition 4.3.3. A net $(Q,P,\mathcal{R})$ is *bipointed* if it is pointed or copointed, and moreover its source object Q is copointed and its target object P is pointed.

The uniqueness property of categorical bipointed morphisms carries over to nets and saturated nets in the following way: any parallel bipointed nets are equivalent, and have the same saturation, which is full. Figure 4.3 shows in detail an equivalence between a copointed net and a parallel pointed one. In the first two steps the copointed net, consisting of the links $\langle 00,\varepsilon\rangle$ and $\langle 1,\varepsilon\rangle$, is moved down from the right root to the two terminal objects of the target tree, vertices 00 and 1. In the resulting net, the subnet highlighted in picture four, between the left root and the bottom right node, is a terminal net. This subnet rewrites into a basic net, consisting of a single rooted terminal link, following Lemma 4.2.1 (picture five). The other subnet, highlighted in the sixth diagram, is also a terminal net, and likewise rewrites to a single link in the next diagram. The result, in the final picture, is a pointed net. The following lemma generalises the above reasoning to the case where one of the nets is partial.

Lemma 4.3.4. *For parallel prenets* p *and* q, *if* p *is a pointed partial net and* q *a copointed net, there is a net* f *such that* $\mathrm{p}\subseteq\mathrm{f}$ *and* $\mathrm{q}\Leftrightarrow\mathrm{f}$. *Dually, if* p *is a pointed net and* q *a copointed partial net then there is a net* g *such that* $\mathrm{p}\Leftrightarrow\mathrm{g}$ *and* $\mathrm{q}\subseteq\mathrm{g}$.

Proof. The case for f will be shown; that for g is dual. The argumentation is as above. Moving the copointed subnet q down proceeds inductively, guided by the construction of the partial net p, as described by Proposition 2.4.4. Since p is pointed, it is either empty, basic, or right-constructible.

- If p is empty then let f be q; trivially, $\mathrm{p}\subseteq\mathrm{f}$ and $\mathrm{q}\Leftrightarrow\mathrm{f}$.
- If p is basic it is the net $(X,\mathbf{1},*)$. Let $\mathrm{f}=(X,\mathbf{1},*)$ as well; that $\mathrm{p}\subseteq\mathrm{f}$ is immediate, and since q is a terminal net, it is equivalent to f by Lemma 4.2.1.
- If p is a partial net $\langle \mathrm{p}_0,\mathrm{p}_1\rangle$, rewrite the copointed net q to the equivalent net $\langle \mathrm{q}_0,\mathrm{q}_1\rangle$ by moving it down from the right root. For $i\in\{0,1\}$ the induction hypothesis, applied to p_i and q_i, gives a net f_i with $\mathrm{p}_i\subseteq\mathrm{f}_i$ and $\mathrm{q}_i\Leftrightarrow\mathrm{f}_i$. Let f be $[\mathrm{f}_0,\mathrm{f}_1]$. The equations below follow.

$$\mathrm{p} \;=\; \langle \mathrm{p}_0,\mathrm{p}_1\rangle \;\subseteq\; \langle \mathrm{f}_0,\mathrm{f}_1\rangle \;=\; \mathrm{f} \qquad\qquad \mathrm{q} \;\Leftrightarrow\; \langle \mathrm{q}_0,\mathrm{q}_1\rangle \;\Leftrightarrow\; \langle \mathrm{f}_0,\mathrm{f}_1\rangle \;=\; \mathrm{f}$$

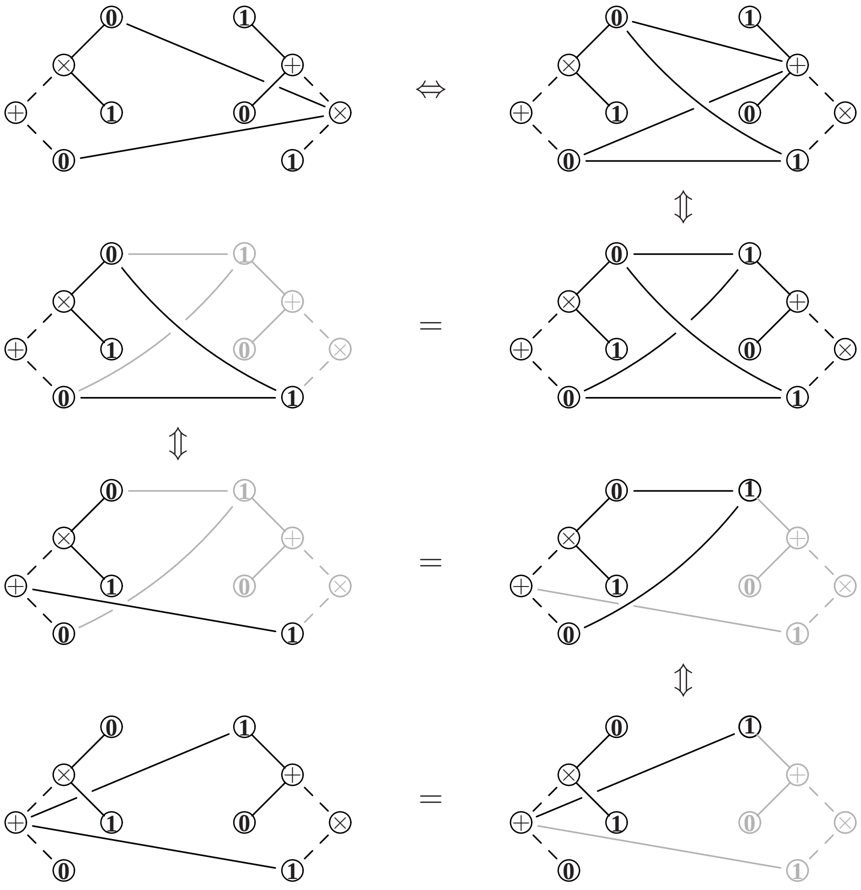


Figure 4.3: Transforming a copointed net into a pointed net

- If $\mathrm{p} = \mathrm{p}';\iota_j$ then form $\mathrm{q}';\iota_j \Leftrightarrow \mathrm{q}$ by moving the links in q down from the root, to the vertex j. The induction hypothesis for p' and q' gives f', with $\mathrm{p}' \subseteq \mathrm{f}'$ and $\mathrm{q}' \Leftrightarrow \mathrm{f}'$. Let f be $\mathrm{f}';\iota_j$, and the equations below follow.

$$\mathrm{p} \;=\; \mathrm{p}';\iota_j \;\subseteq\; \mathrm{f}';\iota_j \;=\; \mathrm{f} \qquad\qquad \mathrm{q} \;\Leftrightarrow\; \mathrm{q}';\iota_j \;\Leftrightarrow\; \mathrm{f}';\iota_j \;=\; \mathrm{f}$$

□

The equivalence of parallel bipointed nets is a direct consequence.

Lemma 4.3.5. *Any two parallel bipointed nets are equivalent.*

Proof. Let f and g be parallel bipointed nets. If one is pointed and the other copointed, Lemma 4.3.4 proves their equivalence immediately. If both are pointed, there is a parallel copointed net h because the common source object of f and g is copointed. The previous argument then gives f ⇔ h ⇔ g. The case where both nets are copointed is dual. □

Next, it will be shown that the saturation of a bipointed net is full. An example, of saturating a copointed net with a pointed target, is illustrated in Figure 4.4. (In the

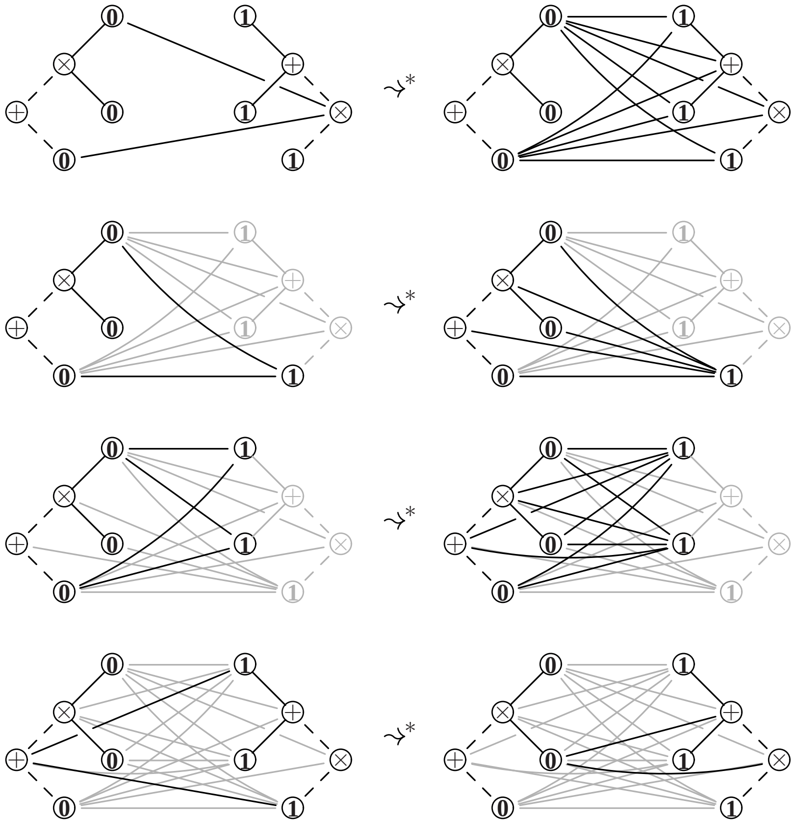


Figure 4.4: Saturating a bipointed net

illustration, the first figure on each line displays the same pre-net as the last figure of the previous line, but with different links highlighted.) Firstly, the copointed net is moved down to all vertices in the target object. This forms a terminal net between the left root and any vertex with a terminal object, as highlighted in the third diagram for the bottom right vertex; filling it in gives the fourth picture. The fifth and sixth diagram fill in the two terminal nets formed by the left root and the other two target vertices with terminal objects. At this point, the pre-net contains all possible terminal links, and all initial links except the two highlighted in the last diagram. These can be added by repeating the above procedure for the pointed net highlighted in the seventh diagram. The argument is formalised in the following two lemmata.

Lemma 4.3.6. *The saturation of a pointed (respectively copointed) net contains all initial (respectively terminal) links.*

Proof. Let $\mathsf{f} = (Q, Y, \mathcal{R})$ be a copointed net; the case for pointed nets is dual. It must be shown that if $Y_w = \mathbf{1}$ and $v \in \mathrm{pos}(Q)$ then the terminal link $\langle v, w\rangle$ is in $\mathcal{R}$. For the vertex w, Lemma 4.3.2 gives a sub-pre-net $(Q, P, \mathcal{R} \cdot w) \subseteq \sigma\mathsf{f}$. Since w is $\mathbf{1}$ this gives a terminal subnet $(Q, \mathbf{1}, \mathcal{R}) \subseteq (\sigma\mathsf{f})_{\varepsilon,w}$, whose saturation is full (by Lemma 4.2.3). ☐

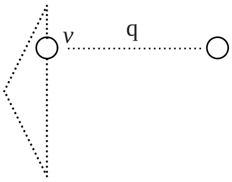
Lemma 4.3.7. *The saturation of a bipointed net is full.*

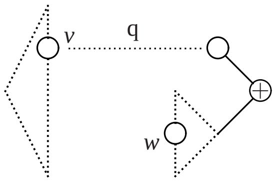
Proof. Let $\mathsf{f} = (Q, P, \mathcal{R})$ be bipointed and copointed; the case for pointed nets is dual. By Lemma 4.3.6 above, $\sigma\mathsf{f}$ contains all possible terminal links. Then since P is pointed, it contains a pointed subnet: pointed objects are precisely those that admit a point, and since the prenet at this stage contains all possible terminal links, it must contain also the point that P admits. Again by Lemma 4.3.6, $\sigma\mathsf{f}$ contains also all initial links, and must be full. ☐

4.4 Saturation via construction

The properties of pointed and copointed nets established in the previous section will be used to characterise the saturation of nets of the form $\pi_i;\mathsf{f}$ and $\mathsf{f};\iota_j$ in terms of $\sigma\mathsf{f}$, in the upcoming Lemma 4.4.1. This lemma will form the basis of the proof of the present case in the soundness proof, concerning parallel nets from a product into a coproduct. Together with Lemma 4.2.3 and Lemma 4.2.5, which describe the saturation of initial and terminal nets and, respectively, product and coproduct nets, Lemma 4.4.1 will give an alternative characterisation of saturation, by induction on the construction of a net.

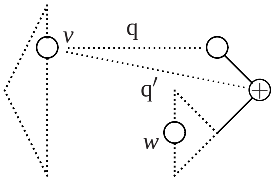
Because the statement of the lemma is relatively complex, it will first be motivated informally. In the illustration below, the net on the left depicts a copointed subnet q, between a node *v* and the right root, in the saturation of a net f. The source object of f is drawn as a dotted triangle, with the node *v* made explicit.

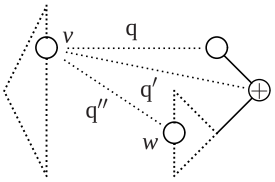




$\mathrm{q} \subseteq (\sigma\mathrm{f})_{v,\varepsilon}$ $\mathrm{q} \subseteq (\sigma\mathrm{f};\iota_0)_{v,0} \subseteq (\sigma(\mathrm{f};\iota_0))_{v,0}$

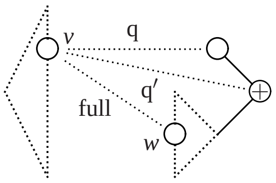
Above on the right, the pre-net $\sigma\mathrm{f};\iota_0$, which is a sub-pre-net of the saturation of $\mathrm{f};\iota_0$, has the same subnet q between vertices *v* and 0. (The vertex *w* is an arbitrary one in the lower branch of the target of $\mathrm{f};\iota_0$.) Because q is copointed, the saturation of $\mathrm{f};\iota_0$ adds the copointed subnet q′ below left, a displaced duplicate of q. This can be viewed as happening through a synchronised saturation step, much like the one illustrated in Figure 4.2.

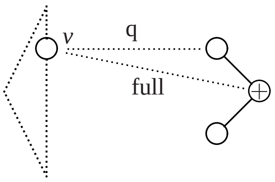




$\mathrm{q}' \subseteq (\sigma(\mathrm{f};\iota_0))_{v,\varepsilon}$ $\mathrm{q}'' \subseteq (\sigma(\mathrm{f};\iota_0))_{v,w}$

Next, in the saturation of $\mathrm{f};\iota_0$ the copointed subnet q′ is duplicated to any vertex in the target tree, as described by Lemma 4.3.2; for a given *w*, the subnet q″ between *v* and *w* is highlighted in the picture above right. If *w* is pointed the subnet q″ is bipointed, and its saturation is full, illustrated below left. Note that if the target of $\mathrm{f};\iota_0$ is itself pointed, the sub-pre-net between *v* and ε will be full in the saturation (below right). Also, if a vertex 0*u* in the upper branch of the target of $\mathrm{f};\iota_0$ is pointed, then σf must already be full between *v* and *u*.





$(\sigma(\mathrm{f};\iota_0))_{v,w}$ is full $(\sigma(\mathrm{f};\iota_0))_{v,\varepsilon}$ is full

To summarise the above, the saturation of a net $\mathrm{f};\iota_0$ contains three, possibly overlapping, collections of links, described in terms of the saturation of f:

- the saturation of f itself, in the context of an injection: $(\sigma\mathrm{f};\iota_0)$—containing, among others, the links in q above;
- any possible link $\langle v,*,w\rangle$, if v has a rooted initial link $\langle v,*,\varepsilon\rangle$ in the saturation of f—the links in q' and q'' above;
- any possible link $\langle v',*,w'\rangle$ that is between some nodes v and w (i.e. $v \leq v'$ and $w \leq w'$) such that w is pointed, and $\sigma\mathrm{f}$ contains a copointed subnet q between v and ε—the links in the full subprenets above.

In formalising this, the following definitions will be convenient. In a pre-net $\mathrm{f} = (X,Y,\mathcal{R})$, say that a vertex v in X has a *rooted copointed subnet* if there is a copointed net $\mathrm{q} \subseteq \mathrm{f}_{v,\varepsilon}$. If v is minimal among the vertices in X that have rooted copointed subnets in f, then v is said to have a *maximal copointed subnet*; let $\textsc{maxcp}(\mathrm{f})$ denote the set of such vertices in f. Note that if v becomes smaller, $\mathrm{f}_{v,\varepsilon}$ becomes larger; hence the *minimal* v gives the *maximal* copointed subnet. Dually, let $\textsc{maxp}(\mathrm{f})$ be the set of vertices in Y that have *maximal pointed subnets*, i.e. are minimal among the vertices that have *rooted pointed subnets*.

Lemma 4.4.1. *For a net* $\mathrm{g};\iota_j$ *the following holds.*

a. Let $\sigma\mathrm{g} = (X,Y_j,\mathcal{R})$ *and let* $\sigma(\mathrm{g};\iota_j) = (X,Y,\mathcal{S})$*. Then* $\mathcal{S} = (\mathcal{R}\cdot j) \cup \Gamma \cup \Delta$*, where*

$$\begin{aligned}
\Gamma &= \{\langle v,*,w\rangle \mid X_v = \mathbf{0},\ \langle v,*,\varepsilon\rangle \in \mathcal{R}\}\\
\Delta &= \{\langle v,*,w\rangle \mid X_v = \mathbf{0} \text{ or } Y_w = \mathbf{1},\\
&\qquad\qquad \exists v' \leq v.\ v' \in \textsc{maxcp}(\sigma\mathrm{g}),\\
&\qquad\qquad \exists w' \leq w.\ Y_{w'} \text{ is pointed}\}
\end{aligned}$$

Dually, for a net $\pi_i;\mathrm{g}$ *the following holds.*

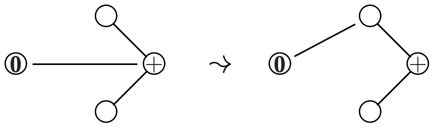
b. Let $\sigma\mathrm{g} = (X_i,Y,\mathcal{R})$ *and let* $\sigma(\pi_i;\mathrm{g}) = (X,Y,\mathcal{S})$*. Then* $\mathcal{S} = (i\cdot\mathcal{R}) \cup \Gamma \cup \Delta$*, where*

$$
\begin{aligned}
\Gamma &= \{\langle v,*,w\rangle \mid Y_w = \mathbf{1},\ \langle \varepsilon,*,w\rangle \in \mathcal{R}\,\} \\
\Delta &= \{\langle v,*,w\rangle \mid X_v = \mathbf{0} \text{ or } Y_w = \mathbf{1}, \\
&\qquad\qquad\qquad \exists v' \leq v.\ X_{v'} \text{ is copointed}, \\
&\qquad\qquad\qquad \exists w' \leq w.\ w' \in \textsc{maxp}(\sigma\mathrm{g})\ \}
\end{aligned}
$$

Proof. Case a. will be treated; b. is dual. Without loss of generality let $j = 0$. One direction, that $(\mathcal{R}\cdot 0) \cup \Gamma \cup \Delta \subseteq \mathcal{S}$, is as follows. That $(\mathcal{R}\cdot 0) \subseteq \mathcal{S}$, or equivalently that $\sigma\mathrm{g};\iota_0 \subseteq \sigma(\mathrm{g};\iota_0)$, follows from the fact that every saturation step $\sim\!\!(\mathrm{h}\,|\,\mathrm{k})\!\!\rightarrowtail_{v,w}$ applied to g has a corresponding step in $\sim\!\!(\mathrm{h}\,|\,\mathrm{k})\!\!\rightarrowtail_{v,0w}$ in $\mathrm{g};\iota_0$. That $\Gamma \subseteq \mathcal{S}$ follows by Lemma 4.2.3, which states that the saturation of an initial net is full: if $\langle v,\varepsilon\rangle$ is an initial link in $\sigma\mathrm{g}$, then this link forms an initial subnet $(!;\iota_0) \subseteq (\sigma(\mathrm{g};\iota_0))_{v,\varepsilon}$; filling this subnet means $\mathcal{S}$ contains all initial links $\langle v,w\rangle$ for any w in Y. For $\Delta \subseteq \mathcal{S}$, if $\mathrm{q} \subseteq (\sigma\mathrm{g})_{v,\varepsilon}$ is a maximal copointed subnet then by a synchronised saturation step there is a copointed subnet $\mathrm{q}' \subseteq (\sigma(\mathrm{g};\iota_0))_{v,\varepsilon}$. Then by Lemma 4.3.2, for any copointed w in Y there is a copointed subnet $\mathrm{q}'' \subseteq (\sigma(\mathrm{g};\iota_0))_{v,\varepsilon}$; this is a bipointed net, which by Lemma 4.3.7 has a saturation that is full; then $\mathcal{S}$ contains all possible unit links of the form $\langle vv',ww'\rangle$.

For the other direction, it will be shown that $(\mathcal{R}\cdot 0) \cup \Gamma \cup \Delta$ is closed under saturation ($\rightarrowtail$). Since it contains the links in $\mathrm{g};\iota_0$, this is sufficient to show that it contains $\mathcal{S}$. There are eight cases to consider, one for each saturation step.

- $\sim\!\!(?\,|\,?;\iota_i)\!\!\rightarrowtail_{v,w}$

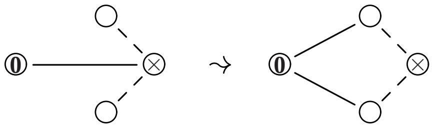


It must be shown that if $\langle v,*,w\rangle$ is in $(\mathcal{R}\cdot 0) \cup \Gamma \cup \Delta$ then so is $\langle v,*,wi\rangle$. The assumption gives three cases. For the first, if $\langle v,w\rangle \in \mathcal{R}\cdot 0$ then $w = 0w'$ for some w' and, since

$$\sigma\mathrm{f} \ \sim\!\!(?\,|\,?;\iota_i)\!\!\rightarrowtail_{v,w'}\ \sigma\mathrm{f}\,,$$

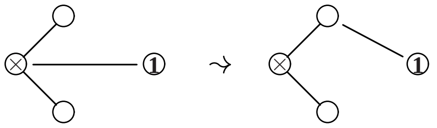
$\langle v,w'i\rangle \in \mathcal{R}$, so that $\langle v,wi\rangle \in \mathcal{R}\cdot 0$. In the second case, $\langle v,w\rangle \in \Gamma$. Since Γ fills the subnet between v and the root of Y, also $\langle v,wi\rangle \in \Gamma$. The third case is $\langle v,w\rangle \in \Delta$. For the first constraint set by Δ, because of the applied rewrite rule v must be $\mathbf{0}$. The second constraint, that some $v' \leq v$ has a maximal copointed subnet, holds for $\langle v,wi\rangle$ as it does for $\langle v,w\rangle$. For the third, if $w' \leq w$ then also $w' \leq wi$. It follows that $\langle v,wi\rangle \in \Delta$.

- $\sim\!(?\,|\,\langle ?,?\rangle)\!\rightarrowtail_{v,w}$



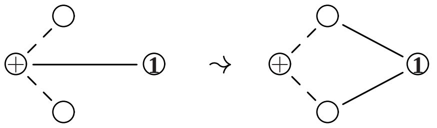
To be shown is that if $\langle v,w\rangle$ is in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$, both $\langle v,w0\rangle$ and $\langle v,w1\rangle$ are as well. The proof is similar to the above: if $\langle v,w\rangle$ is in $\mathcal{R}\cdot 0$, resp. Γ, resp. Δ, so are $\langle v,w0\rangle$ and $\langle v,w1\rangle$.

- $\sim\!(!\,|\,\pi_i;!)\!\rightarrowtail_{v,w}$



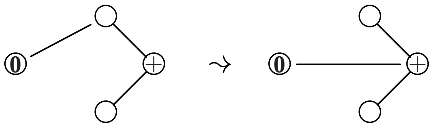
To be shown is that if $\langle v,w\rangle$ is in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$, so is $\langle vi,w\rangle$. The proof is mostly similar to the first case above: if $\langle v,w\rangle$ is in $\mathcal{R}\cdot 0$ resp. Δ, so is $\langle vi,w\rangle$, and $\langle v,w\rangle$ is not in Γ since v is a product, not $\mathbf{0}$.

- $\sim\!(!\,|\,[!,!])\!\rightarrowtail_{v,w}$



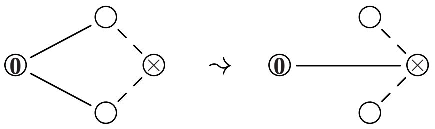
To be shown is that if $\langle v,w\rangle$ is in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$ then so are $\langle v0,w\rangle$ and $\langle v1,w\rangle$. The proof is as above: if $\langle v,w\rangle$ is in $\mathcal{R}\cdot 0$ resp. Δ, so are $\langle v0,w\rangle$ and $\langle v1,w\rangle$, and $\langle v,w\rangle$ is not in Γ.

- $\sim\!(?;\iota_i\,|\,?)\!\rightarrowtail_{v,w}$



To be shown is that if $\langle v,wi\rangle$ is in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$ then so is $\langle v,w\rangle$. If $\langle v,wi\rangle\in\mathcal{R}\cdot 0$ either $0\leq w$, or $w=\varepsilon$ and $i=0$. In the former case also $\langle v,w\rangle\in\mathcal{R}\cdot 0$. In the latter case $\langle v,wi\rangle$ is $\langle v,0\rangle\in\mathcal{R}\cdot 0$; then $\langle v,\varepsilon\rangle$ is a link in $\mathcal{R}$ and, by the definition of the rewrite rule, is initial. It follows that $\langle v,w\rangle=\langle v,\varepsilon\rangle$ is in Γ. Next, if $\langle v,wi\rangle\in\Gamma$ then also $\langle v,w\rangle\in\Gamma$. Finally, if $\langle v,wi\rangle\in\Delta$, then some $v'\leq v$ has a maximal copointed subnet, while v is $\mathbf{0}$ by the definition of the rewrite rule. For the remaining condition, that $\langle v,wi\rangle$ is in Δ means some $w'\leq wi$ is pointed. There are two cases: $w'\leq w$ or $w'=wi$, for which it must be shown that some $w''\leq w$ is pointed. In the former this is immediate. In the latter, since Y_{wi} is pointed and Y_w is $Y_{w0}+Y_{w1}$ (by the applied rewrite rule), w must be pointed. It follows that $\langle v,w\rangle\in\Delta$.

- $\leftarrowtail(\langle ?,?\rangle \mid ?)\rightarrowtail_{v,w}$



To be shown is that if both $\langle v,w0\rangle$ and $\langle v,w1\rangle$ are in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$ then so is $\langle v,w\rangle$. If both $\langle v,wi\rangle$ are in $\mathcal{R}\cdot 0$ then also $\langle v,w\rangle$ is in $\mathcal{R}\cdot 0$. If either $\langle v,wi\rangle$ is in Γ then immediately $\langle v,w\rangle\in\Gamma$. If both $\langle v,wi\rangle$ are in Δ then so is $\langle v,w\rangle$: by the rewrite rule v is $\mathbf{0}$; some $v'\leq v$ has a maximal copointed subnet; either some $w'\leq w$ is pointed or both $w0$ and $w1$ are, in which case w is pointed because Y_w is $Y_{w0}\times Y_{w1}$. This leaves the case where one link, say $\langle v,w1\rangle$, is in $\mathcal{R}\cdot 0$ and the other, $\langle v,w0\rangle$, in Δ. It will be shown that also in this case both links are in Δ, or both are in $\mathcal{R}\cdot 0$.

Firstly, w cannot be the root of Y, since the former is a product and the latter a coproduct. Then $0\leq w$, because $\langle v,w1\rangle$ is in $\mathcal{R}\cdot 0$. For convenience, let $w=0u$, so that u and w are corresponding vertices in f and f;ι_0 respectively. Because $\langle v,w0\rangle\in\Delta$ some $w'\leq w0$ is pointed. If also $w'\leq w$, then $\langle v,w1\rangle$ must be in Δ, a case already covered. So w' must be $w0$. That $\langle v,w0\rangle$ is in Δ also means that some $v'\leq v$ has a maximal copointed subnet in σf. Let this subnet be

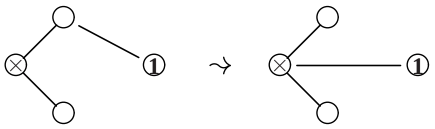
$$(X_{v'},Y_0,\mathcal{Q})\ \subseteq\ (\sigma\mathrm{f})_{v',\varepsilon}\ .$$

Then by Lemma 4.3.2 there is also the sub-pre-net

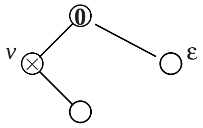
$$(X_{v'},Y_0,\mathcal{Q}\cdot u0)\ \subseteq\ (\sigma\mathrm{f})_{v',\varepsilon}$$

which forms a copointed subnet between v' and $u0$ in σf (note that $u0$ is the position in f corresponding to $w0$ in f;ι_0). As $u0$ is pointed, this subnet is bipointed, and by Lemma 4.3.7 must be full in the saturation of f. This means that $\langle v,u0\rangle$ is in $\mathcal{R}$, or in other words that $\langle v,w0\rangle$, as well as $\langle v,w1\rangle$, is in $\mathcal{R}\cdot 0$, a case already covered.

- $\leftarrowtail(\pi_i;!\mid !)\rightarrowtail_{v,w}$

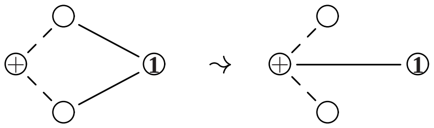


To be shown is that if $\langle vi,w\rangle$ is in $(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$, so is $\langle v,w\rangle$. Firstly, if $\langle vi,w\rangle$ is in $\mathcal{R}\cdot 0$ then so is $\langle v,w\rangle$. Secondly, if $\langle vi,w\rangle\in\Gamma$ then σf contains an initial link $\langle vi,\varepsilon\rangle$. Because of the applied rewrite rule v is the product $X_{v0}\times X_{v1}$. Then the link $\langle vi, \varepsilon \rangle$ forms a copointed subnet between v and ε in σf, illustrated below (for $i = 0$).

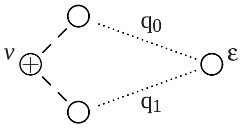


Then there is a $v' \leq v$ with a maximal copointed subnet in σf; moreover, the rewrite rule forces w to be **1**, and hence pointed, which means that $\langle v, w \rangle$ is in Δ. Thirdly, if $\langle vi, w \rangle \in \Delta$ then some $v' \leq vi$ has a maximal copointed subnet in σf. Either $v' = vi$ or $v' \leq v$. The former case is ruled out because a copointed subnet for vi can never be maximal: if q is a copointed subnet in σf between vi and ε, then π_i;q is a copointed subnet between v and ε. In the latter case it is immediate that also $\langle v, w \rangle \in \Delta$.

- $\sim\!([!,!] \mid !)\!\rightsquigarrow_{v,w}$



To be shown is that if both $\langle v0, w \rangle$ and $\langle v0, w \rangle$ are in $(\mathcal{R} \cdot 0) \cup \Gamma \cup \Delta$, then so is $\langle v, w \rangle$. Firstly, if both $\langle vi, w \rangle$ are in $\mathcal{R} \cdot 0$ then so is $\langle v, w \rangle$. Secondly, suppose one $\langle vi, w \rangle$ is in Γ. Then there is an initial link $\langle vi, \varepsilon \rangle$ in the saturation of f, which forms a copointed subnet between vi and ε; then some $v' \leq vi$ has a maximal copointed subnet. Since by the rewrite rule w is **1**, and thus pointed, $\langle vi, w \rangle$ is in Δ; this case is then reduced to the following ones. Thirdly, suppose both $\langle vi, w \rangle$ are in Δ. If some $v' \leq v$ has a maximal copointed subnet, also $\langle v, w \rangle$ is in Δ. The other case is ruled out: if both $v0$ and $v1$ have maximal copointed subnets q_0 resp. q_1, then $[\mathrm{q}_0, \mathrm{q}_1]$ would form a larger copointed subnet (illustrated below).



The final case is where one link, say $\langle v0, w \rangle$, is in $\mathcal{R} \cdot 0$, and the other, $\langle v1, w \rangle$, in Δ. It will be shown that also $\langle v1, w \rangle$ must be in $\mathcal{R} \cdot 0$, reducing this case to a previous one. From $\langle v0, w \rangle \in \mathcal{R} \cdot 0$ it follows that $w = 0u$ for some u, and $\langle v1, w \rangle \in \Delta$ means that some $v' \leq v1$ has a maximal copointed subnet in σf. Let this subnet be

$$(X_{v'}, Y_0, \mathcal{Q}) \subseteq (\sigma \mathrm{f})_{v',\varepsilon} \,.$$

Then by Lemma 4.3.2 there is also the sub-pre-net

$$(X_{v'}, Y_0, \mathcal{Q} \cdot u) \;\subseteq\; (\sigma\mathrm{f})_{v',\varepsilon} \,,$$

which forms a copointed subnet in f between v' and u. This subnet is then bi-pointed, since the applied rewrite rule means u is **1**, and hence pointed. By Lemma 4.3.7 then σf is full between v' and u, and in particular $\langle v1, u\rangle \in \mathcal{R}$, and $\langle v1, w\rangle \in \mathcal{R} \cdot 0$.

□

4.5 Deconstruction of saturated nets

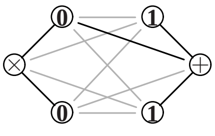
The two remaining obstacles for the present case in the soundness proof, of parallel nets between a product and a coproduct, are:

I nets constructed over different projections or injections may have the same saturation, and

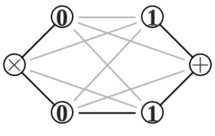
II the induction hypothesis may not apply even when nets are constructed in the same way.

The main lemma of this section, Lemma 4.5.1, will solve the first, and make a start on the second.

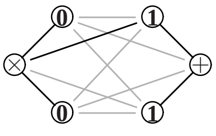
An illustration of the first problem, below, shows three nets, constructed over different projections and injections, with the same, full saturation, indicated by the grey links.



$\pi_0;?$



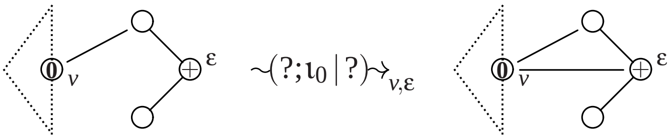
$\pi_1;!;\iota_1$



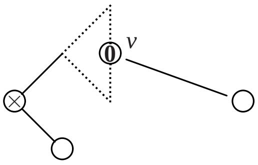
$!;\iota_0$

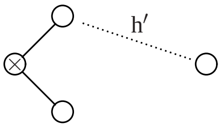
In general, for nets that are constructed differently, e.g. $\mathrm{f};\iota_0$ and $\pi_1;\mathrm{g}$, or $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_1$, there is no hope of applying the induction hypothesis of the soundness proof to f and g, which need not even be parallel.

A direction in which to look for a solution is suggested by the dynamics of saturating a net $\mathrm{f};\iota_0$, as explored in the previous section. After first saturating f, the next step in saturating $\sigma\mathrm{f};\iota_0$ must be to move an initial link $\langle v, 0\rangle$ up to the root, adding $\langle v, \varepsilon\rangle$—all other steps stay within f, and have already been performed.



Then consider a corresponding equivalence step $\mathrm{g} \mathrel{=\!\![} ?;\iota_0 \,|\, ? \mathrel{]\!\!\Rightarrow}_{v,\varepsilon} \mathrm{h}$ between two nets equivalent to $\mathrm{f};\iota_0$, with g containing the initial link $\langle v, 0\rangle$ and h containing $\langle v,\varepsilon\rangle$. Because $\langle v,\varepsilon\rangle$ connects to the right root, the net h cannot be right-constructible. In the case of nets between products and coproducts, it must then be left-constructible, of the form $\pi_i;\mathrm{h}'$. Moreover, as illustrated below, the projection over which this net is constructed is determined by which branch of the source product v resides in: if $0 \leq v$ then it is $\pi_0;\mathrm{h}'$, and if $1 \leq v$ then $\pi_1;\mathrm{h}'$.





To summarise, the presence of a rooted initial or terminal link in a net from a product into a coproduct determines over which projection or injection it is constructed. What the soundness proof needs to show is that any rooted link in σf must occur in some net $\mathrm{f}' \Leftrightarrow \mathrm{f}$. It will then be immediate that two nets with the same saturation, containing the same rooted link, must be constructed similarly.

In fact, Lemma 4.5.1 below proves the following generalisation: any pointed or copointed partial subnet of σf occurs as a partial subnet of some net $\mathrm{f}' \Leftrightarrow \mathrm{f}$. Recalling that a partial net is a pre-net satisfying compatibility, but not necessarily connectedness, another way of phrasing the statement of Lemma 4.5.1 is that any collection of rooted initial links in σf that, by the switching conditions, may occur together in the same net at all, will actually occur in some $\mathrm{f}' \Leftrightarrow \mathrm{f}$; and similarly for any such collection of rooted terminal links.

This generalisation is prompted by two considerations. One is the need for a suitable induction hypothesis in the proof itself. The other is found in an analysis of problem II indicated above, of similarly constructed nets to which the induction hypothesis of the soundness proof nonetheless does not apply. In the illustration in Figure 4.5, in isolation the upper two nets are not equivalent, but after placing them in the context of an injection into $\mathbf{0} \times \mathbf{1}$, forming the lower two nets, they become equivalent. (In the illustration, saturations are indicated by the grey links; no saturation steps apply to the upper two nets.)

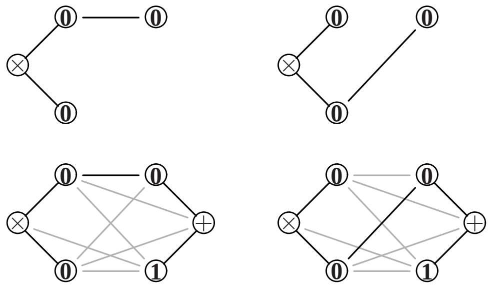
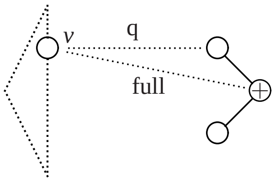
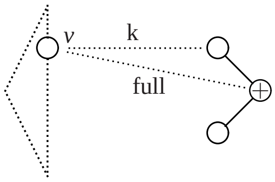


Figure 4.5: Nets may become equivalent by composing with an injection

More generally, suppose $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ are saturated nets that are similar, except that $\sigma\mathrm{f}$ has a copointed subnet q between some vertex v and the right root ε, while $\sigma\mathrm{g}$ has a different copointed subnet k. Then after placing f and g in the context of an injection into a pointed object, the resulting nets $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_0$ will have the same saturation, as schematically illustrated below: the subnets q and k are obscured in the saturation, as the latter is full between v and ε.





The solution, discussed in detail in Section 4.6, will be to show the equivalence of both copointed subnets in the context of the injection, $\mathrm{q};\iota_0$ and $\mathrm{k};\iota_0$. However, q and k are subnets of the saturated nets $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$, but not of f and g themselves. Addressing this is the second reason why Lemma 4.5.1 is stated the way it is: in this particular case, it concludes that q is a subnet of some $\mathrm{f}' \Leftrightarrow \mathrm{f}$, and k of some $\mathrm{g}' \Leftrightarrow \mathrm{g}$.

Lemma 4.5.1. *If* f *is a net and* $\mathrm{q} \subseteq \sigma\mathrm{f}$ *is a partial pointed or copointed net, then there is a net* g *s.t.* $\mathrm{q} \subseteq \mathrm{g}$ *and* $\mathrm{f} \Leftrightarrow \mathrm{g}$.

Proof. The proof is by induction on the construction of f. The case where q is copointed is treated explicitly, while the case for pointed q follows by duality. The two cases should be considered as simultaneous, as both forms of the induction hypothesis are needed for the present case.

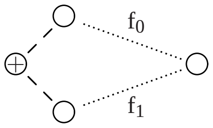
Recall that a partial copointed net q is either the basic net ?, is empty, or is constructed as $[\mathrm{q}_0, \mathrm{q}_1]$ or as $\pi_i;\mathrm{q}_i$. If q is empty then g can be chosen as $\mathrm{g} = \mathrm{f}$; for this reason q is generally assumed non-empty below. The construction of f has seven cases.

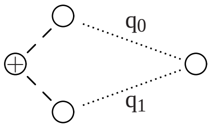
- If $\mathrm{f} = (A, B, a)$ then q must be empty.

- If $\mathrm{f} = (\mathbf{0}, Y, *)$ then q must be $(\mathbf{0}, Y, *)$ (it is not empty). Let $\mathrm{g} = \mathrm{f}$.

- If $\mathrm{f} = (X, \mathbf{1}, *)$ it is pointed, while q is a (parallel) partial copointed net. For f and q Lemma 4.3.4 gives the required net g with $\mathrm{f} \Leftrightarrow \mathrm{g}$ and $\mathrm{q} \subseteq \mathrm{g}$.

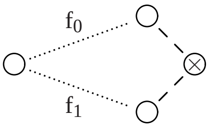
- If $\mathrm{f} = [\mathrm{f}_0, \mathrm{f}_1]$, then q is of the form $[\mathrm{q}_0, \mathrm{q}_1]$, as it must be left-constructible.

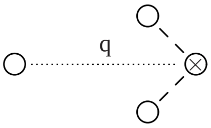


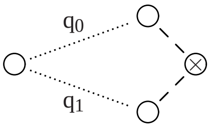


For $i \in \{0, 1\}$, by Lemma 4.2.6 the two saturated nets $\sigma\mathrm{f}_i$ are the subnets $(\sigma\mathrm{f})_{i,\varepsilon}$. Then since $\mathrm{q} \subseteq \sigma\mathrm{f}$ also $\mathrm{q}_i \subseteq \sigma\mathrm{f}_i$. The induction hypothesis then provides g_0 and g_1, from which g is constructed as $\mathrm{g} = [\mathrm{g}_0, \mathrm{g}_1]$.

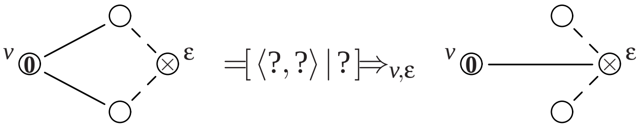
- If $\mathrm{f} = \langle \mathrm{f}_0, \mathrm{f}_1 \rangle$ and q is the partial net $(X, Y, \mathcal{Q})$, then let $\langle \mathrm{q}_0, \mathrm{q}_1 \rangle$ be the partial net $(X, Y, \mathcal{Q} \cdot 0 \ \cup \ \mathcal{Q} \cdot 1)$, obtained from q by moving it down from the root of the target object.





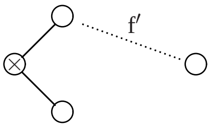


Each q_i is a sub-prenet of $\sigma\mathrm{f}_i$, since by Lemma 4.2.6 $\sigma\mathrm{f}_i = (\sigma\mathrm{f})_{\varepsilon,i}$. The induction hypothesis provides g_0 and g_1 such that $\mathrm{f}_i \Leftrightarrow \mathrm{g}_i$ and $\mathrm{q}_i \subseteq \mathrm{g}_i$. However, $\langle \mathrm{g}_0, \mathrm{g}_1 \rangle$ has $\langle \mathrm{q}_0, \mathrm{q}_1 \rangle$ as a sub-prenet, but not q itself. To obtain g from $\langle \mathrm{g}_0, \mathrm{g}_1 \rangle$ the links in q_0 and q_1 must be moved up to the root again, which is done by applying the following rewrite step to $\langle \mathrm{g}_0, \mathrm{g}_1 \rangle$, for every link $\langle v, \varepsilon \rangle$ in q.


v
ε
$=\!\![\,\langle ?,? \rangle \mid ?\,]\!\!\Rightarrow_{v,\varepsilon}$
v
ε

Then $\mathrm{q} \subseteq \mathrm{g}$ and $\mathrm{f} \ = \ \langle \mathrm{f}_0, \mathrm{f}_1 \rangle \ \Leftrightarrow \ \langle \mathrm{g}_0, \mathrm{g}_1 \rangle \ \Leftrightarrow \ \mathrm{g}$.

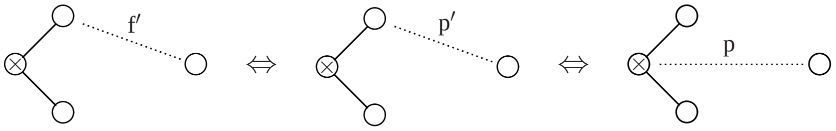
- If $\mathrm{f} = \pi_0;\mathrm{f}'$ (the case for $\pi_1;\mathrm{f}'$ is symmetric) and $\sigma\mathrm{f}' = (X_0, Y, \mathcal{R})$, then the linking of $\sigma\mathrm{f} = (X, Y, \mathcal{S})$ is described by Lemma 4.4.1b as the collection $\mathcal{S} = (0 \cdot \mathcal{R}) \cup \Gamma \cup \Delta$.



Three cases will be distinguished: one, some link in q is in Γ; two, some link in q is in Δ; and three, all links in q are in $\sigma\mathrm{f}'$.

For the first case, recall that Γ is a collection of terminal links, with target $\mathbf{1}$. If it contains a link from q, which are all of the form $\langle v,\varepsilon\rangle$, then Y must be $\mathbf{1}$. Then f is a terminal net, and by Lemma 4.2.1 equivalent to the basic net $(X,\mathbf{1},*)$. For this net, which is pointed, and the partial copointed net q, Lemma 4.3.4 gives the required net g, for which $\mathrm{q}\subseteq\mathrm{g}$ and $\mathrm{g}\Leftrightarrow(X,\mathbf{1},*)\Leftrightarrow\mathrm{f}$

For the second case, if some $\langle v,\varepsilon\rangle$ is in Δ, by the definition of Δ the right root ε must have a (maximal) pointed subnet $\mathrm{p}'\subseteq\sigma\mathrm{f}'$. Applying the induction hypothesis—in its dual form to the one being discussed—to f' and the pointed net p' gives a net g' with $\mathrm{f}'\Leftrightarrow\mathrm{g}'$ and $\mathrm{p}'\subseteq\mathrm{g}$, and since p' is a net, g' must be p' itself. Then also $\mathrm{f}=\pi_0;\mathrm{f}'$ and $\pi_0;\mathrm{p}'$ are equivalent, while the latter has an equivalent pointed net p, by moving it up to the left root.



For the pointed net p and the partial copointed net q Lemma 4.3.4 gives the net g, with $\mathrm{q}\subseteq\mathrm{g}$, completing the equivalence below.

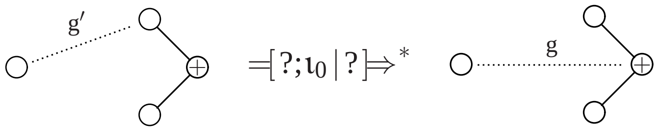
$$\mathrm{f}\;=\;\pi_0;\mathrm{f}'\;\Leftrightarrow\;\pi_0;\mathrm{p}'\;\Leftrightarrow\;\mathrm{p}\;\Leftrightarrow\;\mathrm{g}$$

In the remaining case, $\mathrm{q}\subseteq\pi_0;\mathrm{f}'$, which means that q must be of the form $\pi_0;\mathrm{q}'$. The induction hypothesis for f' and q' gives a net g' such that $\mathrm{q}'\subseteq\mathrm{g}'$ and $\mathrm{f}'\Leftrightarrow\mathrm{g}'$. These two properties carry over to $\pi_0;\mathrm{g}'$, which is the required net g, as per the following.

$$\mathrm{q}\;=\;\pi_0;\mathrm{q}'\;\subseteq\;\pi_0;\mathrm{g}'\;=\;\mathrm{g}\qquad\qquad\mathrm{f}\;=\;\pi_0;\mathrm{f}'\;\Leftrightarrow\;\pi_0;\mathrm{g}'\;=\;\mathrm{g}$$

- If $\mathrm{f}=\mathrm{f}';\iota_0$ (the case for $\mathrm{f}';\iota_1$ is symmetric) and $\sigma\mathrm{f}'=(X,Y_0,\mathcal{R})$, then the linking in $\sigma\mathrm{f}=(X,Y,\mathcal{S})$ is described by Lemma 4.4.1a as the collection $\mathcal{S}=(\mathcal{R}\cdot 0)\cup\Gamma\cup\Delta$. Let $\mathrm{q}\subseteq\sigma\mathrm{f}$ be the partial copointed net $(X,Y,\mathcal{Q})$. Firstly, since the links in q are all of the form $\langle v,\varepsilon\rangle$, none can be in $\mathcal{R}\cdot 0$. Two further cases will be distinguished: one, all links in q are in Γ; and two, some link in q is in Δ.

  For the first, if $\mathcal{Q}\subseteq\Gamma$ then, by the definition of Γ, for any link $\langle v,\varepsilon\rangle$ in $\mathcal{Q}$ there is a link $\langle v,0\rangle$ in $(\sigma\mathrm{f}');\iota_0$. These constitute a partial copointed subnet $\mathrm{q}'\subseteq\sigma\mathrm{f}'$, for which the induction hypothesis gives a net g' equivalent to f' and containing q'. Then $\mathrm{g}';\iota_0$ has $\mathrm{q}';\iota_0$ as a sub-pre-net, but not q itself; g is obtained from $\mathrm{g}';\iota_0$ by moving q' up to the root, as follows.



The remaining case is where Δ contains at least one link $\langle v,\varepsilon\rangle$ in q. By the definition of Δ, the target object Y of f must be pointed. Then $\Gamma \subseteq \Delta$, and since $\mathcal{Q}$ does not share any links with $\mathcal{R}\cdot 0$, also $\mathcal{Q} \subseteq \Delta$.

To apply the induction hypothesis, a partial copointed net $\mathrm{q}' \subseteq \sigma\mathrm{f}'$ will be built from a selection of the maximal copointed subnets in $\sigma\mathrm{q}'$. Let V be the following collection of vertices in X with maximal copointed subnets in $\sigma\mathrm{f}'$.

$$V \;=\; \{v \in \textsc{maxcp}(\sigma\mathrm{f}') \mid \exists u \geq v.\ \langle u,*,\varepsilon\rangle \in \mathcal{Q}\}$$

Note that for every link $\langle u,\varepsilon\rangle$ in q there is a $v \leq u$ in V, because $\langle u,\varepsilon\rangle$ is in Δ. Next, For each $v \in V$ choose a maximal copointed subnet k_v of $\sigma\mathrm{f}'$.

$$\mathrm{k}_v \;=\; (X_v, Y_0, \mathcal{K}_v) \;\subseteq\; (\sigma\mathrm{f}')_{v,\varepsilon}$$

Construct q' as the combination of all k_v, as follows, so that $\mathrm{q}'_{v,\varepsilon} = \mathrm{k}_v$.

$$\mathrm{q}' \;=\; (X, Y_0, \mathcal{Q}') \qquad\qquad \mathcal{Q}' \;=\; \bigcup_{v\in V} (v\cdot\mathcal{K}_v)$$

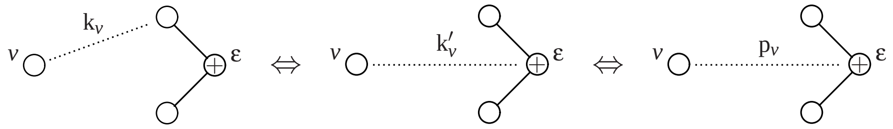
By construction q' is a copointed sub-pre-net of $\sigma\mathrm{f}'$. For it to be a partial net, any two links in q' must be compatible. For links within a single net k_v this is immediate. For links in different k_v and $\mathrm{k}_{v'}$ it is sufficient to show that v and v' are compatible. Firstly, neither $v \leq v'$ nor $v' \leq v$, by maximality of k_v and k'_v. Secondly, by the definition of V, there are links $\langle u,\varepsilon\rangle$ and $\langle u',\varepsilon\rangle$ in q, with $v \leq u$ and $v' \leq u'$. Since the least common ancestor of v and v' is the same as that of u and u', from $v \# v'$ it would follow that $u \# u'$. But then the links $\langle u,\varepsilon\rangle$ and $\langle u',\varepsilon\rangle$ in q would be incompatible, a contradiction since q is a partial net.

The induction hypothesis applied to q' and f' gives a net g' equivalent to f' and containing q'. In particular, for each $v \in V$,

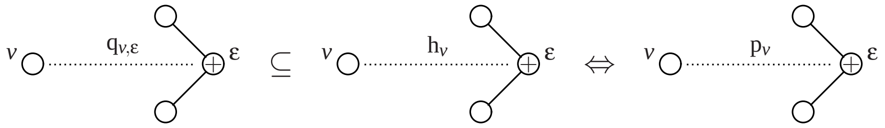
$$\mathrm{g}'_{v,\varepsilon} \;=\; \mathrm{q}'_{v,\varepsilon} \;=\; \mathrm{k}_v\ .$$

The net g will be obtained from $\mathrm{g}';\iota_0$ by replacing, for every v in V, the subnet $\mathrm{k}_v;\iota_0$ by an equivalent subnet h_v containing $\mathrm{q}_{v,\varepsilon}$. For a given $v \in V$, firstly, $\mathrm{k}_v;\iota_0$

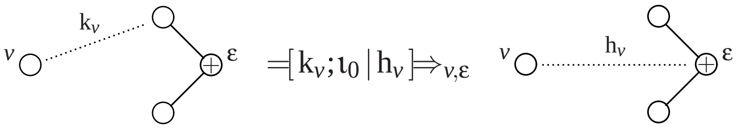
is equivalent to a copointed net k'_v by moving it up to the root. Recall that Y, the target of k'_v, is pointed (because the links in q are in Δ). Then there is a pointed net p_v to which k'_v is equivalent, by Lemma 4.3.5, since both are bipointed.



Since $q_{v,\varepsilon}$ is a partial copointed net parallel to p_v Lemma 4.3.4 applies, and gives a net h_v equivalent to p_v that has $q_{v,\varepsilon}$ as a sub-pre-net.



Then g is obtained from $g';\iota_0$ by applying the following rewrite for each $v \in V$.


v
k_v
ε
$=\!\![k_v;\iota_0 \,|\, h_v]\!\!\Rightarrow_{v,\varepsilon}$
v
h_v
ε

Because the vertices in V do not dominate one another, the domains of the different rewrites are disjoint, so that none invalidates the precondition for another (that the subnet to be replaced, between source vertex v and target vertex 0, must be k_v). It follows that $g \Leftrightarrow g';\iota_0$, since for each $v \in V$ the nets $k_v;\iota_0$ and h_v are equivalent. Recall that $g';\iota_0 \Leftrightarrow f';\iota_0 = f$ by the induction hypothesis, giving $g \Leftrightarrow f$. Finally, because any link in q is in some h_v, and $g_{v,\varepsilon} = h_v$ for all $v \in V$, it follows that $q \subseteq g$.

□

The argument at the start of this section, showing how Lemma 4.5.1 solves the problem of equivalent nets that are constructed over different projections and injections, gives the following lemma. In the statement of the lemma, recall that non-constructible nets are those that are neither left-constructible nor right-constructible.

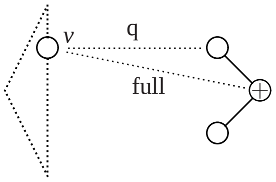
Lemma 4.5.2. *Let* f *and* g *be parallel nets between a product X and a coproduct Y, with the same saturation* $\sigma f = \sigma g = (X, Y, \mathcal{R})$. *If this saturation is non-constructible then there are nets* $\pi_i;f' \Leftrightarrow f$ *and* $\pi_i;g' \Leftrightarrow g$ *constructed with the same projection* π_i, *and nets* $f'';\iota_j \Leftrightarrow f$ *and* $g'';\iota_j \Leftrightarrow g$ *constructed with the same injection* ι_j.

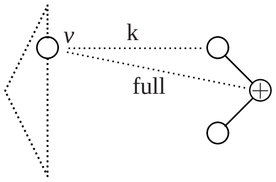
Proof. Without loss of generality let f be of the form $f_0;\iota_0$. Let the linking of σf be described by $\mathcal{R} \cdot 0 \cup \Gamma \cup \Delta$ as in Lemma 4.4.1, where $\mathcal{R}$ is the linking of σf_0. Since f is non-constructible at least one of Γ and Δ must be non-empty; but if Δ is non-empty, σf_0 has a maximal copointed subnet, whose rooted initial links are in Γ. Thus Γ is non-empty, and contains at least one rooted initial link $\langle iv, \varepsilon \rangle$. This link forms a partial copointed net, and by Lemma 4.5.1 there is a net equivalent to f containing this link. This net cannot be right-constructible and so must be of the form $\pi_i;f'$. Since g has the same saturation as f, which contains $\langle iv, \varepsilon \rangle$, by the same argument there is a net $\pi_i;g'$. By duality, $\pi_i;f'$ and $\pi_i;g'$, having the same, non-constructible saturation as f, are equivalent to nets $f'';\iota_j$ and $g'';\iota_j$ respectively. ☐

4.6 Matching points

The present case of the soundness proof, of parallel nets from a product into a coproduct, is nearly complete. It was shown that if their (common) saturation is a constructible prenet, the induction hypothesis can be applied immediately, and that if it is not constructible, they are equivalent to nets constructed over the same injection or projection, say $f;\iota_0$ and $g;\iota_0$. A final obstacle, already highlighted in Section 4.5, where it inspired the formulation of Lemma 4.5.1, is the fact that their components f and g need not have the same saturation, and indeed need not be equivalent. The general mechanism by which this transpires is that bipointed nets have saturations that are full: if σf and σg contain copointed subnets q and k, these are no longer recognisable in the saturations of $f;\iota_0$ and $g;\iota_0$.

In a little more detail, the saturation of $f;\iota_0$ has the subnet $q;\iota_0$ between v and ε. By applying a synchronised saturation step (see Figure 4.2), $\sigma(f;\iota_0)$ contains the copointed subnet q' between v and ε, as well. Then if the target of $f;\iota_0$ is pointed, q' is bipointed, and its saturation must be full.





The above prompts two observations. Firstly, if the target of $f;\iota_0$ is not pointed, the final steps in this scenario do not pertain, and no links are added to the saturation of f. Secondly, if the target of f is pointed, the subnet q of f is already bipointed, and its saturation full; then saturating $q;\iota_0$ cannot add any more links. These two unproblematic cases are summarised by the following lemma.

Lemma 4.6.1. *Let* $\mathsf{f};\iota_j$ *be a net from* X *to* Y*. If* Y_j *is pointed or* Y *is not pointed, then*

$$\sigma\mathsf{f} \quad = \quad (\sigma(\mathsf{f};\iota_j))_{\varepsilon,j}\ .$$

Dually, let $\pi_i;\mathsf{g}$ *be a net from* X *to* Y*. If* X_i *is copointed or* X *is not copointed then*

$$\sigma\mathsf{g} \quad = \quad (\sigma(\pi_i;\mathsf{g}))_{i,\varepsilon}\ .$$

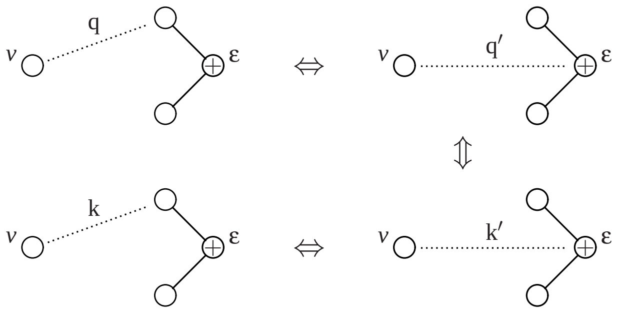
Proof. Consider the case for f; that for g is dual. Without loss of generality let $j = 0$ and, following Lemma 4.4.1a, let the saturations of f and $\mathsf{f};\iota_0$ be described as follows.

$$\sigma\mathsf{f} \ = \ (X, Y_0, \mathcal{R}) \qquad \sigma(\mathsf{f};\iota_0) \ = \ (X, Y, \mathcal{S}) \qquad \mathcal{S} \ = \ (\mathcal{R}\cdot 0) \cup \Gamma \cup \Delta$$

In one direction, it is then immediate that $(\sigma\mathsf{f};\iota_0) \subseteq \sigma(\mathsf{f};\iota_0)$. In the other direction, let $\langle v, 0w\rangle$ be a link in the saturation of $(\mathsf{f};\iota_0)$, i.e. in $\mathcal{S}$. It must be shown that $\langle v, w\rangle$ is in $\mathcal{R}$. The non-trivial cases are where $\langle v, 0w\rangle$ is in Γ or Δ. If $\langle v, 0w\rangle$ is in Γ then by the definition of Γ there is a link $\langle v, *, \varepsilon\rangle$ in $\mathcal{R}$. Then the initial subnet $(X_v, Y_0, *)$ of σf, between v and ε, is full, by Lemma 4.2.3, and $\langle v, w\rangle$ is in $\mathcal{R}$.

If $\langle v, 0w\rangle$ is in Δ then some $v' \leq v$ has a maximal copointed subnet in σf, and some $w' \leq w$ is pointed. If Y_0 is pointed then σf has a bipointed subnet between v' and ε, which is then full, containing in particular and $\langle v, w\rangle$. If Y is not pointed then $w' \neq \varepsilon$. Since $w' \leq w$ it must be that $w' = 0u$ for some u. By Lemma 4.3.2, the maximal copointed subnet at v' in σf has a corresponding copointed subnet between v' and u, which is then bipointed. Then the subnet of σf between v' and u is full, and contains $\langle v, w\rangle$. ☐

The solution for the last remaining instance is as follows. Suppose that nets $\mathsf{f};\iota_0$ and $\mathsf{g};\iota_0$ have the same saturation, while σf and σg have different copointed subnets q and k between some vertex v and ε. By moving them up to the root, $\mathsf{q};\iota_0$ and $\mathsf{k};\iota_0$ each have corresponding copointed nets q′ and k′. By the above lemma, the target of $\mathsf{q};\iota_0$ and $\mathsf{k};\iota_0$ must be pointed, making them bipointed, and thus equivalent (by Lemma 4.3.5), illustrated below.



The equivalence of $\mathrm{q};\iota_0$ and $\mathrm{k};\iota_0$ does not immediately show $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_0$ to be equivalent. Rather, the argument proceeds as follows. Firstly, since q is a subnet of $\sigma\mathrm{f}$, it is a subnet of a net equivalent to f (by Lemma 4.5.1); for simplicity, assume q is a subnet of f itself. Because $\mathrm{q};\iota_0$ is equivalent to $\mathrm{k};\iota_0$, after replacing q with k in f there is the following equivalence.

$$\mathrm{f};\iota_0 \quad \Leftrightarrow \quad (\mathrm{f}\{\mathrm{k}\}_{v,\varepsilon});\iota_0$$

The final step is then to show that $\mathrm{f}\{\mathrm{k}\}_{v,\varepsilon}$ has the same saturation as g, so that the induction hypothesis can be applied to show their equivalence.

In fleshing out this argument there are a few remaining obstacles. One is that $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$ may differ on several copointed subnets, and not just on q and k. If the copointed subnets of $\sigma\mathrm{f}$, taken together, form a partial net, then Lemma 4.5.1 can still be applied. However, that they do form a partial net is far from obvious, and will need proof.

Another issue is the following. A natural way of proving that, in the running example, $\mathrm{f}\{\mathrm{k}\}_{v,\varepsilon}$ and g have the same saturation, would be to show it by induction on their construction, using the fact that $(\mathrm{f}\{\mathrm{k}\}_{v,\varepsilon});\iota_0$ and $\mathrm{g};\iota_0$ have the same saturation. Unfortunately, this proof idea does not go through, because the latter property is not preserved in the induction steps. A weaker statement that does carry over in the induction, is the following: if the saturations of $\mathrm{f}\{\mathrm{k}\}_{v,\varepsilon}$ and $\sigma\mathrm{g}$ have the same maximal copointed subnet k at the same vertex v, they are identical between v and ε. To make this work, firstly, it will be immediate from Lemma 4.6.2 below that the same vertices have maximal copointed subnets in $\sigma\mathrm{f}$ and $\sigma\mathrm{g}$, given that $\mathrm{f};\iota_0$ and $\mathrm{g};\iota_0$ have the same saturation. After that, Lemma 4.6.3 will prove the statement above, generalised to allow for multiple copointed subnets. In the statement of the following lemma, recall that $\textsc{maxcp}(\mathrm{f})$ denotes the collection of vertices in the source of f that have maximal copointed subnets; and that dually $\textsc{maxp}(\mathrm{f})$ collects the vertices with maximal pointed subnets.

Lemma 4.6.2. *For a net* f *the following statements hold.*

$$\textsc{maxcp}(\sigma(\mathrm{f};\iota_j)) = \textsc{maxcp}(\sigma\mathrm{f}) \qquad\qquad \textsc{maxp}(\sigma(\pi_i;\mathrm{f})) = \textsc{maxp}(\sigma\mathrm{f})$$

Proof. Consider the case for $\mathrm{f};\iota_0$ and f; that for $\mathrm{f};\iota_1$ is symmetric, and that for $\pi_i;\mathrm{f}$ is dual. It must be shown that a vertex v, in the common source object of $\mathrm{f};\iota_0$ and f, has a maximal copointed subnet in $\sigma(\mathrm{f};\iota_0)$ if and only if it has one in $\sigma\mathrm{f}$. In both directions, it will be shown that if v has a maximal copointed subnet in one saturation, some $u \leq v$ has one in the other; the statement then follows by the minimality of v.

In one direction, if v has a maximal copointed subnet

$$\mathrm{q} \;=\; (X_v, Y_0, \mathcal{Q})$$

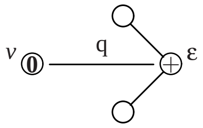
in $\sigma\mathrm{f}$, then in $\sigma(\mathrm{f};\iota_0)$ there is a corresponding copointed subnet q', obtained by moving q up to the root, from $\mathrm{q};\iota_0$ to the parallel q'. Then some $u \leq v$ has a maximal copointed subnet in $\sigma(\mathrm{f};\iota_0)$.

In the other direction, let v have a maximal copointed subnet q in $\sigma\mathrm{f}$:

$$\mathrm{q} \;\subseteq\; (\sigma(\mathrm{f};\iota_0))_{v,\varepsilon}\ .$$

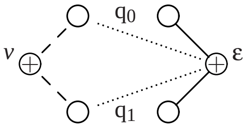
It will be shown by induction on q that some $u \leq v$ has a copointed subnet q' in $\sigma\mathrm{f}$.

- If $\mathrm{q} = (X_v, Y, *)$ it consists of the link $\langle v, \varepsilon \rangle$.

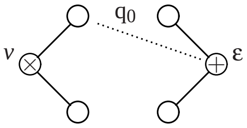
  Let $\sigma(\mathrm{f};\iota_0) = (X, Y, \mathcal{S})$ and let $\sigma\mathrm{f} = (X, Y_0, \mathcal{R})$, so that $\mathcal{S} = (\mathcal{R} \cdot 0) \cup \Gamma \cup \Delta$ in accordance with Lemma 4.4.1a. Since the link $\langle v, \varepsilon \rangle$ is in $\mathcal{S}$ there are three cases. Firstly, $\langle v, \varepsilon \rangle$ cannot be in $\mathcal{R} \cdot 0$. Secondly, if $\langle v, \varepsilon \rangle$ is in Γ, then the link $\langle v, 0 \rangle$ in $\mathcal{R} \cdot 0$ forms a copointed subnet in $\sigma\mathrm{f}$ between v and ε; let this be the required q'. Thirdly, if $\langle v, \varepsilon \rangle$ is in Δ then some $v' \leq v$ has a copointed subnet in $\sigma\mathrm{f}$; let q' be this subnet.

- If $\mathrm{q} = [\mathrm{q}_0, \mathrm{q}_1]$,

  then by the induction hypothesis some $v' \leq v0$ has a copointed subnet q'_0 in $\sigma\mathrm{f}'$ and some $v'' \leq v1$ has a copointed subnet q'_1 in $\sigma\mathrm{f}'$. If $v' \leq v$ let $\mathrm{q}' = \mathrm{q}'_0$, if $v'' \leq v$ let $\mathrm{q}' = \mathrm{q}'_1$, and otherwise let $\mathrm{q}' = [\mathrm{q}'_0, \mathrm{q}'_1]$.

- If $\mathrm{q} = \pi_0;\mathrm{q}_0$ (the case $\mathrm{q} = \pi_1;\mathrm{q}_1$ is symmetric),

  then by the induction hypothesis some $v' \leq v0$ has a copointed subnet q'_0 in $\sigma\mathrm{f}'$. If $v' \leq v$ let $\mathrm{q}' = \mathrm{q}'_0$, otherwise let $\mathrm{q}' = \pi_0;\mathrm{q}_0$.

□

The main argument is then carried out by the following lemma.

Lemma 4.6.3. *Let* f *and* g *be parallel nets such that* $\textsc{maxcp}(\sigma f) = \textsc{maxcp}(\sigma g)$, *and whose target is not pointed. Then there is a net* h *with the following properties:*

(1) $f;\iota_j(Y) \Leftrightarrow h;\iota_j(Y)$ *if* Y *is pointed;*

(2) $\textsc{maxcp}(\sigma h) = \textsc{maxcp}(\sigma f) = \textsc{maxcp}(\sigma g)$*;*

(3) $(\sigma h)_{v,\varepsilon} = (\sigma g)_{v,\varepsilon}$ *for any* $v \in \textsc{maxcp}(\sigma h)$.

Proof. Item (2) is present solely for the purpose of clarity, as it follows from (1): by completeness (Theorem 3.2.3) $\sigma(f;\iota_j) = \sigma(h;\iota_j)$; then Lemma 4.6.2 for f and h gives

$$\textsc{maxcp}(\sigma f) = \textsc{maxcp}(\sigma(f;\iota_j)) = \textsc{maxcp}(\sigma(h;\iota_j)) = \textsc{maxcp}(\sigma h) .$$

Items (1) and (3) will be shown by induction on X.

If $\textsc{maxcp}(\sigma f) = \textsc{maxcp}(\sigma g) = \varnothing$, which is precisely when σf and σg contain no rooted initial links, then both (1) and (3) are immediate for h = f.

If $\textsc{maxcp}(\sigma f) = \textsc{maxcp}(\sigma g) = \{\varepsilon\}$ (note that by minimality, if ε has a maximal copointed subnet, no other vertex does), let h = g, so that (3) is immediate. For (1), let $q \subseteq \sigma f$ and $k \subseteq \sigma g$ be copointed subnets. Lemma 4.5.1 gives $q \Leftrightarrow f$ and $k \Leftrightarrow g$. In the context of the injection, $q;\iota_j(Y)$ and $k;\iota_j(Y)$ are equivalent to copointed nets q' and k', respectively. Then if Y is pointed both are bipointed, and equivalent by Lemma 4.3.5, completing the equivalence chain below.

$$f;\iota_j \quad\Leftrightarrow\quad q;\iota_j \quad\Leftrightarrow\quad q' \quad\Leftrightarrow\quad k' \quad\Leftrightarrow\quad k;\iota_j \quad\Leftrightarrow\quad g;\iota_j$$

In the remaining case, some vertex v other than ε has a maximal copointed subnet in σf. By Lemma 4.5.2 f is equivalent to a net containing this copointed subnet, which must then be left-constructible (if it was basic, it would be $(\mathbf{0}, Y_j, *)$, but then $\textsc{maxcp}(\sigma f)$ would be $\{\varepsilon\}$, a case already considered). In g, the same vertex v must have a copointed subnet, too; then g is likewise equivalent to a left-constructible net, and moreover if the source of f and g is a product, both have equivalent nets constructed over the same projection. Thus, there are two cases to consider:

$$f \Leftrightarrow [f_0, f_1] \quad\text{and}\quad g \Leftrightarrow [g_0, g_1] \qquad\qquad f \Leftrightarrow \pi_i;f' \quad\text{and}\quad g \Leftrightarrow \pi_i;g' .$$

- In the first case, where $\mathsf{f} \Leftrightarrow [\mathsf{f}_0,\mathsf{f}_1]$ and $\mathsf{g} \Leftrightarrow [\mathsf{g}_0,\mathsf{g}_1]$, the saturation of each f_i and g_i is a sub-pre-net of that of f and g, according to Lemma 4.2.6, as follows.

  $$\sigma\mathsf{f}_i = (\sigma\mathsf{f})_{i,\varepsilon} \qquad\qquad \sigma\mathsf{g}_i = (\sigma\mathsf{g})_{i,\varepsilon}$$

  Since $\varepsilon \notin \textsc{maxcp}(\sigma\mathsf{f})$, i.e. there is no copointed subnet $\mathsf{q} \subseteq \sigma\mathsf{f}$, any maximal copointed subnet q in σf is between a vertex iv and the right root ε, and is also a maximal copointed subnet between v and ε in f_i. This gives the following.

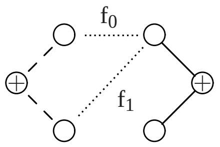
  $$\begin{aligned} \textsc{maxcp}(\sigma\mathsf{f}) &= \{iv \mid v \in \textsc{maxcp}(\sigma\mathsf{f}_i)\} \\ \textsc{maxcp}(\sigma\mathsf{g}) &= \{iv \mid v \in \textsc{maxcp}(\sigma\mathsf{g}_i)\} \end{aligned}$$

  Then $\textsc{maxcp}(\sigma\mathsf{f}_i) = \textsc{maxcp}(\sigma\mathsf{g}_i)$. The induction hypothesis gives nets h_0 and h_1 such that each h_i satisfies (1), (2) and (3) w.r.t. f_i and g_i (note that the target of f_i and g_i is not pointed, as required for the induction hypothesis, because it is the same as that of f and g).

  Let $\mathsf{h} = [\mathsf{h}_0,\mathsf{h}_1]$. The following equations show that h satisfies (1), i.e. that the equivalence $\mathsf{f};\iota_j \Leftrightarrow \mathsf{h};\iota_j$ holds for injections into a pointed target.

  $$\begin{array}{ccccc} \mathsf{f};\iota_j & \Leftrightarrow & [\mathsf{f}_0,\mathsf{f}_1];\iota_j & = & [(\mathsf{f}_0;\iota_j),(\mathsf{f}_1;\iota_j)] \\ & & & & \Updownarrow \\ \mathsf{h};\iota_j & = & [\mathsf{h}_0,\mathsf{h}_1];\iota_j & = & [(\mathsf{h}_0;\iota_j),(\mathsf{h}_1;\iota_j)] \end{array}$$

  They are justified by the equivalence $(\mathsf{f}_i;\iota_j) \Leftrightarrow (\mathsf{h}_i;\iota_j)$, which is the property (1) for h_0 and h_1, and the equations for bi-constructible nets in Proposition 2.5.1, e.g. that $[\mathsf{f}_0,\mathsf{f}_1];\iota_0$ and $[(\mathsf{f}_0;\iota_0),(\mathsf{f}_1;\iota_0)]$ denote the same net, illustrated below.

  Next, h satisfies (2) as it follows from (1), which means that the same vertices have maximal copointed subnets in σf, σg, and σh. Then, since $\varepsilon \notin \textsc{maxcp}(\sigma\mathsf{h})$ and because Lemma 4.2.4 gives $(\sigma\mathsf{h})_{i,\varepsilon} = \sigma\mathsf{h}_i$,

  $$\textsc{maxcp}(\sigma\mathsf{h}) = \{iv \mid v \in \textsc{maxcp}(\sigma\mathsf{h}_i)\} .$$

  By the equations below h satisfies (3): for any vertex iv in $\textsc{maxcp}(\sigma\mathsf{h})$,

  $$(\sigma\mathsf{h})_{iv,\varepsilon} = (\sigma\mathsf{h}_i)_{v,\varepsilon} = (\sigma\mathsf{g}_i)_{v,\varepsilon} = (\sigma\mathsf{g})_{iv,\varepsilon} .$$

  The middle equation is due to h_i and g_i satisfying (3), while the first and last follow from Lemma 4.2.4.

- In the second case, without loss of generality let $\mathrm{f} \Leftrightarrow \pi_0;\mathrm{f}'$ and $\mathrm{g} \Leftrightarrow \pi_0;\mathrm{g}'$. To apply the induction hypothesis to f' and g' it must be shown that their saturations have maximal copointed subnets at the same vertices. Let $\sigma\mathrm{g}$ and $\sigma\mathrm{g}'$ be described as follows, as in Lemma 4.4.1b.

  $$\sigma\mathrm{g}' = (X_0, Y', \mathcal{R}) \qquad \sigma\mathrm{g} = (X, Y', \mathcal{S}) \qquad \mathcal{S} = (0 \cdot \mathcal{R}) \cup \Gamma \cup \Delta$$

  Let $\mathrm{q} \subseteq (\sigma\mathrm{g})_{v,\varepsilon}$ be a maximal copointed subnet. If any link $\langle u, \varepsilon\rangle$ in q is in Γ, then Y' must be $\mathbf{1}$, while if $\langle u, \varepsilon\rangle$ is in Δ, then there must be a pointed subnet $\mathrm{p} \subseteq \sigma\mathrm{g}'$. In both cases Y', the target of f and g, must be pointed, which contradicts the assumption that it isn't. Consequently, all links in q must be in $(0 \cdot \mathcal{R})$, forming a maximal copointed subnet q' in $\sigma\mathrm{g}'$. This gives the two statements below (the first by repeating the argument for f).

  $$\begin{aligned} \textsc{maxcp}(\sigma\mathrm{f}) &= \{0v \mid v \in \textsc{maxcp}(\sigma\mathrm{f}')\} \\ \textsc{maxcp}(\sigma\mathrm{g}) &= \{0v \mid v \in \textsc{maxcp}(\sigma\mathrm{g}')\} \end{aligned}$$

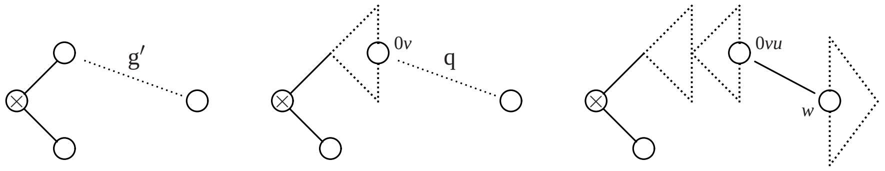
  Then $\textsc{maxcp}(\sigma\mathrm{f}') = \textsc{maxcp}(\sigma\mathrm{g}')$, and the induction hypothesis gives a net h' satisfying (1), (2) and (3). Let $\mathrm{h} = \pi_0;\mathrm{h}'$. That h satisfies (1) follows by the equations below (the centre one is (1) for h').

  $$\mathrm{f};\iota_j \quad \Leftrightarrow \quad \pi_0;\mathrm{f}';\iota_j \quad \Leftrightarrow \quad \pi_0;\mathrm{h}';\iota_j \quad = \quad \mathrm{h};\iota_j\ .$$

  As (1) implies (2), $\textsc{maxcp}(\sigma\mathrm{h}) = \textsc{maxcp}(\sigma\mathrm{f})$, and as for f and g earlier, the following holds for h.

  $$\textsc{maxcp}(\sigma\mathrm{h}) \ = \ \{0v \mid v \in \textsc{maxcp}(\sigma\mathrm{h}')\}$$

  Then for (3) it must be shown that the sub-pre-nets of $\sigma\mathrm{h}$ and $\sigma\mathrm{g}$ between a vertex $0v \in \textsc{maxcp}(\sigma\mathrm{h})$ and the right root ε are equal. Let $\langle 0vu, w\rangle$ be a link in $\sigma\mathrm{g}$, so that $\langle u, w\rangle$ is a link in the sub-pre-net $(\sigma\mathrm{g})_{0v,\varepsilon}$. Let q be a maximal copointed subnet between $0v$ and ε in $\sigma\mathrm{g}$, and between v and ε in $\sigma\mathrm{g}'$.

  It will be shown that $\langle 0vu, w\rangle$ is in $\sigma\mathrm{h}$. Recall that $\mathcal{R}$ and $\mathcal{S} = (0 \cdot \mathcal{R}) \cup \Gamma \cup \Delta$ denote the links in $\sigma\mathrm{g}'$ and $\sigma\mathrm{g}$ respectively. If $\langle 0vu, w\rangle$ is in $(0 \cdot \mathcal{R})$ then $\langle vu, w\rangle$ is in $\sigma\mathrm{g}'$, and because h' satisfies (3) and v has a maximal copointed subnet in

$\sigma h'$, the link $\langle vu, w\rangle$ is in $\sigma h'$, and $\langle 0vu, w\rangle$ is in σh. Otherwise, if $\langle 0vu, w\rangle$ is in Γ or Δ, then some $w' \leq w$ is pointed: in the first case because w is $\mathbf{1}$, in the second because some $w' \leq w$ has a maximal pointed subnet. Since $0v \in \textsc{maxcp}(\sigma h)$ there is a copointed subnet q' between $0v$ and ε in σh. By moving it down from ε to w', there is a copointed subnet q'' between $0v$ and w'. Then q'' is bipointed, and by Lemma 4.3.7 σh is full between $0v$ and w', and must contain $\langle 0vu, w\rangle$.

The reverse argument, that a link $\langle 0vu, w\rangle$ in σh must be in σg, is symmetric to the above case. Then h satisfies (3).

□

The final case of the soundness proof can now be concluded.

Lemma 4.6.4. *For parallel nets* f *and* g *whose target is not pointed, if* $f;\iota_j(Y)$ *and* $g;\iota_j(Y)$ *have the same saturation and* Y *is pointed, there is a net* h *such that* $f;\iota_j(Y)$ *and* $h;\iota_j(Y)$ *are equivalent and* g *and* h *have the same saturation.*

Proof. Because $f;\iota_j$ and $g;\iota_j$ have the same saturation, and by Lemma 4.6.2, the following equations hold.

$$\textsc{maxcp}(\sigma f) \;=\; \textsc{maxcp}(\sigma(f;\iota_j)) \;=\; \textsc{maxcp}(\sigma(g;\iota_j)) \;=\; \textsc{maxcp}(\sigma g)$$

Then Lemma 4.6.3 applies to f and g, giving the net h such that

(1) $f;\iota_j \;\Leftrightarrow\; h;\iota_j$,

(2) $\textsc{maxcp}(\sigma h) \;=\; \textsc{maxcp}(\sigma f) \;=\; \textsc{maxcp}(\sigma g)$, and

(3) $(\sigma h)_{v,\varepsilon} \;=\; (\sigma g)_{v,\varepsilon}$ for any $v \in \textsc{maxcp}(\sigma h)$.

It remains to show that $\sigma g = \sigma h$. Using Lemma 4.4.1, let the saturations of the nets involved be given by the following equations—note that by (1) and completeness (Theorem 3.2.3) $f;\iota_j$ and $h;\iota_j$ have the same saturation.

$$\sigma(f;\iota_j) \;=\; \sigma(g;\iota_j) \;=\; \sigma(h;\iota_j) \;=\; (X, Y, \mathcal{S})$$

$$\sigma g \;=\; (X, Y_j, \mathcal{R}) \qquad \mathcal{S} \;=\; (\mathcal{R}\cdot j) \cup \Gamma \cup \Delta$$

$$\sigma h \;=\; (X, Y_j, \mathcal{R}') \qquad \mathcal{S} \;=\; (\mathcal{R}'\cdot j) \cup \Gamma' \cup \Delta'$$

It will be shown that $\mathcal{R} \subseteq \mathcal{R}'$; the reverse follows symmetrically. Let $\langle v, w\rangle$ be a link in $\mathcal{R}$. Then $\langle v, jw\rangle \in (\mathcal{R}\cdot j) \subseteq \mathcal{S}$. The case $\langle v, jw\rangle \in (\mathcal{R}'\cdot j)$ is immediate. Otherwise, some $v' \leq v$ has a maximal copointed subnet in $\sigma\mathrm{h}$—if $\langle v, jw\rangle$ is in Δ', by definition, and if it is in Γ', because $\langle v, \varepsilon\rangle$ is a rooted initial link in $\sigma\mathrm{h}$. Then by (3), $(\sigma\mathrm{h})_{v',\varepsilon} = (\sigma\mathrm{g})_{v',\varepsilon}$, and $\langle v, w\rangle$ is in $\mathcal{R}'$.

□

4.7 Finale

To complete the soundness proof is a matter of connecting the different lemmata.

Proof of Theorem 3.2.4 (Soundness). *For* $\Sigma\Pi(\mathcal{C})$*-nets* f *and* g, *if* $\sigma\mathrm{f} = \sigma\mathrm{g}$ *then* $\mathrm{f} \Leftrightarrow \mathrm{g}$.

Proof. Let f and g be parallel nets with source X and target Y. The proof is by induction on X and Y.

- If X is an atom or **1** then $\sigma\mathrm{f} = \mathrm{f}$ and $\sigma\mathrm{g} = \mathrm{g}$, so that $\mathrm{f} = \mathrm{g}$. The same holds when Y is an atom or **0**. If X is **0** or $Y = \mathbf{1}$, then $\mathrm{f} \Leftrightarrow \mathrm{g}$ by Lemma 4.2.1.

- If X is a coproduct, then by Lemma 4.2.4 f is equivalent to a net $[\mathrm{f}_0, \mathrm{f}_1]$, and g to a net $[\mathrm{g}_0, \mathrm{g}_1]$. Lemma 4.2.6 gives the equations below, showing that f_i and g_i have the same saturation (for $i \in \{0,1\}$).

  $$\sigma\mathrm{f}_i \;=\; (\sigma\mathrm{f})_{i,\varepsilon} \;=\; (\sigma\mathrm{g})_{i,\varepsilon} \;=\; \sigma\mathrm{g}_i$$

  The induction hypothesis gives $\mathrm{f}_i \Leftrightarrow \mathrm{g}_i$, from which the equation below follows.

  $$\mathrm{f} \;\Leftrightarrow\; [\mathrm{f}_0, \mathrm{f}_1] \;\Leftrightarrow\; [\mathrm{g}_0, \mathrm{g}_1] \;\Leftrightarrow\; \mathrm{g}$$

  The case where Y is a product is dual.

- In the remaining case X is a product and Y a coproduct. If the saturation of f and g is constructible, say of the form $\pi_0;\mathrm{h}$ (without loss of generality), then accordingly f and g are of the form $\pi_0;\mathrm{f}'$ and $\pi_0;\mathrm{g}'$ respectively. Lemma 4.4.1 gives the equations below since, in the terminology of the lemma, Γ and Δ are empty for both f' and g'.

  $$\pi_0;\sigma\mathrm{f}' \;=\; \sigma(\pi_0;\mathrm{f}') \;=\; \sigma(\pi_0;\mathrm{g}') \;=\; \pi_0;\sigma\mathrm{g}'$$

  As $\sigma\mathrm{f}' = \sigma\mathrm{g}'$ the induction hypothesis gives $\mathrm{f}' \Leftrightarrow \mathrm{g}'$, so that

  $$\mathrm{f} \;=\; \pi_i;\mathrm{f} \;\Leftrightarrow\; \pi_i;\mathrm{g} \;=\; \mathrm{g}\ .$$

If the saturation of f and g is not constructible, then by Lemma 4.5.2 they are equivalent to nets constructed over the same projection or injection, say

$$\mathrm{f} \Leftrightarrow \mathrm{f}';\iota_0 \qquad \mathrm{g} \Leftrightarrow \mathrm{g}';\iota_0 \,.$$

If Y is not pointed or Y_0 is pointed, by Lemma 4.6.1

$$\sigma\mathrm{f}' = (\sigma\mathrm{f})_{\varepsilon,0} = (\sigma\mathrm{g})_{\varepsilon,0} = \sigma\mathrm{g}'$$

from which the induction hypothesis gives $\mathrm{f}' \Leftrightarrow \mathrm{g}'$. It follows that

$$\mathrm{f} \Leftrightarrow \mathrm{f}';\iota_0 \Leftrightarrow \mathrm{g}';\iota_0 \Leftrightarrow \mathrm{g} \,.$$

Finally, if Y is pointed and Y_0 is not pointed, then Lemma 4.6.4 gives a net h such that $\mathrm{h};\iota_0 \Leftrightarrow \mathrm{f}';\iota_0$ and $\sigma\mathrm{h} = \sigma\mathrm{g}'$. By the induction hypothesis, $\mathrm{h} \Leftrightarrow \mathrm{g}'$. This completes the equivalence of f and g, as below.

$$\mathrm{f} \Leftrightarrow \mathrm{f}';\iota_0 \Leftrightarrow \mathrm{h};\iota_0 \Leftrightarrow \mathrm{g}';\iota_0 \Leftrightarrow \mathrm{g}$$

□

4.8 Characterising saturated nets

The main lemmata of the soundness proof provide a basis from which to complete two outstanding proofs from Chapter 3. The first is the proof of Proposition 3.3.2, that a saturated net is the union over an equivalence class of nets. This will be completed in the present section. The second is the proof of Proposition 3.4.5, the correctness condition for saturated nets. This will be completed in the next section, where, in addition, a sequentialisation algorithm for saturated nets will be given.

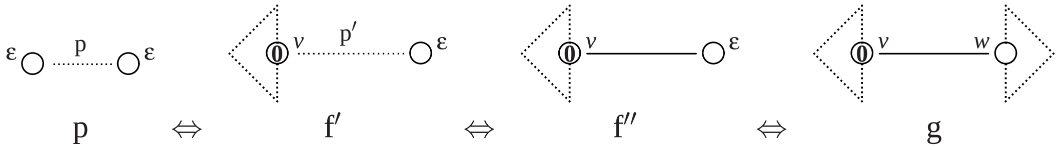
Formally, Proposition 3.3.2 states that

$$\sigma\mathrm{f} = \bigcup\{\mathrm{g} \mid \mathrm{f} \Leftrightarrow \mathrm{g}\} \,.$$

(Note that this does not itself imply soundness, which requires that different equivalence classes must have different unions.) To prove the proposition, it must be shown that any link in a saturated net σf occurs in a net equivalent to f. This will first be shown for the saturation of a bipointed net.

Lemma 4.8.1. *For a bipointed net* f *and a unit link* $\langle v,*,w\rangle$ *in* σf *there is a net* $\mathrm{g} \Leftrightarrow \mathrm{f}$ *containing* $\langle v,*,w\rangle$.

Proof. Let $\langle v,*,w\rangle$ be an initial link (the case for terminal links is dual) and without loss of generality let f be a pointed net $\mathrm{p} = (Q,P,\mathcal{P})$ (by Lemma 4.3.5 even if f itself is not pointed it is equivalent to a pointed net). Moving p down from the left root, as in Lemma 4.3.2, gives an equivalent net f' with a pointed subnet p' between v and ε. Because $\langle v,*,w\rangle$ is an initial link, $Q_v = \mathbf{0}$ and p' is an initial net, equivalent to $?_P = (\mathbf{0},P,*)$ by Lemma 4.2.1. Consequently, f' is equivalent to $\mathrm{f}'' = \mathrm{f}'\{?\}_{v,\varepsilon}$.



Finally, the net g containing $\langle v,*,w\rangle$ is obtained by moving the initial link $\langle v,\varepsilon\rangle$ in f'' down towards w. $\square$

The proof of the general proposition is completed below.

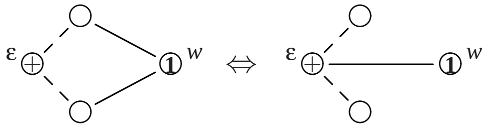
Proof of Proposition 3.3.2. *The saturation of a net* f *is*

$$\bigcup \{\mathrm{g} \mid \mathrm{f} \Leftrightarrow \mathrm{g}\} \; .$$

Proof. One direction, that $\bigcup\{\mathrm{g} \mid \mathrm{f} \Leftrightarrow \mathrm{g}\} \subseteq \sigma\mathrm{f}$, is immediate from completeness (Theorem 3.2.3). For the other it will be shown, by induction on the construction of f, that any link $\langle v,w\rangle$ in $\sigma\mathrm{f}$ belongs to some net $\mathrm{g} \Leftrightarrow \mathrm{f}$.

- For basic nets, if f is atomic then $\sigma\mathrm{f} = \mathrm{f}$. Next, if f is an initial net $(\mathbf{0},Y,*)$, for any link $\langle\varepsilon,*,w\rangle$ in its (full) saturation a net g can be found by moving the link $\langle\varepsilon,*,\varepsilon\rangle$ in f down towards the leaves. The case for a terminal net $\mathrm{f} = (X,\mathbf{1},*)$ is dual.

- If $\mathrm{f} = [\mathrm{f}_0,\mathrm{f}_1]$ then by Lemma 4.2.5 its saturation is $(X,Y,\mathcal{R}\cup\mathcal{S})$, where $\mathcal{R}$ are the combined links if $\sigma\mathrm{f}_0$ and $\sigma\mathrm{f}_1$, and $\mathcal{S}$ contains precisely the rooted terminal links $\langle\varepsilon,u\rangle$ for which also both $\sigma\mathrm{f}_0$ and $\sigma\mathrm{f}_1$ contain a rooted terminal link $\langle\varepsilon,u\rangle$. For the link $\langle v,w\rangle$, if $v = 0v'$ the induction hypothesis on $\sigma\mathrm{f}_0$ gives a net g_0 containing $\langle v',w\rangle$. Then $\mathrm{g} = [\mathrm{g}_0,\mathrm{f}_1]$ is equivalent to f and contains $\langle v,w\rangle$. The case for $v = 1v'$ is symmetric, leaving that for $v = \varepsilon$. In that case, $\langle\varepsilon,w\rangle$ must be in $\mathcal{S}$, and $\langle 0,w\rangle$ and $\langle 1,w\rangle$ are in $\mathcal{R}$. The induction hypothesis, applied to f_0 and f_1, gives a net $[\mathrm{g}_0,\mathrm{g}_1]$ containing $\langle 0,w\rangle$ and $\langle 1,w\rangle$. Then g is obtained by a single rewrite step applied to these links.

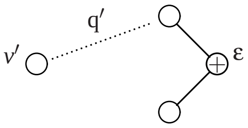


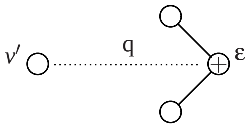
So $\langle v,w\rangle$ is in g. The case $\mathrm{f} = \langle \mathrm{f}_0, \mathrm{f}_1\rangle$ is dual.

- If $\mathrm{f} = \mathrm{f}';\iota_0$ then let $\sigma\mathrm{f} = (X, Y, \mathcal{S})$ and $\sigma\mathrm{f}' = (X, Y_0, \mathcal{R})$, so that $\mathcal{S} = (\mathcal{R}\cdot 0) \cup \Gamma \cup \Delta$ as in Lemma 4.4.1. If $\langle v,w\rangle$ is a link $\langle v, 0w'\rangle$ in $\mathcal{R}\cdot 0$, then $\langle v,w'\rangle$ is in $\sigma\mathrm{f}'$. The induction hypothesis gives a net $\mathrm{g}' \Leftrightarrow \mathrm{f}'$ containing $\langle v,w'\rangle$. Then $\mathrm{g} = \mathrm{g}';\iota_0$ is equivalent to f and contains $\langle v,w\rangle$.

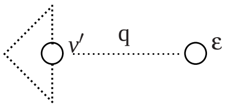
  If $\langle v,w\rangle$ is in Γ then $\langle v,\varepsilon\rangle$ is a rooted initial link in $\sigma\mathrm{f}$. This link forms a partial copointed subnet, for which Lemma 4.5.1 gives a net $\mathrm{g}' \Leftrightarrow \mathrm{f}$ containing $\langle v,\varepsilon\rangle$. By moving the initial link $\langle v,\varepsilon\rangle$ down from the right root to w, the net g is obtained from g'.

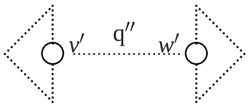
  If $\langle v,w\rangle$ is in Δ, then some $v' \leq v$ has a maximal copointed subnet q' in $\sigma\mathrm{f}'$, and some $w' \leq w$ is pointed. Then $\sigma\mathrm{f}$ contains a copointed subnet q between v and ε, found by moving q' up to the root.





  The copointed subnet q constitutes a partial copointed subnet of $\sigma\mathrm{f}$, for which Lemma 4.5.1 gives a net $\mathrm{g}' \Leftrightarrow \mathrm{f}$ such that $\mathrm{g}'_{v',\varepsilon} = \mathrm{q}$.





  In g', by moving q down towards w', an equivalent net g'' is obtained containing a copointed subnet $\mathrm{q}'' = \mathrm{g}'_{v',w'}$. Because w' is pointed q'' is bipointed, and as $\sigma\mathrm{q}''$ is full it contains the link $\langle v'', w''\rangle$, where $v = v'v''$ and $w = w'w''$. Then by Lemma 4.8.1 there is an equivalent net $\mathrm{k} \Leftrightarrow \mathrm{q}''$ containing $\langle v'', w''\rangle$. Finally, $\mathrm{g} \Leftrightarrow \mathrm{f}$ is obtained from g'' by replacing q'' with k.

  The case $\mathrm{f} = \mathrm{f}';\iota_1$ is symmetric, and $\mathrm{f} = \pi_i;\mathrm{f}'$ is dual.

$\square$

4.9 Sequentialisation

An important aspect of saturated nets still to be addressed is a *sequentialisation* procedure: a translation from saturated nets back to sum–product terms. As was mentioned in Section 1.3, in general, sequentialisation and correctness of proof nets are closely related. In the case of saturated nets, sequentialisation will naturally proceed via a notion of *desaturation*, a translation from saturated nets to nets that is inverse, up to equivalence, to saturation. Such a desaturation procedure will be provided by the outstanding proof of Proposition 3.4.5, the correctness condition for saturated nets, that will be completed in this section. The proposition states that a prenet that is connected, saturated, and close-knit is a saturated net. Since a (constructive) proof of this proposition provides a net of whose saturation is again the original prenet, it will naturally constitute a desaturation algorithm. Before making this explicit, a simpler approach to desaturation will briefly be demonstrated to be inadequate.

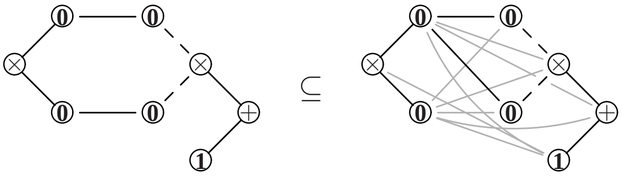


Figure 4.6: One saturated net as a subnet of another

A natural question is whether simply taking a subnet of a saturated net constitutes, in itself, a desaturation method. This turns out not to be the case, as is illustrated by Figure 4.6. The figure displays two nets, with their saturation included in grey; the left one, which is already saturated, is a subnet of the saturation of the right one. Thus, saturating a subnet of a saturated net is not the identity relation.

The desaturation algorithm used in the proof Proposition 3.4.5 is given below. The algorithm is non-deterministic, which is natural, given the fact that it finds one net from an equivalence class of nets. Although it may be possible, technically, to construct a deterministic desaturation algorithm, this would require non-canonical choices, for example between source object and target objects, or between the two projections of a product. Also, in the present formulation, not all nets in an equivalence class are found, non-deterministically, by desaturation. It is not unlikely that giving a desaturation algorithm that does return all nets, i.e. one that is the inverse relation to saturation, would be possible. (This was not pursued, for the reason that it is likely to require significant effort to find all the nets whose saturation is that of a (co)pointed net, while the (co)pointed net itself is readily found.)

Definition 4.9.1 (Desaturation)**.** A *desaturation* of a prenet $\mathrm{h} = (X, Y, \mathcal{R})$ is a prenet f obtained by the following algorithm.

- If $X = \mathbf{1}$, $Y = \mathbf{0}$, or one of X and Y is atomic, let f be h; if $X = \mathbf{0}$ let f be $(\mathbf{0}, Y, *)$; if $Y = \mathbf{1}$ let f be $(X, \mathbf{1}, *)$.

- If $X = X_0 + X_1$ then recursively obtain prenets f_0 from $\mathrm{h}_{0,\varepsilon}$ and f_1 from $\mathrm{h}_{1,\varepsilon}$, and let $\mathrm{f} = [\mathrm{f}_0, \mathrm{f}_1]$. If $Y = Y_0 \times Y_1$ then recursively obtain prenets f_0 from $\mathrm{h}_{\varepsilon,0}$ and f_1 from $\mathrm{h}_{\varepsilon,1}$, and let $\mathrm{f} = \langle \mathrm{f}_0, \mathrm{f}_1 \rangle$.

- If X is a product and Y a coproduct then of the sub-prenets $\mathrm{h}_{i,\varepsilon}$ and $\mathrm{h}_{\varepsilon,j}$ of h, choose one that is connected. For $\mathrm{h}_{\varepsilon,j}$, construct the sub-prenet $\mathrm{g} \subseteq \mathrm{h}_{\varepsilon,j}$ as follows. For each $u \in \textsc{maxcp}(\mathrm{h})$ choose a copointed subnet $\mathrm{q}_u \subseteq \mathrm{h}_{u,j}$, then let $\textsc{maxcp}(\mathrm{h}) = \{u_1, \ldots, u_n\}$ and construct the following series of pre-nets.

  $$\mathrm{h}_{\varepsilon,j} = \mathrm{g}_0,\ \mathrm{g}_1,\ \ldots,\ \mathrm{g}_n = \mathrm{g} \qquad \text{where} \qquad \mathrm{g}_i = \mathrm{g}_{i-1}\{\sigma \mathrm{q}_{u_i}\}_{u_i,\varepsilon}$$

  From g recursively obtain a net f'; let $\mathrm{f} = \mathrm{f}';\iota_j$. For a sub-prenet $\mathrm{h}_{i,\varepsilon}$ the procedure is dual.

The desaturation algorithm is essentially an inversion of the inductive description of saturation, in Lemma 4.2.5 and, mainly, Lemma 4.4.1. In particular the third case, where X is a product and Y a coproduct, reverses the process of obtaining the saturation $\sigma(\mathrm{f}';\iota_j)$ from $\sigma\mathrm{f}';\iota_j$ as described in Lemma 4.4.1. There, $\sigma(\mathrm{f}';\iota_j)$ is given as $(\sigma\mathrm{f}';\iota_j) \cup \Gamma \cup \Delta$ (abusing notation), where Γ contains the duplication of rooted initial links in $\sigma\mathrm{f}'$, and Δ contains the bipointed nets formed by links in Γ. To reverse this operation, for a prenet h with a connected sub-prenet $\mathrm{h}_{\varepsilon,j}$, for each vertex u that has a maximal copointed subnet, desaturation takes $\mathrm{h}_{\varepsilon,j}$ and replaces the sub-prenet between u and ε with $\sigma(\mathrm{q}_u)$, the saturation of a copointed subnet $\mathrm{q}_u \subseteq \mathrm{h}_{u,j}$. The idea behind this treatment is that if the saturated net $\sigma\mathrm{f}'$ contains a maximal copointed subnet q_u, then the subnet $(\sigma\mathrm{f}')_{u,\varepsilon}$ is precisely $\sigma(\mathrm{q}_u)$.

It remains to be shown the this algorithm yields the desired result, i.e. that for saturated nets, desaturation has saturation as its inverse. This is shown by Proposition 4.9.2 below, which, together with sequentialisation for nets (Corollary 2.4.5), gives sequentialisation for saturated nets. The proof of this statement will be combined with that of the correctness condition, in Lemma 4.9.3 below.

Proposition 4.9.2. *For a saturated net* h *desaturation gives a net* f *such that* $\sigma\mathrm{f} = \mathrm{h}$.

Proof. The statement follows from Lemma 4.9.3, below, since a saturated net is connected, saturated, and close-knit by Proposition 3.4.4. ☐

Restatement of Proposition 3.4.5. *If a pre-net* h *is connected, saturated, and close-knit, it is a saturated net* $\sigma\mathrm{f}$.

Proof. By Lemma 4.9.3, below. ☐

Lemma 4.9.3. *If a pre-net* h *is connected, saturated, and close-knit, then desaturating it gives a net* f *such that* $\sigma\mathrm{f} = \mathrm{h}$.

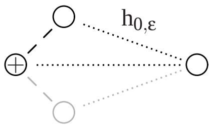
Proof. The proof will naturally follow the desaturation algorithm.

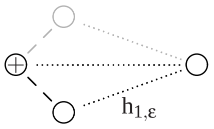
- If $X = \mathbf{1}$, $Y = \mathbf{0}$, or one of X and Y is atomic, let f be h; if $X = \mathbf{0}$ let f be $(\mathbf{0}, Y, *)$; if $Y = \mathbf{1}$ let f be $(X, \mathbf{1}, *)$.

If $X = \mathbf{1}$, $Y = \mathbf{0}$, or either is atomic, the neighbouring relation $\frown$ must be empty since all links in $\mathcal{R}$ connect only to leaves. As h is close-knit it must be then compatible, i.e. its switchings switch on at most one link, and since it is also connected it is a net. Next, if $X = \mathbf{0}$ or $Y = \mathbf{1}$ then h is full since it is connected and saturated, and f may be chosen as $?_Y$ or $!_X$ respectively.

- If $X = X_0 + X_1$ then by induction obtain prenets f_0 from $\mathrm{h}_{0,\varepsilon}$ and f_1 from $\mathrm{h}_{1,\varepsilon}$, and let $\mathrm{f} = [\mathrm{f}_0, \mathrm{f}_1]$. If $Y = Y_0 \times Y_1$ then by induction obtain prenets f_0 from $\mathrm{h}_{\varepsilon,0}$ and f_1 from $\mathrm{h}_{\varepsilon,1}$, and let $\mathrm{f} = \langle \mathrm{f}_0, \mathrm{f}_1 \rangle$.

The case where X is a coproduct will be shown, that where Y is a product is dual. It is immediate that the sub-prenets $\mathrm{h}_{0,\varepsilon}$ and $\mathrm{h}_{1,\varepsilon}$ are saturated; that they are also connected and close-knit will be established below.





Let $\varsigma = (\varsigma_L, \varsigma_R)$ be a switching for $\mathrm{h}_{0,\varepsilon}$, and let $\tau = (\tau_L, \tau_R)$ be a switching on h that agrees with ς on vertices in $\mathrm{h}_{0,\varepsilon}$ and chooses 0 on ε in X, i.e.,

$$\tau_L(\varepsilon) \;=\; 0\,, \quad \tau_L(0u) \;=\; \varsigma_L(u)\,, \quad \text{and} \quad \tau_R \;=\; \varsigma_R\,.$$

To show $\mathrm{h}_{0,\varepsilon}$ is connected, let $\tau \mathbin{\text{⏻}} \langle v, w \rangle$ in h. If $v = 0v'$ then $\varsigma \mathbin{\text{⏻}} \langle v', w \rangle$ in $\mathrm{h}_{0,\varepsilon}$. Otherwise, if $v = \varepsilon$ then $\langle v, w \rangle$ is a terminal link, and since h is saturated it contains also $\langle 0, w \rangle$, so that $\varsigma \mathbin{\text{⏻}} \langle \varepsilon, w \rangle$ in $\mathrm{h}_{0,\varepsilon}$. Note that τ would switch off v in case $1 \leq v$. Next, it will be shown that $\mathrm{h}_{0,\varepsilon}$ is close-knit. By design, $\tau_L \mathbin{\text{⏻}} 0v$ if and only if $\varsigma_L \mathbin{\text{⏻}} v$, while $\tau_R = \varsigma_R$;

this means that $\tau \text{⏻} \langle 0v,w\rangle$ if and only if $\varsigma \text{⏻} \langle v,w\rangle$. Then if $\varsigma \text{⏻} \langle v_1,w_1\rangle, \langle v_n,w_n\rangle$ in $\mathrm{h}_{0,\varepsilon}$ the corresponding links $\langle 0v_1,w_1\rangle$ and $\langle 0v_n,w_n\rangle$ in h must be connected by a path of neighbours,

$$\langle 0v_1,w_1\rangle \frown_\tau \ \dots \ \frown_\tau \langle 0v_n,w_n\rangle \ .$$

This translates directly into a path of neighbours in $\mathrm{h}_{0,\varepsilon}$, unless some v_i is ε (no vertex $1v$ in X is switched on by τ). But a link $\langle \varepsilon,w_i\rangle$ has only one neighbour, $\langle 0,w_i\rangle$; then the path in h must contain the segment below left, which can be replaced by that below right.

$$\dots\langle 0,w\rangle \frown_\tau \langle \varepsilon,w\rangle \frown_\tau \langle 0,w\rangle\dots \qquad\qquad \dots\langle 0,w\rangle\dots$$

After so removing all edges $\langle \varepsilon,w\rangle$ from the path of neighbours in h, it translates into a path $\langle v_1,w_1\rangle \frown_\varsigma^* \langle v_n,w_n\rangle$ in $\mathrm{h}_{0,\varepsilon}$. This shows that $\mathrm{h}_{0,\varepsilon}$ is close-knit. By a symmetric argument, also $\mathrm{h}_{1,\varepsilon}$ is connected, saturated, and close-knit.

Applying the induction hypothesis gives nets g_0 and g_1 such that $\sigma(\mathrm{g}_i) = \mathrm{h}_{i,\varepsilon}$; let $\mathrm{g} = [\mathrm{g}_0,\mathrm{g}_1]$. Since h is saturated and $\mathrm{g} \subseteq \mathrm{h}$ it holds that $\sigma\mathrm{g} \subseteq \mathrm{h}$, and it follows by Lemma 4.2.5 that $\mathrm{h} \subseteq \sigma\mathrm{g}$, so that $\mathrm{h} = \sigma\mathrm{g}$.

- If X is a product and Y a coproduct then of the sub-prenets $\mathrm{h}_{i,\varepsilon}$ and $\mathrm{h}_{\varepsilon,j}$ of h, choose one that is connected. For $\mathrm{h}_{\varepsilon,j}$, construct the sub-prenet $\mathrm{g} \subseteq \mathrm{h}_{\varepsilon,j}$ as follows. For each $u \in \textsc{maxcp}(\mathrm{h})$ choose a copointed subnet $\mathrm{q}_u \subseteq \mathrm{h}_{u,j}$, then let $\textsc{maxcp}(\mathrm{h}) = \{u_1,\dots,u_n\}$ and construct the following series of pre-nets.

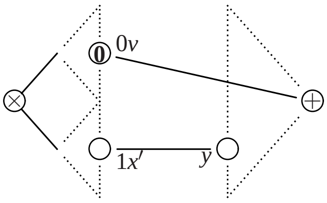
  $$\mathrm{h}_{\varepsilon,j} \ = \ \mathrm{g}_0,\ \mathrm{g}_1,\ \dots,\ \mathrm{g}_n \qquad \text{where} \qquad \mathrm{g}_i \ = \ \mathrm{g}_{i-1}\{\sigma\mathrm{q}_{u_i}\}_{u_i,\varepsilon}$$

  From g obtain a net f_j by induction; let $\mathrm{f} = \mathrm{f}_j;\iota_j$. For a sub-prenet $\mathrm{h}_{i,\varepsilon}$ the procedure is dual.

First, it will be shown that $\mathrm{h}_{i,\varepsilon}$ or $\mathrm{h}_{\varepsilon,j}$ is connected, for some i or j. If h contains a rooted link, say $\langle 0v,\varepsilon\rangle$, then $\mathrm{h}_{0,\varepsilon}$ is connected by the following argument. Let $\varsigma = (\varsigma_L,\varsigma_R)$ be a switching for $\mathrm{h}_{0,\varepsilon}$, and let τ be a switching for h that agrees with ς, as follows.

$$\tau_L(0u) \ = \ \varsigma_L(u) \qquad\qquad \tau_R \ = \ \varsigma_R$$

By connectedness τ switches on at least one link $\langle x,y\rangle$. If $x = 0x'$ then $\varsigma \text{⏻} \langle x',y\rangle$, and if $x = \varepsilon$ then h must also contain $\langle 0,y\rangle$ because it is saturated, and so $\varsigma \text{⏻} \langle \varepsilon,y\rangle$.



Otherwise, $x = 1x'$. Construct a second switching ρ for h that switches on both $\langle 0v, \varepsilon\rangle$ and $\langle 1x', y\rangle$, in the following way.

$$\rho_L(0u) \;=\; \begin{cases} 0 & \text{if } 0u0 \leq 0v \\ 1 & \text{otherwise} \end{cases} \qquad \rho_L(1u) \;=\; \tau_L(1u) \qquad \rho_R \;=\; \tau_R$$

Because h is close-knit, $\langle 0v, \varepsilon\rangle \frown_{\rho}^{*} \langle 1x', y\rangle$. Since this path of neighbouring links contains links both of the form $\langle 1u, z\rangle$ and of the form $\langle 0u, z\rangle$, it must contain a section

$$\langle 0v, \varepsilon\rangle \frown_{\rho} \ldots \frown_{\rho} \langle 1, w\rangle \frown_{\rho} \langle \varepsilon, w\rangle \frown_{\rho} \langle 0, w\rangle \frown_{\rho} \ldots \frown_{\rho} \langle 1x', y\rangle$$

for some vertex w. But since $\rho_R = \tau_R = \varsigma_R$ the vertex w is switched on by ς_R, while the vertex 0 cannot be switched off; then $\varsigma \mathrel{⏻} \langle 0, w\rangle$. Then $\mathrm{h}_{\mathbf{0},\varepsilon}$ is connected.

The above showed that one of $\mathrm{h}_{i,\varepsilon}$ and $\mathrm{h}_{\varepsilon,j}$ is connected, under the assumption that h contains a rooted link. It will now be shown that h does in fact contain a rooted link $\langle u, \varepsilon\rangle$ or $\langle \varepsilon, z\rangle$. Assume for contradiction that none of the four sub-prenets is connected. Then there exist switchings τ and ρ such that, without loss of generality, $\tau \mathrel{⏻} \langle 0v, 0w\rangle$ and $\rho \mathrel{⏻} \langle 1x, 1y\rangle$. A switching $\varsigma = (\varsigma_L, \varsigma_R)$ is constructed from τ and ρ that switches on both these links, as follows.

$$\varsigma_L(0u) \;=\; \tau_L(0u) \qquad \varsigma_L(1u) \;=\; \rho_L(1u) \qquad \varsigma_R(0z) \;=\; \tau_R(0z) \qquad \varsigma_R(1z) \;=\; \rho_R(1z)$$

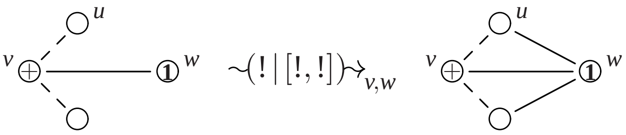
Then $\varsigma \mathrel{⏻} \langle 0v, 0w\rangle$ and $\varsigma \mathrel{⏻} \langle 1x, 1y\rangle$, and since h is close-knit, $\langle 0v, 0w\rangle \frown_{\varsigma}^{*} \langle 1x, 1y\rangle$. As before, this path of neighbouring links must contain the following segments, for some u in X and some z in Y.

$$\langle u, 0\rangle \frown_{\varsigma} \langle u, \varepsilon\rangle \frown_{\varsigma} \langle u, 1\rangle \qquad\qquad \langle 0, z\rangle \frown_{\varsigma} \langle \varepsilon, z\rangle \frown_{\varsigma} \langle 1, z\rangle$$

Then h contains two rooted links, $\langle u, \varepsilon\rangle$ and $\langle \varepsilon, z\rangle$, and by the above one of $\mathrm{h}_{0,\varepsilon}$ and $\mathrm{h}_{1,\varepsilon}$, and one of $\mathrm{h}_{\varepsilon,0}$ and $\mathrm{h}_{\varepsilon,1}$, must be connected.

Having shown that at least one $\mathrm{h}_{i,\varepsilon}$ or $\mathrm{h}_{\varepsilon,j}$ is connected, without loss of generality suppose that the algorithm selects the connected prenet $\mathrm{h}_{\varepsilon,0}$. Recall that g is then obtained from $\mathrm{h}_{\varepsilon,0}$ by replacing each sub-prenet between a vertex $u \in \text{MAXCP}(\mathrm{h})$ and ε with $\sigma(\mathrm{q}_u)$, the saturation of a copointed subnet $\mathrm{q}_u \subseteq \mathrm{h}_{u,0}$. Such a choice q_u for every $u \in \text{MAXCP}(\mathrm{h})$ exists since h is saturated: it is obtained from the maximal copointed subnet of u by a synchronised saturation step moving initial links $\langle v, \varepsilon\rangle$ down to $\langle v, 0\rangle$. Since h contains q_u and is saturated, $\sigma\mathrm{q}_u \subseteq \mathrm{h}_{u,0}$, and as g is obtained from $\mathrm{h}_{\varepsilon,0}$ by replacing $\mathrm{h}_{u,0}$ with $\sigma\mathrm{q}_u$ (for each $u \in \text{MAXCP}(\mathrm{h})$), also $\mathrm{g} \subseteq \mathrm{h}_{\varepsilon,0}$. In the following, let q_u be fixed for every $u \in \text{MAXCP}(\mathrm{h})$

To apply the desaturation algorithm recursively to g, it must be shown that g is connected, saturated, and close-knit. For the first, since $\mathrm{h}_{\varepsilon,0}$ is connected, so is g—each q_u is connected, so replacing a sub-prenet with $\sigma\mathrm{q}_u$ cannot break connectedness. For the second, $\mathrm{h}_{\varepsilon,0}$ is saturated because h is. Also, each $\sigma\mathrm{q}_u$ is saturated. It will be shown that replacing a subnet $\mathrm{h}_{u,0}$ with $\sigma\mathrm{q}_u$ does not break saturatedness. Assuming the contrary, there is a saturation step $\sim\!\!\prec(\mathrm{f}\,|\,\mathrm{k})\succ\!\!\rightarrow_{v,w}$ on g, where f is a subnet of g, but some link in k is not in g. Since $\sigma\mathrm{q}_u \subseteq \mathrm{h}_{u,0}$, this link in k must be in the latter but not the former; however, if $u \leq v$ the entire rewrite step is in $\sigma\mathrm{q}_u$, which is already saturated. This rules out the four saturation steps where v is $\mathbf{0}$. Then w must be $\mathbf{1}$, and u must be $v0$ or $v1$; without loss of generality, let u be $v0$. The two saturation steps where v is a product are ruled out: the copointed subnet q_u is maximal, while $\pi_i;\mathrm{q}_u$ would be a larger copointed subnet, for v. Of the two remaining saturation steps, only the following one adds a link that is in $\mathrm{h}_{u,0}$.



But this link, $\langle u,w\rangle$, is already in $\sigma\mathrm{q}_u$, since the latter contains all terminal links, by Lemma 4.3.6. Thus g is saturated.

Before it is shown that g is close-knit the following statement will be proved.

I If $\langle x,1y\rangle$ is a link in h then some $z \leq x$ is in $\textsc{maxcp}(\mathrm{h})$.

Since $\mathrm{h}_{\varepsilon,0}$ is connected every switching ς that switches on $\langle x,1y\rangle$ must also switch on a link $\langle v,0w\rangle$; and because h is close-knit, $\langle v,0w\rangle \frown_{\varsigma}^{*} \langle x,1y\rangle$. The path of neighbouring links connecting these links must pass through a rooted initial link, and traversing the path from $\langle v,0w\rangle$ to $\langle x,1y\rangle$, there is a last such link $\langle u,\varepsilon\rangle$. In other words, for every ς such that $\varsigma \mho \langle x,1y\rangle$ there is a link $\langle u,\varepsilon\rangle$ such that

$$\langle u,\varepsilon\rangle \frown_{\varsigma} \langle u,1\rangle \frown_{\varsigma} \langle v_1,1w_1\rangle \frown_{\varsigma} \ldots \frown_{\varsigma} \langle v_n,1w_n\rangle \frown_{\varsigma} \langle x,1y\rangle$$

Without loss of generality let $\langle x,1y\rangle$ be such that x is minimal, i.e. no $\langle x',1y'\rangle$ exists in h such that $x' < x$. Then $x \leq v_i$ for all $i \leq n$, and in particular $x \leq u$: the path cannot reach a link such that $x \not\leq v_i$ without also crossing a link $\langle v_j,w_j\rangle$ where v_j is the common root, the greatest common prefix, of x and v_i. For every ς such that $\varsigma \mho \langle x,1y\rangle$, this argument provides a rooted initial link $\langle u,\varepsilon\rangle$ with $x \leq u$ and $\varsigma \mho \langle u,\varepsilon\rangle$. Then in the subnet $\mathrm{h}_{x,\varepsilon}$, every switching switches on at least one rooted initial link. Selecting exactly one such link for each switching then provides a copointed subnet $\mathrm{q} \subseteq \mathrm{h}_{x,\varepsilon}$.

Since x has a copointed subnet, it follows that some $z \leq x$ has a maximal copointed subnet, showing **I**.

To show that g is close-knit, let ς be a switching for g and let $\varsigma \text{⏻} \langle v,w\rangle$ and $\varsigma \text{⏻} \langle x,y\rangle$. Since $\mathrm{g} \subseteq \mathrm{h}_{\varepsilon,0}$ and h is close-knit, for an arbitrary switching τ that agrees with ς where possible, i.e. $\tau \text{⏻} \langle v,0w\rangle$ whenever $\varsigma \text{⏻} \langle v,w\rangle$, there is a path of neighbouring links in h

$$\langle v,0w\rangle \;=\; \langle v_0,w_0\rangle \frown_\tau \langle v_1,w_1\rangle \frown_\tau \dots \frown_\tau \langle v_n,w_n\rangle \;=\; \langle x,0y\rangle \;.$$

It will be shown that g contains a path of neighbours $\langle v,w\rangle \frown_\varsigma^* \langle x,y\rangle$. This path will be obtained from the above path in h by replacing the stretches where $u \leq v_i$ for some $u \in \textsc{maxcp}(\mathrm{h})$. For the other links, those $\langle v_i,w_i\rangle$ where no $u \leq v_i$ is in $\textsc{maxcp}(\mathrm{h})$, **I** above gives that $0 \leq w_i$. In addition, if w_i cannot be ε, since a link $\langle v_i,\varepsilon\rangle$ must be an initial link, constituting a copointed net, and contradicting the assumption that no $u \leq v_i$ has a maximal copointed subnet. Thus, if no $u \leq v_i$ is in $\textsc{maxcp}(\mathrm{h})$ then w_i must be of the form $0w_i'$, and since the link $\langle v_i,0w_i'\rangle$ is not replaced in the substitution of some $\mathrm{h}_{u,0}$ by $\sigma\mathrm{q}_u$, the link $\langle v_i,w_i'\rangle$ is in g.

Next, consider a vertex $ui \in \textsc{maxcp}(\mathrm{h})$, and the subnet $\mathrm{h}_{ui,\varepsilon}$. For a link $\langle v,w\rangle$ where $ui \leq v$, no neighbour $\langle v,w'\rangle$ is outside $\mathrm{h}_{ui,\varepsilon}$. Moreover, a neighbour $\langle v',w\rangle$ is outside $\mathrm{h}_{ui,\varepsilon}$ only if $ui \not\leq v'$. That is, if $\langle v,w\rangle \frown_\tau \langle v',w\rangle$, while $ui \leq v$ but $ui \not\leq v'$, then it must be that $v' = u$ and $v = ui$, and that w is $\mathbf{1}$. In addition, by the above, since v' is not dominated by a vertex that has a maximal copointed subnet, $0 \leq w$. Together, these observations imply that the only neighbouring steps between a link inside $\mathrm{h}_{ui,\varepsilon}$ and a link outside it, are of the form

$$\langle ui,0w\rangle \;\frown_\tau\; \langle u,0w\rangle$$

where $0w$ is terminal. Then a segment of the path of neighbours in h that enters and exits the subnet $\mathrm{h}_{ui,\varepsilon}$ is of the following form:

$$\dots\langle u,0w_i\rangle \;\frown_\tau\; \langle ui,0w_i\rangle \;\frown_\tau\; \dots \;\frown_\tau\; \langle ui,0w_j\rangle \;\frown_\tau\; \langle u,0w_j\rangle\dots$$

It will be shown that in g, this segment can be replaced by another path of neighbours,

$$\dots\langle u,w_i\rangle \;\frown_\varsigma\; \langle ui,w_i\rangle \;\frown_\varsigma^*\; \langle ui,w_j\rangle \;\frown_\varsigma\; \langle u,w_j\rangle\dots$$

Such a path exists because of two facts. Firstly, since the links $\langle ui,w_i\rangle$ and $\langle ui,w_j\rangle$ are terminal links, the corresponding links $\langle \varepsilon,w_i\rangle$ and $\langle \varepsilon,w_j\rangle$ exist in $\sigma\mathrm{q}_{ui}$, because by Lemma 4.3.6 $\sigma\mathrm{q}_{ui}$ contains all terminal links. Secondly, since $\sigma\mathrm{q}_{ui}$ is a saturated net, by Proposition 3.4.4 it must be close-knit; then there must be a path of neighbours between $\langle \varepsilon, w_i \rangle$ and $\langle \varepsilon, w_j \rangle$ in $\sigma \mathrm{q}_{ui}$ for any switching—including the one that agrees with ς and τ. This shows the above segment in g exists. Then the path $\langle v, w \rangle \frown^*_\varsigma \langle x, y \rangle$ in g is constructed by taking the corresponding path in h while no $u \leq v_i$ has a maximal copointed subnet, and replacing the path by another by the above construction for the segments where some $u \leq v_i$ does have a maximal copointed subnet. Thus, g is close-knit.

At this point g has been shown to be connected, saturated, and close-knit, and by induction the desaturation algorithm gives a net f such that $\sigma \mathrm{f} = \mathrm{g}$. It remains to show that $\sigma(\mathrm{f}; \iota_0) = \mathrm{h}$. In one direction, $\sigma(\mathrm{f}; \iota_0) \subseteq \mathrm{h}$ is immediate since $\mathrm{f}; \iota_0 \subseteq \mathrm{g}; \iota_0 \subseteq \mathrm{h}$, and h is saturated. For the other direction, first the following statement will be proved.

II. If $\langle x, y \rangle$ is a link in h such that some $u \leq x$ is in $\textsc{maxcp}(\mathrm{h})$, but $\langle x, \varepsilon \rangle$ is not in q_u, then some $w \leq y$ is pointed.

For the link $\langle x, y \rangle$, let y be minimal in the following sense: there is no link $\langle x, y' \rangle$ in h such that $y' \leq y$. Let ς be an arbitrary switching such that ς ⏻ $\langle x, y \rangle$. Since $u \leq x$ also u is switched on, and since q_u is a copointed net, by the switching condition at least one link in q_u, an initial link $\langle v, \varepsilon \rangle$ with $u \leq v$, is switched on by ς. By assumption, $\langle x, \varepsilon \rangle$ is not in q_u, so $\langle v, \varepsilon \rangle$ is distinct from $\langle x, y \rangle$. Then since h is close-knit it contains a path of neighbours of the form

$$\langle x, y \rangle \frown_\varsigma \langle v_1, w_1 \rangle \frown_\varsigma \ldots \frown_\varsigma \langle v_n, w_n \rangle \frown_\varsigma \langle v, \varepsilon \rangle \ .$$

Since x is distinct from v, this path must contain at least one segment of terminal links $\langle v_i, w \rangle, \ldots, \langle v_j, w \rangle$, where w is $\mathbf{1}$. Assume this is the first such segment. Then the path before it must be of the form $\langle x, y \rangle, \ldots, \langle x, w \rangle$; that is, if w_i is the first terminal target vertex in the path above, $v_k = x$ for all $k \leq i$. By the assumption of minimality of y, it follows that $w \leq y$. This argument provides, for every switching ς that switches on y, a terminal node $w \leq y$. This is equivalent to y being pointed.

To show that $\sigma(\mathrm{f}; \iota_0) \supseteq \mathrm{h}$, Lemma 4.4.1 describes $\sigma(\mathrm{f}; \iota_0)$ as $\sigma \mathrm{f}; \iota_0 \cup \Gamma \cup \Delta$, where Γ are all links $\langle v, w \rangle$ where v is 0 and $\langle v, \varepsilon \rangle$ is in $\sigma \mathrm{f}$, and Δ contains all links $\langle v, w \rangle$ where some $v' \leq v$ has a maximal copointed subnet and some $w' \leq w$ is pointed. Let $\langle x, y \rangle$ be a link in h. If $u \leq x$ for some $u \in \textsc{maxcp}(\mathrm{h})$ then by **II** either the link $\langle x, \varepsilon \rangle$ is in q_u, in which case $\langle x, y \rangle$ is in Γ, or some $w \leq y$ is pointed, in which case $\langle x, y \rangle$ is in Δ. Otherwise, if $u \not\leq x$ for all $u \in \textsc{maxcp}(\mathrm{h})$, then $1 \not\leq y$ by **I**. Moreover, $y \neq \varepsilon$, because otherwise $\langle x, y \rangle$ would be an initial link constituting a copointed subnet. Then $0 \leq y$ and $\langle x, y \rangle$ is in $\mathrm{g}; \iota_0 = \sigma \mathrm{f}; \iota_0$. This concludes the proof that $\mathrm{h} = \sigma(\mathrm{f}; \iota_0)$. □

Part II

Classical proof forestry

Chapter 5

Classical proof forests

5.1 Introduction

In this part of the dissertation a canonical graphical calculus for first-order classical logic, here called *classical proof forests*, is investigated. The cut-free calculus was first described by Dale Miller [79] as *expansion tree proofs*, a compact representation of first-order and higher-order classical proof. The present approach, based on Herbrand's Theorem and a semantics of backtracking games in the style of Thierry Coquand [26] and the exponential modalities $(?, !)$ of linear logic, adds composition via cut and cut-elimination. The current chapter will discuss background material and related work and present the forests themselves. The next chapter, Chapter 6, will introduce the cut-reduction steps and give a proof of cut-elimination. Chapter 7 will discuss variations on the reduction relation, and provide a detailed discussion of related work.

Classical proof forests, as a representation of first-order classical proof, have a strict focus on witness assignment to quantifiers and dependencies between such assignments, and ignore the (decidable) propositional side of classical logic. This approach is familiar from Herbrand's Theorem, which shows that a suitable witness assignment to quantifiers of a first-order formula is sufficient to make it decidable. By allowing the dependencies between different witness assignments to form a partial order, the proof forests factor out the permutations of the sequent calculus, and are in that sense canonical. The game-theoretic semantics allows an intuitive interpretation of the forest proofs as strategies for a two-player game, and provides valuable insights in addressing several of the more technical issues encountered in this work.

An interesting challenge for such a representation of proof is to find a notion of composition via cut-elimination. Unlike in the sequent calculus, whose pervasive bureaucracy means cut-elimination is dominated by permutations and similar inessential operations, it may be expected that cut-elimination in a canonical formalism such as proof forests consists solely of conversions that are significant. In addition, the undesirable reduction behaviour of the sequent calculus is commonly attributed to cuts on two weakened formulae (the Lafont example in Figure 1.3) and cuts on two contracted formulae (see [28, Section 3]). Since proof forests rule out such cuts, because contraction and weakening are restricted to existentially quantified formulae, it may be hoped that cut-elimination is well-behaved.

This part of the dissertation describes the results of a programme investigating composition via cut-elimination for classical proof forests. A first contribution is the definition of a cut-reduction relation, naturally inspired by both the structure of the forests, their game semantics, and the interpretation of sequent proofs. Still, these reduction steps turn out to be badly behaved: certain cuts cannot be reduced, and what is worse, such badly behaved cuts can be reached by reduction from perfectly ordinary ones. The example proof forest exhibiting such bad reduction behaviour is non-trivial, and its discovery is a main contribution of this work. Two further principal contributions are the two solutions to this problem that will be presented. The first solution identifies the structure causing bad reduction behaviour as redundant, and provides a way of removing it. A modified reduction relation that includes this an operation removing the unwanted structure is shown to be weakly normalising, and conjectured to be strongly normalising. The second solution is based on an analysis of when reduction steps cause the loss of weak normalisation, and consists of a reduction strategy that avoids those steps, obtaining weak normalisation for the original reduction relation.

The present chapter will discuss the proof forests, and introduce a notion of cut. Section 5.2 will introduce the proof forests informally from a discussion of the background material; Section 5.3 will discuss composition with cut, and in Section 5.4 the forests be will defined formally. In Section 5.5, translation procedures between sequent proofs and proof forests will be discussed, and it is illustrated how proof forests factor out the bureaucracy of the sequent calculus. Different variants of cut-elimination will be treated in Chapters 6 and 7; the latter chapter will, in addition, compare proof forests to related work in more detail. The results in Chapters 5 and 6 appeared in [48], which is included as an appendix; the material in Chapter 7 is new.

5.2 Background

In this section classical proof forests will be introduced and motivated, from three points of view: one, Herbrand's Theorem; two, backtracking games; and three, the sequent calculus. Proof forests will first be treated informally, in a cut-free setting.

Herbrand's Theorem

Herbrand's Theorem [50] states that a first-order formula A is valid, if and only if it can be transformed into a propositional tautology by the combination of the following operations (applied to the formula transformed to negation-normal form).

1. *Expansion*: an occurrence of a subformula $\exists x.B$ is replaced by a disjunction of any number of copies of itself, $\exists x.B \vee \ldots \vee \exists x.B$. This may be repeated an arbitrary number of times.

2. *Prenexification*: casting the expanded formula into prenex-normal form, by moving quantifiers from inside the formula to the front (and renaming variables when necessary).

3. *Witness assignment*: the existentially quantified variables in the prenex formula, are each replaced with a first-order term. A term t substituted for a variable y in a formula $Q_1 x_1 \ldots Q_n x_n.\exists y.B$, where each Q_i is a quantifier, $\forall$ or $\exists$, may use no other bound variables than those of $x_1 \ldots x_n$ that are universally quantified. Of the resulting universally quantified formula, the matrix is taken (the propositional part).

In [20] Samuel Buss describes a calculus of *Herbrand proofs*, which consist of the above three steps, followed by a tautology check.

The expansion of the formula essentially allows an arbitrary number of choices of instantiating each existentially quantified formula. This suggest a tree-notation in which universal quantifiers have unique successors, and existential quantifiers arbitrarily many. The prenexification is a topological sort of the quantifiers in the expanded formula (it imposes a linear order that respects their original tree-ordering). This determines what universally quantified variables may be used in the witnessing terms for the existentially quantified variables. However, the same substitutions may be enabled by several different ways of turning a formula into prenex-normal form. The suggestion is then, that rather than imposing a linear order on quantifiers, a partial order may be more pertinent. These two suggestions combined, of a tree-notation with a superimposed partial order, are at the basis of Miller's expansion tree proofs [79]. Here the same ideas inspire classical proof forests, which are closely modelled on expansion tree proofs.

Backtracking games

Backtracking games were used by Coquand [26] in the early 1990s as a means of giving evidence for statements of classical arithmetic. Backtracking games can be defined in several ways: for instance, some games allow backtracking for both players; others, like the ones used here, for just one of the two. Since not much hinges on the precise choice of definition, the games will only be informally sketched.

A game is played by two players, '∀belard' (falsifier) and '∃loise' (verifier), on a chosen structure. The players take turns assigning witnesses, elements from the domain of the structure, to the quantifiers in a sequent of prenex formulae. Positions in the game are (partially) instantiated subformulae. ∃loise can revert to any previous position where it was her turn and assign a new witness; her current position is recorded and can be a target for later backtracking. She wins the game if it reaches a quantifier-free position that is true in the structure.

A proof is a strategy for ∃loise that is winning on any structure. Traditionally, strategies are functions that, given the history of a game, provide the next move. Proof forests deviate from this, abstracting away from irrelevant choices in the order of moves: moves in the strategy are only partially ordered, and given the history of a game the strategy suggests a range of possible moves. Restrictions made by proof forests are that the strategies the represent are finite, and uniform, in the sense that it is not influenced by which structure the game is played on.

Cut-free proof forests

A classical proof forest represents a strategy for ∃loise, for a game specified by a sequent of first-order formulae in prenex-normal form. A forest contains a tree for each formula in the sequent and is defined as a graph, with edges representing moves and nodes corresponding to positions. The order in which moves are played is only partially specified, by means of a partial order on nodes and edges called the *dependency*.

As an example, consider the proof of the drinker's formula[1] in Figure 5.1.

---

[1] This typical example of a classically valid formula with no constructive proof is so named after the

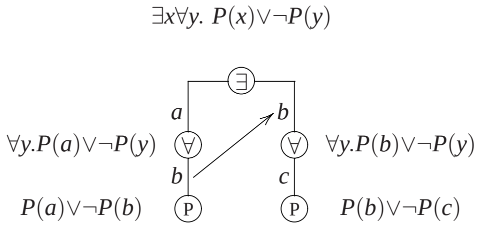


Figure 5.1: A forest proof of the drinker's formula

The root node at the top is the starting position: in the illustrations, edges point downwards. The strategy opens on the left branch, where ∃loise assigns an arbitrary value from the domain (represented by the variable a) to the existential quantifier. Next, ∀belard instantiates the universal quantifier with a certain value, recorded as b. If the position bottom left is true for these values, ∃loise wins. Otherwise, she backtracks to the root of the tree, this time taking the right branch and assigning the value b to the existential quantifier. Then, whichever value c ∀belard chooses for y, at the bottom right position $P(b) \vee \neg P(c)$ must be true, since previously in the game $P(a) \vee \neg P(b)$ was false.

The arrow in the diagram indicates where ∃loise's choice of witness relies on earlier witness assignments by ∀belard. Together with the ordering of the nodes and edges in a tree—which reflects that before the subformulae of a position can be reached the position must be reached itself—this forms the *dependency ordering*. Backtracking is represented by branching at existential positions, where the strategy does not necessarily define which branch to take first.

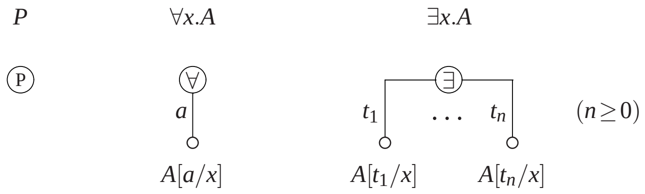


Figure 5.2: Forest components

A cut-free classical proof forest is a forest of trees built from the components in Figure 5.2, plus a dependency ordering over the combined nodes and edges. In the diagram, P and A are propositional and prenex formulae, respectively, and the smaller

---

interpretation: 'there is a man in a bar, and if anyone drinks, he drinks.' This is also the example used by Miller [79].

circles represent arbitrary nodes (that need not be leaves). From left to right are displayed a propositional position, a move by ∀belard, and several moves from the same position by ∃loise.

A dependency ordering on a proof forest will be a relation on nodes and edges subject to three conditions: 1) an edge is larger than its source node and smaller than its target, 2) an edge carrying ∀belard's choice *a* is smaller than an edge indicating ∃loise's choice *t* if *a* occurs free in *t*, and 3) it is a partial order. Since the dependency indicates a constraint on the order of play, two distinct moves depending on each other would constitute a form of *deadlock*, where each is waiting for the other; the latter condition, that the dependency must be a partial order, can thus be seen as preventing deadlock. The smallest dependency on a forest is called the *minimal* one. Later, a forest will be allowed to carry a non-minimal dependency, but for now the minimal one will be used.

A correctness condition for cut-free proof forests follows naturally from the game-theoretic interpretation. A proof forest is a proof of its sequent if it represents a winning strategy for ∃loise, regardless of the actual structure on which any particular game is played. This is precisely the case when the disjunction over all propositional nodes in the forest forms a tautology. A cut-free forest with this property will be called *correct*.

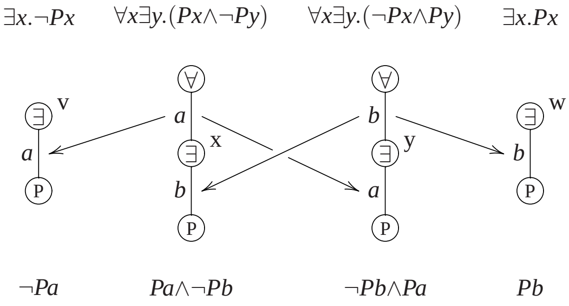


Figure 5.3: An example proof forest

A second example forest, pictured in Figure 5.3, illustrates a dependency that is not a linear order. A play starts with either of ∀belard's two moves, top center, assigning *a* or *b*—which one is not determined by the strategy. ∀belard's move *a* enables ∃loise's move at the vertex v, and ∀belard's move *b* enables her move at w. The moves at x and y depend on *both* of ∀belard's moves. As with ∀belard's moves before, the strategy does not give an order of play for the four moves by ∃loise.

The dependency, central to classical proof forests, already appears in Miller's expansion tree proofs [79], of which cut-free proof forests are the (prenex) first-order fragment. Soundness and completeness are established in that paper, and also follow from translations with the sequent calculus, described informally in the next subsection, and in more detail in Section 5.5.

A first-order sequent calculus

$$\frac{}{\vdash A_1,\ldots,A_n}\text{Taut}^{*} \qquad \frac{\vdash \Gamma, A[a/x]}{\vdash \Gamma, \forall x.A}\forall\text{R}^{**} \qquad \frac{\vdash \Gamma, A[t/x]}{\vdash \Gamma, \exists x.A}\exists\text{R}$$

$$\frac{\vdash \Gamma, \exists x.A, \exists x.A}{\vdash \Gamma, \exists x.A}\text{C}\exists \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \exists x.A}\text{W}\exists$$

$$\frac{\vdash \Gamma, A, A}{\vdash \Gamma, A}\text{CR} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, A}\text{WR} \qquad \frac{\vdash \Gamma, A \qquad \vdash A^{\perp}, \Gamma'}{\vdash \Gamma, \Gamma'}\text{Cut}$$

* $\bigvee_{i=1}^{n} A_i$ is a propositional tautology $\qquad$ ** $a \notin fv(\Gamma)$

Figure 5.4: A sequent calculus for first-order prenex formulae

Figure 5.4 displays a one-sided sequent calculus for prenex formulae. The five rules above the central line, together referred to as the *strict* calculus, are a tautology axiom, universal and existential introduction rules, and contraction and weakening on existentially quantified formulae. This calculus is called *strict* because in addition to being cut-free, it restricts contractions and weakenings to existentially quantified formulae. Due to the absence of cuts and conjunctions, proofs in the strict calculus do not exhibit any branching. The three inference rules below the central line are admissible. For the general contraction rule, this follows from the proof transformations in Figure 5.5, mentioned by Buss in [20]; the argument for general weakening is similar. Admissibility of the cut rule follows from Gentzen's sharpened Hauptsatz (also known as the midsequent theorem) [40]. As a consequence, the strict calculus of Figure 5.4 is sound and complete.

Cut-free proof forests and sequent proofs in this system can be translated back and forth straightforwardly. Here, the translation procedure will be briefly sketched; a

$$\frac{\dfrac{}{\vdash \Gamma,P,P}\,\text{Taut}}{\vdash \Gamma,P}\,\text{CR} \qquad\Rightarrow\qquad \frac{}{\vdash \Gamma,P}\,\text{Taut}$$

$$\frac{\dfrac{\dfrac{\begin{matrix}\Pi\\ \vdots\\ \vdash \Gamma,A[a/x],A[b/x]\end{matrix}}{\vdash \Gamma,A[a/x],\ \forall x.A}\,\forall\text{R}}{\vdash\Gamma,\ \forall x.A,\ \ \forall x.A}\,\forall\text{R}}{\vdash \Gamma,\ \forall x.A}\,\text{CR} \qquad\Rightarrow\qquad \frac{\dfrac{\begin{matrix}\Pi[a/b]\\ \vdots\\ \vdash\Gamma,A[a/x],A[a/x]\end{matrix}}{\vdash\Gamma,A[a/x]}\,\text{CR}}{\vdash\Gamma,\ \forall x.A}\,\forall\text{R}$$

Figure 5.5: Admissible contractions on propositional and universal formulae

complete treatment, which includes cut, can be found in Section 5.5. Edges in a forest correspond to ∀R-inferences and ∃R-inferences, branching on existential nodes to contraction, and an existential position without branches corresponds to a weakening. The dependency witnesses a non-permutable ordering of inferences. In the strict calculus this may arise, by transitivity, for two reasons: one, because one inference's conclusion is another's premise, or two, due to the *eigenvariable condition*, the side-condition on ∀R-inferences that the eigenvariable may not occur free in the context. Both are illustrated in Figure 5.6; an occurrence of these will be called an *impermutability*.

$$\frac{\dfrac{\vdash\Gamma,A}{\vdash\Gamma,B}\,\text{R1}}{\vdash\Gamma,C}\,\text{R2} \qquad\qquad \frac{\dfrac{\vdash\Gamma,A[a/x],B[t(a)/y]}{\vdash\Gamma,A[a/x],\ \ \exists y.B}\,\exists\text{R}}{\vdash\Gamma,\ \forall x.A,\ \ \ \exists y.B}\,\forall\text{R}$$

Figure 5.6: Impermutabilities

Informally, then, the dependants of a move in a forest correspond to the inferences in the smallest possible subproof of a sequent inference, in all the possible permutations of the sequent proof. To translate a forest to a sequent proof involves making contractions explicit and topologically sorting the dependency; the other direction involves the reverse.

Proof forests factor out the remaining two forms of bureaucracy of the strict calculus of Figure 5.4 (after restricting contraction and weakening to existential formulae). Firstly, proof forests use branching in place of binary contractions; although it should be noted that a similar effect can be obtained in sequent calculus as well, by having contractions of arbitrary arity and forcing these to occur immediately above the rule that has the contracted formula as a premise. Secondly, proof forests factor out the possible permutations in sequent proofs in the strict calculus, in the same way that they abstract over the choice of prenexification in Herbrand proofs, and the precise order of moves in a backtracking game: by allowing the dependency to be a partial order, where otherwise a linear order is used. For these reasons, proof forests may be considered bureaucracy-free, and in that way canonical for classical proof. A more detailed discussion will follow in Section 5.5.

5.3 Cut

A notion of cut, used to compose forests, will be introduced informally. Two game-theoretic interpretations of cuts will be discussed; one will be the main inspiration for the formal implementation of cuts, the other will provide guidance in designing the cut-reduction steps in Chapter 6. Finally, it will be shown how to decompose a forest along a cut, yielding a correctness criterion for forests with cut.

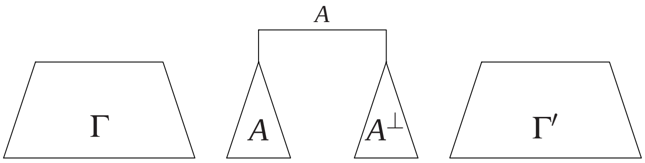


Figure 5.7: Composing forests for Γ, A and $A^\perp, \Gamma'$ with a cut

Forests for sequents Γ, A and $A^\perp, \Gamma'$ (where $A^\perp$ denotes the DeMorgan dual of A) can be composed using a *cut*, a link between the two dual trees from both forests. Figure 5.7 gives a schematic impression, where triangles and trapezoids abbreviate trees and forests respectively, and the cut is labelled with the *cut-formula*. The result is a forest for the sequent Γ, Γ', whose formulae are represented by the remaining root nodes.

A first interpretation of the cut is as a composition of strategies. The common game-theoretic interpretation of composition, among many others found in [26], is to let the two strategies play against each other on the formulae A and $A^\perp$ linked by the cut. Moves by ∃loise in one game are interpreted as moves by ∀belard in the other game, and vice versa.

This interpretation works well with strategies as functions indicating the next move, but not so well in the present setting of backtracking and partially ordered moves. In particular, if backtracking occurs in both the strategy in A and that in $A^{\perp}$, it is not obvious that when they play against each other, the game terminates. Coquand's argument in [26] uses the linear ordering of moves available in that setting; but a notion of cut that depends on a given linear order on a forest is not canonical. In addition, the interpretation of a cut as an interaction between strategies is closer to a description of *cut-elimination* than a description of cut itself. For these reasons the above interpretation will guide the design of the cut-reduction steps in Chapter 6, while the formal definition of a cut will be guided by a different, complementary interpretation in terms of moves in a game.

In this second interpretation a cut consists of two successive moves: firstly, ∃loise chooses a cut-formula A, introducing the position $A \wedge A^{\perp}$; next, ∀belard chooses one branch of this conjunction. To represent the first move by an edge in a forest, it will be modelled as a move instantiating the generic contradiction $\perp$ with a specific one $A \wedge A^{\perp}$. The idea that $\perp$ is a position available to ∃loise at all times is natural from the view that it is the empty sequent, and the unit of disjunction (as embodied by the commas of a sequent). The combined construction is displayed in Figure 5.8; the simple bar on the left will be used as an abbreviation.

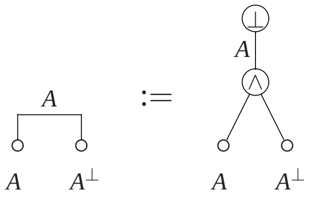


Figure 5.8: Cuts

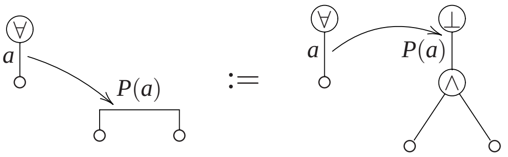
The translation of a cut in the sequent calculus is by composing, with a cut, the forest translations of the two subproofs of the cut in the sequent proof. For example, after translating the subproofs Π and Π' below to forests for Γ, A and $A^{\perp}, \Gamma'$, the translation of the whole, including the cut, will be as in Figure 5.7.

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma, A\end{array} \qquad \begin{array}{c}\Pi'\\ \vdots\\ \vdash A^{\perp}, \Gamma'\end{array}}{\vdash \Gamma, \Gamma'}\,\text{Cut}$$

In a sequent proof, a cut-formula may contain occurrences of the eigenvariables of

$\forall\mathcal{R}$-inferences, as illustrated below left.

$$\dfrac{\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma, P(a)\end{array} \qquad \begin{array}{c}\Pi'\\ \vdots\\ \vdash \neg P(a), \Gamma', B[a/x]\end{array}}{\vdash \Gamma, \Gamma', B[a/x]}\text{Cut}}{\vdash \Gamma, \Gamma', \forall x.B}\forall\text{R}$$

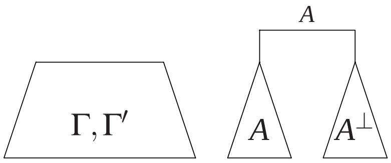


The fact that this constitutes an impermutability means that in proof forests, cuts must be part of the dependency, as illustrated above right: if a cut-formula contains an occurrence of an eigenvariable introduced in a move by ∀belard, then that cut must depend on ∀belard's move.

This interpretation of cuts has several conceptual advantages. By describing a cut as two consecutive moves in the game, it gives an interpretation that is internal to the game. Moreover, it accounts for the fact that the dependency ranges over cuts in a natural way, by describing the introduction of a cut-formula as a move by ∃loise, that may depend on previous choices by ∀belard.

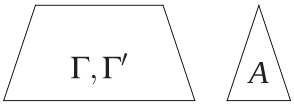
Correctness and decomposition

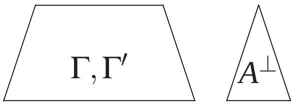
Two more, closely related, issues will be taken up here. Firstly, proof forests with cut will need a correctness criterion. Secondly, for sequentialisation (a translation back to the sequent calculus), it must be possible to de-compose a proof forest along a cut; i.e. from a proof forest with a cut on trees for A and $A^{\perp}$, it must be possible create two forests, one with the tree for A, and one with the tree for $A^{\perp}$. As suggested by the illustration below, after proof forests for Γ, A and for $A^{\perp}, \Gamma$ have been composed with a cut, it is not generally possible, in the composed forest, to determine which trees and branches in the combined Γ and Γ' used to belong to which original forest.



An idea towards solutions to both issues is provided by the interpretation of the cut as two consecutive moves in a game. The second of these moves, the conjunction, is a choice by ∀belard for either branch. In any given play, the positions (nodes) in the branch *not* chosen by ∀belard will never be played—and neither will those depending on them. Since a proof forest, representing a winning strategy for ∃loise, should offer a counter-strategy to *any* possible move by ∀belard, this suggests the following treatment of cuts: for the two trees linked by a cut, removing either one plus all of its dependants should leave a proof forest that is a winning strategy.

To decompose a proof forest along a cut on trees for A and $A^{\perp}$, one forest is obtained by removing the tree for $A^{\perp}$, and all its dependants, the other by removing that for A, plus all dependants. The proof forest illustrated above, for Γ, Γ' and with a cut on A and $A^{\perp}$, is thus decomposed into the following two forests (assuming no dependencies between the trees for A, $A^{\perp}$ and Γ, Γ').





The resulting proof forests above are for Γ, Γ', A and $\Gamma, \Gamma', A^{\perp}$. Using decomposition to translate a cut in a proof forest to one in sequent calculus thus gives the cut in its *additive* formulation, below—as opposed to the *multiplicative* formulation in Figure 5.4.

$$\dfrac{\vdash \Gamma, A \qquad \vdash \Gamma, A^{\perp}}{\vdash \Gamma}\,\text{Add.Cut}$$

The additive and multiplicative formulations are equivalent classical logic, due to the presence of contraction and weakening. This will be used in Section 5.5 to provide a translation from proof forests with cuts to sequent proofs in the calculus of Figure 5.4. Another, more concrete example of decomposition is pictured in Figure 5.9.

The correctness criterion that will be introduced for proof forests is closely related to decomposition. Let a *switching* be a choice for one branch of every cut—intuitively, a strategy for ∀belard on conjunctions. Then for every switching, after removing for every cut the branch *not* indicated by the switching, plus all its dependants, the disjunction over the remaining propositional positions must form a tautology. This will be formalised in Section 5.4.

The most important aspect of the correctness criterion is that it should be preserved by the following operations:

- *composition*: the composition of two correct proof forests must be correct;
- *decomposition*: the two proof forests resulting from the decomposition of a correct proof forest must be correct;
- *cut-elimination*: cut-reduction steps, to be defined in Chapter 6, when applied to a correct proof forest must again yield a correct proof forest.

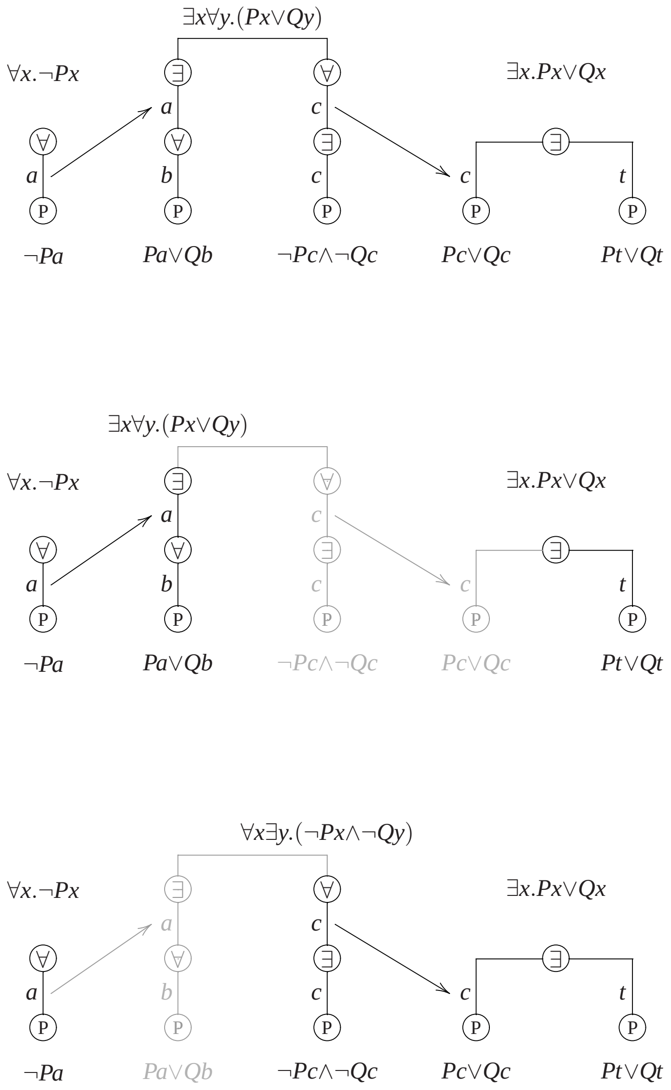


Figure 5.9: Decomposing a proof forest

The first two of these requirements are treated in Section 5.5, the first by Proposition 5.5.1, the second by Lemma 5.5.3. The third will be addressed in the next chapter, after defining the reduction steps.

A brief discussion of the game-theoretic interpretation of the cut will conclude the present section. There are clear conceptual advantages to viewing a cut as a combination of two moves in a game, which follow from having an interpretation of the cut that is internal to the game semantics:

- the interpretation of the cut is independent of cut-elimination;
- it naturally accounts for the participation of cuts in the dependency; and
- it provides a natural correctness condition for proof forests with cuts.

On the technical side, a choice has to be made to use one or the other implementation; though nothing hinges on the exact choice, except convenience. However, in that respect it is not clear-cut whether it is better to implement a cut as a link between two trees, as in the abbreviated notation, or as a combination of a $\perp$-vertex and a $\wedge$-vertex. The former has the disadvantage that a cut is an undirected edge, where the other edges in a forest are directed; the latter has the problem of introducing two additional kinds of vertex. The choice was made in favour of the latter implementation.

5.4 Classical proof forests

In this section proof forests and their translation from sequent calculus will be formalised. The definition of proof forests will closely mirror the diagrams; in particular, the arrows drawn to relate dependent moves will be implemented as an explicit dependency relation $(\rightarrow)$ on edges, from which the dependency ordering $(\leq)$ will then be generated. This will provide a better basis for reduction steps than directly defining the dependency as a partial order.

First, the language of first-order classical logic, over an arbitrary but fixed signature Σ, will be formalised. Let $\textsc{var}$ be a (countably infinite) set of variables and let the signature Σ consist of a collection of function symbols f, each of a given arity n, and a (distinct) collection of proposition symbols P of a given arity n. The first-order language then consists of the following fragments.

- A collection of *terms* $\textsc{terms}$ defined by the grammar

  $$t \ := \ x \in \textsc{var} \ \mid \ f(t_1, \ldots, t_n)$$

- A collection of *atomic formulae* ATOMS defined by the grammar

$$X \ := \ P(t_1,\ldots,t_n)$$

- A collection of *formulae* FORM defined by the grammar

$$F \ := X \ \mid \neg X \ \mid \bot \mid F \vee F \ \mid F \wedge F \ \mid \forall x.F \ \mid \exists x.F$$

For convenience also the fragments of propositional and prenex formulae, included in FORM, will be identified.

- The fragment of *propositional formulae* is defined by

$$P \ := \ X \ \mid \ \neg X \ \mid \ \bot \ \mid \ P \vee P \ \mid \ P \wedge P$$

- The fragment of *prenex formulae* is defined by

$$A \ := \ P \ \mid \ \forall x.A \ \mid \ \exists x.A$$

In this definition, negation is restricted to atomic propositions. Generalised negation is implemented using DeMorgan duality, by the meta-operator $(-)^{\bot}$.

$$\begin{array}{rclcrclcrcl} X^{\bot} & \triangleq & \neg X & \quad & (F \vee G)^{\bot} & \triangleq & F^{\bot} \wedge G^{\bot} & \quad & (\exists x.F)^{\bot} & \triangleq & \forall x.F^{\bot} \\ (\neg X)^{\bot} & \triangleq & X & & (F \wedge G)^{\bot} & \triangleq & F^{\bot} \vee G^{\bot} & & (\forall x.F)^{\bot} & \triangleq & \exists x.F^{\bot} \end{array}$$

In addition, there are reserved characters L and R, used to indicate the left and right branch of a conjunction.

Definition 5.4.1 (Pre-proof forests)**.** A *pre-proof forest* F is a tuple

$$(\mathrm{V}, \bot, lab, \mathrm{E}, \rightarrow)$$

consisting of a finite set of vertices V with a distinguished element $\bot$, a labelling function $lab : \mathrm{V} \rightarrow$ FORM assigning first-order formulae to vertices, a set of labelled edges

$$\mathrm{E} \quad \subseteq \quad \mathrm{V} \ \times \ \big(\textsc{terms} \cup \textsc{form} \cup \{\mathrm{L},\mathrm{R}\}\big) \ \times \ \mathrm{V}\ ,$$

and a *dependency relation* $(\rightarrow) \subseteq \mathrm{E} \times \mathrm{E}$; with the edges forming a forest of trees:

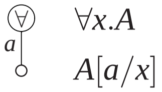
$$\begin{array}{lcll} \langle \mathrm{v}_1, l_1, \mathrm{w}\rangle, \langle \mathrm{v}_2, l_2, \mathrm{w}\rangle \in \mathrm{E} & \Rightarrow & \mathrm{v}_1 = \mathrm{v}_2,\ l_1 = l_2 & \text{(parents are unique)} \\ \langle \mathrm{v}_1, l_1, \mathrm{v}_2\rangle, \ldots, \langle \mathrm{v}_n, l_n, \mathrm{v}_{n+1}\rangle \in \mathrm{E}\ \ (n \geq 1) & \Rightarrow & \mathrm{v}_1 \neq \mathrm{v}_{n+1} & \text{(acyclicity).} \end{array}$$

The variable letters $\mathrm{u}, \mathrm{v}, \ldots, \mathrm{z}$ range over vertices, while e is used for edges. An edge $\langle \mathrm{v}, l, \mathrm{w} \rangle$ may be rendered $\langle \mathrm{v}, \mathrm{w} \rangle$ when its label l is understood or irrelevant. Standard notions used are as follows: *root* nodes are those not the target of any edge; the *edges* of a vertex are those of which it is the source; *leaves* are vertices without edges; and the *children* of a node are the targets of its edges.

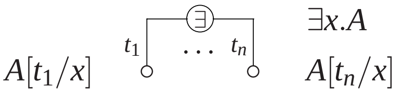
To ensure that nodes and edges a proof forest are well-configured, five types of vertex are defined below, forming four disjoint subsets of $\mathrm{V} - \{\bot\}$ in a given forest: $\mathrm{V}(\forall)$, $\mathrm{V}(\exists)$, $\mathrm{V}(\textsc{p})$, and $\mathrm{V}(\wedge)$. Nodes in these subsets are said to be in a *legal configuration*; in a proof forest all vertices will be required to be such. For consistency $\mathrm{V}(\bot)$ will denote the set $\{\bot\}$.

- A *propositional* vertex $\mathrm{v} \in \mathrm{V}(\textsc{p})$ is one that is a leaf, and is labelled with a propositional formula, $lab(\mathrm{v}) \in \textsc{prop}$.

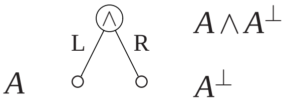
- A *universal* vertex $\mathrm{v} \in \mathrm{V}(\forall)$ is one that is labelled with a universally quantified prenex formula, $lab(\mathrm{v}) = \forall x.A \in \textsc{prenex}$, and has exactly one edge $\langle \mathrm{v}, a, \mathrm{w} \rangle$, labelled with a variable $a \in \textsc{var}$ and with a target labelled $lab(w) = A[a/x]$.



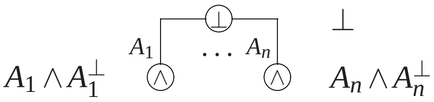
- An *existential* vertex $\mathrm{v} \in \mathrm{V}(\exists)$ is one that is labelled with an existentially quantified prenex formula $lab(\mathrm{v}) = \exists x.A \in \textsc{prenex}$, and that has any number of edges $\langle \mathrm{v}, t, \mathrm{w} \rangle$ such that the label t is a term $t \in \textsc{terms}$ and the target w labelled $lab(w) = A[t/x]$.



- A *cut* vertex $\mathrm{v} \in \mathrm{V}(\wedge)$ is one that is the target of an edge $\langle \bot, - \rangle v$, is labelled $lab(\mathrm{v}) = A \wedge A^{\perp}$ where $A \in \textsc{prenex}$ is a prenex formula, and has precisely two edges, one $\langle \mathrm{v}, \mathrm{L}, \mathrm{u} \rangle$ with $lab(\mathrm{u}) = A$ and one $\langle \mathrm{v}, \mathrm{R}, \mathrm{w} \rangle$ with $lab(w) = A^{\perp}$.



- The special vertex $\bot$ is in a legal configuration if it is labelled $lab(\bot) = \bot$, and each of its arbitrarily many edges $\langle \bot, A, \mathrm{v} \rangle$ is labelled with a prenex formula $A \in \textsc{prenex}$ and has a target cut vertex $\mathrm{v} \in \mathrm{V}(\wedge)$ labelled $lab(\mathrm{v}) = A \wedge A^{\bot}$.



Four types of edge are derived from the type of their source node: an edge $\mathrm{e} = \langle \mathrm{v}, \mathrm{w} \rangle$ is a *universal* edge $\mathrm{e} \in \mathrm{E}(\forall)$ if v is a universal vertex $\mathrm{v} \in \mathrm{V}(\forall)$; it is an *existential* edge $\mathrm{e} \in \mathrm{E}(\exists)$ if $\mathrm{v} \in \mathrm{V}(\exists)$; it is a *conjunction* edge $\mathrm{e} \in \mathrm{E}(\wedge)$ if $\mathrm{v} \in \mathrm{V}(\wedge)$; and it is a *cut* edge $\mathrm{e} \in \mathrm{E}(\bot)$ if $\mathrm{v} = \bot \in \mathrm{V}(\bot)$ (note that there are no propositional edges). The name *cut* will refer to both cut edges ($\mathrm{E}(\bot)$) and cut vertices ($\mathrm{V}(\wedge)$).

To define proof forests, only the notion of a dependency must still be formalised.

Definition 5.4.2 (Dependency)**.** The *dependency ordering* $\leq$ on a pre-proof forest is the smallest preorder on nodes and edges ($\mathrm{V} \cup \mathrm{E}$) such that

$$(\rightarrow) \ \subseteq \ (\leq) \qquad \text{and} \qquad \mathrm{v} \leq \langle \mathrm{v}, \mathrm{w} \rangle \leq \mathrm{w} \ .$$

The choice to have the dependency range over both edges and vertices was made for technical convenience.

Definition 5.4.3 (Proof forests)**.** A pre-proof forest

$$\mathrm{F} = (\mathrm{V}, \bot, lab, \mathrm{E}, \rightarrow)$$

is a *classical proof forest* for a sequent Γ of prenex, first-order formulae if

1. all nodes in V are in legal configurations,
2. Γ is equal to the multiset of the labels of root nodes in $\mathrm{V} - \{\bot\}$;
3. for a universal edge $\langle \mathrm{v}, a, \mathrm{w} \rangle \in \mathrm{E}(\forall)$ the following conditions hold:
    - a is not free in any formula in Γ,
    - $a \neq b$ for any other universal edge $\langle \mathrm{x}, b, \mathrm{y} \rangle \in \mathrm{E}(\forall)$,
    - $\langle \mathrm{v}, a, \mathrm{w} \rangle \rightarrow \langle \mathrm{x}, l, \mathrm{y} \rangle$ if $\langle \mathrm{x}, \mathrm{y} \rangle \in \mathrm{E}(\exists) \cup \mathrm{E}(\bot)$ and $a \in fv(l)$;
4. if $\mathrm{e}_1 \rightarrow \mathrm{e}_2$ then $\mathrm{e}_1 \in \mathrm{E}(\forall)$ and $\mathrm{e}_2 \in \mathrm{E}(\exists) \cup \mathrm{E}(\bot)$; and
5. the dependency $(\leq)$ is a partial order (it is antisymmetric).

Condition 3 in the above definition governs the *eigenvariables* representing the choices made by $\forall$belard. Since $\forall$belard's moves are independent of each other, in the sense that he may assign different values for each, eigenvariables are required to be unique. An existential edge or cut edge whose witnessing term or cut-formula contains an occurrence of an eigenvariable a represents a move by $\exists$loise responding to the move where $\forall$belard chooses a; then $\exists$loise's move must depend on $\forall$belard's.

A dependency over a forest can be computed using the occurrence of eigenvariables alone—this will be called the *minimal* dependency. The use of the explicit relation $(\to)$ is a natural generalisation to allow larger dependencies, along the idea that a dependency represents $\exists$loise responding to $\forall$belard's moves. To enforce this natural property, Condition 4 of Definition 5.4.3 requires that non-minimal dependencies respect the pattern that $(\to)$ relates universal edges to existential edges and cut edges. The minimal dependency on a proof forest, denoted $\leq_{\mathrm{M}}$, is imposed by replacing $(\to)$ with $(\to_{\mathrm{M}})$, defined as follows:

$$\langle \mathrm{v}, a, \mathrm{w}\rangle \to \langle \mathrm{x}, l, \mathrm{y}\rangle \quad \overset{\Delta}{\Longleftrightarrow} \quad \langle \mathrm{v}, \mathrm{w}\rangle \in \mathrm{E}(\forall) \ \wedge \ \langle \mathrm{x}, \mathrm{y}\rangle \in \mathrm{E}(\exists) \cup \mathrm{E}(\bot) \ \wedge \ a \in fv(l) \ ,$$

It is easily observed from the definitions that the dependency $\leq_{\mathrm{M}}$ is indeed minimal, in the sense that given a forest F with an arbitrary relation $(\to)$, for all $\mathrm{v}, \mathrm{w} \in \mathrm{V}$

$$\mathrm{v} \leq_{\mathrm{M}} \mathrm{w} \quad \Rightarrow \quad \mathrm{v} \leq \mathrm{w} \ .$$

Let the *minimisation* of a proof forest F be the proof forest $\mathrm{F}_{\mathrm{M}} = (\mathrm{V}, \bot, lab, \mathrm{E}, \to_{\mathrm{M}})$.

Correctness

Next, the correctness condition for proof forests will be defined. First, a switching is a function a choice for one of the two branches of each cut node.

Definition 5.4.4 (Switching)**.** A *switching* ς in a forest F is a function $\varsigma : \mathrm{V}(\wedge) \to \{\mathrm{L}, \mathrm{R}\}$, indicating a set $\mathrm{E}^{\varsigma} \subseteq \mathrm{E}(\wedge)$ that contains one branch of each conjunction:

$$\mathrm{E}^{\varsigma} \ = \ \{\langle \mathrm{v}, l, \mathrm{w}\rangle \in \mathrm{E}(\wedge) \mid \varsigma(\mathrm{v}) \neq l\} \ .$$

A vertex v is *switched off* by a switching ς if $\mathrm{e} \leq \mathrm{v}$ for some $\mathrm{e} \in \mathrm{E}^{\varsigma}$, and *switched on* otherwise.

The edges E^{ς} are the branches $\forall$belard does *not* choose; their dependent positions become unreachable in the game, and are ignored in the *value* of the forest, the disjunction over the remaining propositional nodes.

Definition 5.4.5. The *value* $val(\mathrm{F},\varsigma)$ of a proof forest F under a switching ς is the disjunction over the propositional nodes in F that are not switched off by ς:

$$val(\mathrm{F},\varsigma) \;=\; \bigvee\{lab(\mathrm{v}) \mid \mathrm{v}\in \mathrm{V}(\mathrm{P}) \;\wedge\; \forall \mathrm{e}\in \mathrm{E}^{\varsigma}.\ \mathrm{e} \not\lesssim \mathrm{v}\}\ .$$

Correctness is then defined as follows.

Definition 5.4.6 (Correctness)**.** A proof forest F is *correct* if for any switching ς the value $val(\mathrm{F},\varsigma)$ is a tautology.

A first convenient property is that correctness is preserved under minimisation.

Proposition 5.4.7. *If* F *is a correct proof forest, so is* $\mathrm{F_M}$.

Proof. Any switching ς for $\mathrm{F_M}$ is one for F, and if ς switches on a vertex v in F, it switches on v in $\mathrm{F_M}$. Then if $val(\mathrm{F},\varsigma)$ is a tautology, so is $val(\mathrm{F_M},\varsigma)$; it follows that $\mathrm{F_M}$ is correct if F is. ☐

Operations on proof forests

Finally, two natural operations on forests will be defined: *substitution* will be introduced as a means of manipulating vertices and edges, and a notion of *subforests*, as a suitable kind of subgraph of a forest, will be given. These will prove useful in the definitions of translation with sequent proofs, in Section 5.5, and in the definition of the reduction steps.

The standard *substitution* operation, as used on formulae and terms, will be applied as a natural way of renaming nodes and variables throughout a forest. On a forest F, let the substitution $[\beta/\alpha]$, where α and β are either both variables, both vertices, or both edges, be defined as follows.

- $\alpha[\beta/\alpha] \;=\; \beta$: if the substitution encounters the variable, vertex, or edge α it replaces it with β; otherwise,
- $S[\beta/\alpha] \;=\; \{X[\beta/\alpha] \mid X \in S\}$ (S is a set): if the substitution encounters a set, such as V, *lab*, E, or $(\rightarrow)$, it is applied to all its elements; otherwise,
- $(X_1,\dots,X_n)[\beta/\alpha] \;=\; (X_1[\beta/\alpha],\dots,X_n[\beta/\alpha])$: if the substitution encounters a tuple, such as a pre-proof forest $(\mathrm{V},\perp,lab,\mathrm{E},\rightarrow)$, a pair (e,e') in $(\rightarrow)$, or an edge $\langle \mathrm{v},l,\mathrm{w}\rangle$ while α and β are variables or vertices, it is applied pointwise; otherwise,

- $\gamma[\beta/\alpha] = \gamma$: if the substitution encounters anything else, such as a vertex, variable, or edge other than α, or a formula when α is a vertex or edge, it stops.

For example, this allows a substitution of one eigenvariable for another, say $[b/a]$, to be applied easily throughout a (pre-)proof forest. A second example, it provides an easy notation for merging two vertices v and w in a forest F, by simply applying the substitution $\mathrm{F}[\mathrm{v}/\mathrm{w}]$—or symmetrically by $\mathrm{F}[\mathrm{w}/\mathrm{v}]$, or by merging both with a fresh vertex x, as in $\mathrm{F}[\mathrm{x}/\mathrm{v}][\mathrm{x}/\mathrm{w}]$.

To obtain a reasonable notion of *subforest* the general graph-theoretical notion of *induced subgraph*, which is the largest subgraph over a subset of vertices, is extended to forests. Let $f|_X$ denote the restriction of the function $f : Y \to Z$ to the subdomain $X \subseteq Y$, and let $R|_X$ be the relation $R \subseteq Y \times Y$ confined to $X \times X$ (where $X \subseteq Y$). Define:

$$\mathrm{F}|_X = (X \cup \bot,\ \bot,\ lab|_X,\ \mathrm{E}|_X,\ \to|_{(\mathrm{E}|_X)})\ ,$$

where $X \subseteq \mathrm{V}$. In this characterisation $\mathrm{F}|_X$ is the largest subgraph of F over the domain $X \cup \{\bot\} \subseteq \mathrm{V}$; clearly, the axioms of pre-proof forests are preserved under this operation. If $\mathrm{F}|_X$ is a proof forest, it is called a *subforest* of F. In particular, a subforest contains the children of any universal and conjunction vertex it contains, which are the vertices with a fixed number of edges—conceptually, such a set X may be seen as closed under ∀belard's moves. In addition, it must respect that eigenvariables do not occur free at root nodes, part of condition 3 of Definition 5.4.3. For example, if X is closed under dependency then $\mathrm{F}|_X$ is a subforest, and if X is $\{\mathrm{v} \mid \mathrm{x} \not\leq \mathrm{v}\}$ with x a cut node or existential node, then, too, $\mathrm{F}|_X$ is a subforest. The subforest $\mathrm{F}|_X$ where $X = \{\mathrm{v} \mid \alpha \leq \mathrm{v}\}$ for some vertex or edge α is the *dependent subforest* of α.

5.5 Proof forests and the sequent calculus

In this section the translation between proof forests and sequent proofs, in both directions, will be discussed. The first direction to be formalised is the translation from sequent proofs, in the calculus of Figure 5.4 plus cut, to proof forests. A sequent proof Π, whose eigenvariables are assumed to be distinct, *translates* to a proof forest F, written $[\![\Pi]\!] = \mathrm{F}$, as follows.

- An instance of the tautology axiom,

$$\frac{}{\vdash P_1, \ldots, P_n}\,\mathrm{Taut}$$

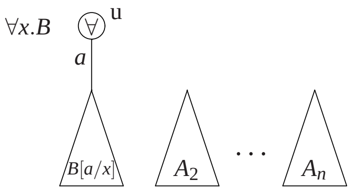
translates to a proof forest F consisting solely of propositional vertices, with $\mathrm{V} = \{\mathrm{v}_1, \ldots, \mathrm{v}_n, \bot\}$, with $lab(\mathrm{v}_i) = P_i$, with $\mathrm{E} = \varnothing$, and with $(\rightarrow) = \varnothing$.

For the remainder, let the sequent proof Π with conclusion sequent $A_1, \ldots, A_n$ translate to a proof forest $\mathrm{F_A}$ with root vertices $\{\mathrm{v}_1, \ldots, \mathrm{v}_n, \bot\}$, labelled $lab(\mathrm{v}_i) = A_i$. In all cases below, for the resulting proof forest $\mathrm{F_B}$ the dependency $\rightarrow_{\mathrm{B}}$ is chosen to be the minimal one $(\rightarrow_{\mathrm{M}})$. It should be noted that another natural choice would be to take the maximal possible dependency consistent with the ordering of the inferences in the sequent proof.

- The proof Π followed by an application of the $\forall$-right rule to $A_1 = B[a/x]$ translates to a proof forest $\mathrm{F_B}$, as follows.

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash B[a/x], A_2, \ldots, A_n\end{array}}{\vdash\ \forall x.B,\ A_2, \ldots, A_n}\,\forall\mathrm{R} \qquad \begin{array}{lcl} \mathrm{V_B} & = & \mathrm{V_A} \cup \{\mathrm{u}\} \quad (\mathrm{u} \notin \mathrm{V_A}) \\ lab_{\mathrm{B}} & = & lab_{\mathrm{A}} \cup \{\mathrm{u} \mapsto \forall x.B\} \\ \mathrm{E_B} & = & \mathrm{E_A} \cup \{\langle \mathrm{u}, a, \mathrm{v}_1 \rangle\} \end{array}$$

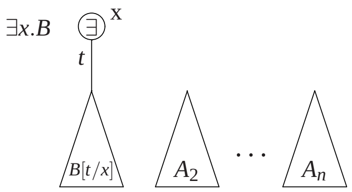
The proof forest $\mathrm{F_B}$ is illustrated below.



- The proof Π followed by an application of the $\exists$-right rule to $A_1 = B[t/x]$ translates to a proof forest $\mathrm{F_B}$ as follows. (It is assumed that a suitable term t is provided by the sequent proof also when x is not free in B.)

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash B[t/x], A_2, \ldots, A_n\end{array}}{\vdash\ \exists x.B,\ A_2, \ldots, A_n}\,\exists\mathrm{R} \qquad \begin{array}{lcl} \mathrm{V_B} & = & \mathrm{V_A} \cup \{\mathrm{x}\} \quad (\mathrm{x} \notin \mathrm{V_A}) \\ lab_{\mathrm{B}} & = & lab_{\mathrm{A}} \cup \{\mathrm{x} \mapsto \exists x.B\} \\ \mathrm{E_B} & = & \mathrm{E_A} \cup \{\langle \mathrm{x}, t, \mathrm{y}_1 \rangle\} \end{array}$$

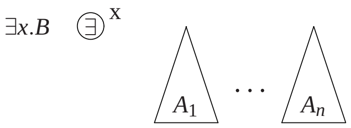
The proof forest $\mathrm{F_B}$ is illustrated below.



- The proof Π followed by an application of the $\exists$-weakening rule translates to a proof forest $\mathrm{F_B}$, as follows.

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \quad A_1,\dots,A_n\end{array}}{\vdash \exists x.B, A_1,\dots,A_n}\,\mathrm{W}\exists \qquad \begin{array}{lcl}\mathrm{V_B} & = & \mathrm{V_A} \cup \{\mathrm{x}\} \quad (\mathrm{x} \notin \mathrm{V_A})\\ lab_\mathrm{B} & = & lab_\mathrm{A} \cup \{\mathrm{x} \mapsto \exists x.B\}\\ \mathrm{E_B} & = & \mathrm{E_A}\end{array}$$

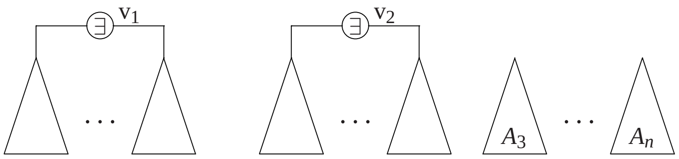
The proof forest $\mathrm{F_B}$ is illustrated below.

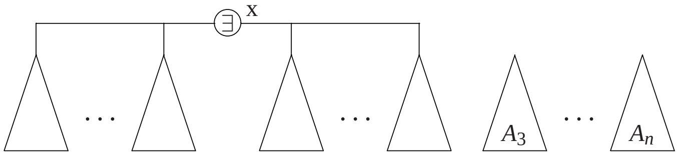


- The proof Π followed by an application of the ∃-contraction rule to A_1 and A_2 translates to a proof forest $\mathrm{F_B}$, as follows.

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \exists x.B, \exists x.B, A_3,\dots,A_n\end{array}}{\vdash \quad \exists x.B, \quad A_3,\dots,A_n}\,\mathrm{C}\exists \qquad \begin{array}{lcl}\mathrm{V_B} & = & \mathrm{V_A} - \{\mathrm{v_1},\mathrm{v_2}\} \cup \{\mathrm{x}\} \quad (\mathrm{x} \notin \mathrm{V_A})\\ lab_\mathrm{B} & = & lab_\mathrm{A}[\mathrm{x}/\mathrm{v_1},\mathrm{x}/\mathrm{v_2}]\\ \mathrm{E_B} & = & \mathrm{E_A}[\mathrm{x}/\mathrm{v_1},\mathrm{x}/\mathrm{v_2}]\end{array}$$

The first picture below illustrates $\mathrm{F_A}$, the second $\mathrm{F_B}$.





For the translation of the cut-rule, let the sequent proof Π translate to $\mathrm{F_A}$ and Π' to $\mathrm{F_B}$, where Π has conclusions $A_1,\dots,A_i,B$ and Π' conclusions $B^\perp,A_{k+1},\dots,A_n$. Assume that the proof forests $\mathrm{F_A}$ and $\mathrm{F_B}$ have no vertices in common, except, conveniently, the ⊥-node: $\mathrm{V_A} \cap \mathrm{V_B} = \{\perp\}$. Apart from ⊥, let the root nodes of $\mathrm{F_A}$ be $\mathrm{v}_1,\dots,\mathrm{v}_k,\mathrm{x}$ with $lab_\mathrm{A}(\mathrm{v}_i) = A_i$ and $lab_\mathrm{A}(\mathrm{x}) = B$, and let those of $\mathrm{F_B}$ be $\mathrm{y},\mathrm{v}_{k+1},\dots,\mathrm{v}_n$ with $lab_\mathrm{B}(\mathrm{y}) = B^\perp$ and $lab_\mathrm{B}(\mathrm{v}_i) = A_i$.

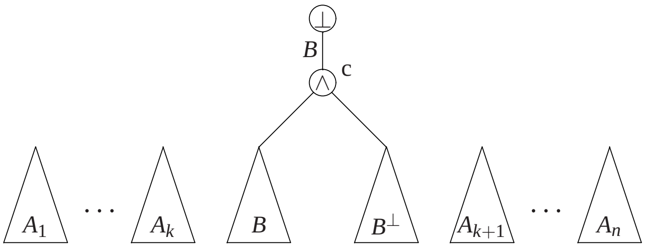
- The combination of the proofs Π and Π' by a cut on B and $B^\perp$,

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash A_1,\dots,A_k,B\end{array} \qquad \begin{array}{c}\Pi'\\ \vdots\\ \vdash B^\perp,A_{k+1},\dots,A_n\end{array}}{\vdash A_1,\dots,A_n}\,\mathrm{Cut}$$

translates to a proof forest $\mathrm{F_C}$ as follows.

$$\begin{array}{lcl}\mathrm{V_C} & = & \mathrm{V_A} \cup \mathrm{V_B} \cup \{\mathrm{c}\} \qquad (\mathrm{c} \notin \mathrm{V_A} \cup \mathrm{V_B})\\ lab_\mathrm{C} & = & lab_\mathrm{A} \cup lab_\mathrm{B} \cup \{\mathrm{c} \mapsto (B \wedge B^\perp)\}\\ \mathrm{E_C} & = & \mathrm{E_A} \cup \mathrm{E_B} \cup \{\langle \perp, B, \mathrm{c}\rangle, \langle \mathrm{c},\mathrm{L},\mathrm{u}\rangle, \langle \mathrm{c},\mathrm{R},\mathrm{w}\rangle\}\end{array}$$

The proof forest $\mathrm{F_C}$ is illustrated below.



Proposition 5.5.1. *The translation* $[\![\Pi]\!]$ *of a sequent proof* Π *with conclusion* Γ *is a correct proof forest for* Γ*.*

Proof. It is immediate from the translation that $[\![\Pi]\!]$ is a pre-proof forest satisfying conditions 1 (all vertices are in legal configurations) and 2 (the labels of root nodes form Γ). Conditions 3 and 5 follow from the eigenvariable condition on $\forall$R-inferences, which enforces that below a $\forall$R-inference with eigenvariable a no formula A contains a freely. Then $a \notin fv(\Gamma)$, and any edge added in a translation step is always minimal in the dependency: in the case of an existential edge $\langle \mathrm{u}, t, \mathrm{v}\rangle$ because $a \notin fv(t)$, in the case of a cut edge $\langle \bot, B, \mathrm{c}\rangle$ because $a \notin fv(B)$, for any eigenvariable a in Π. Condition 4 $((\rightarrow) \subseteq \mathrm{E}(\forall) \times (\mathrm{E}(\exists) \cup \mathrm{E}(\bot)))$ follows because the minimal dependency is used.

Then $[\![\Pi]\!]$ is a proof forest; it remains to show that it is also correct. It is immediate that the translation of a tautology axiom is correct, and that translating an inference other than a cut preserves correctness. For the translation of a cut, let the proofs Π and Π', the forests $\mathrm{F_A}$, $\mathrm{F_B}$ and $\mathrm{F_C}$, and the vertex c be as above. A switching ς'' for $\mathrm{F_C}$ is the union of a switching ς for $\mathrm{F_A}$, a switching ς' for $\mathrm{F_B}$, and either $\{\mathrm{c} \mapsto \mathrm{L}\}$ or $\{\mathrm{c} \mapsto \mathrm{R}\}$. If $\varsigma''(\mathrm{c}) = \mathrm{L}$, i.e. the tree for B from $\mathrm{F_A}$ is switched on, all the propositional vertices from $\mathrm{F_A}$ under the switching ς are switched on in $\mathrm{F_C}$ (plus, possibly, some propositional vertices from $\mathrm{F_B}$). Then $val(\mathrm{F_A}, \varsigma)$ implies $val(\mathrm{F_C}, \varsigma'')$, and since the former is a tautology, so is the latter. Symmetrically, if $\varsigma''(\mathrm{c}) = \mathrm{R}$ then $val(\mathrm{F_B}, \varsigma') \Rightarrow val(\mathrm{F_C}, \varsigma'')$, and the latter must be a tautology. Then translating a cut preserves correctness. ☐

The translation of the cut immediately gives a notion of composition for proof forests. One thing to note about cuts is that, in a sequent proof, the cut-formula of an inner cut (one not at the root) may contain occurrences of eigenvariables of $\forall R$-inferences below it. When translated to a forest, these cuts will then be dependent on moves by ∀belard. However, otherwise there is nothing to distinguish them from the translation of a top-level cut. This is only natural: cut-formulae have no ancestors in a sequent proof, and since cuts may often be permuted, which cut is actually at the root is not always significant.

Translating proof forests to sequent proofs

The translation in the other direction, from proof forests to sequent proofs, will first be described for proof forests without cuts. Translation steps are mostly the direct inverse to those in the translation from proofs to forests (see also Proposition 5.5.2 below). A correct, cut-free proof forest F *translates* to a sequent proof Π in the strict calculus of Figure 5.4, written $\mathrm{F} \mapsto \Pi$, if Π can be obtained from F by the following inductive, non-deterministic procedure.

- If F contains a universal root node v, with unique edge $\langle \mathrm{v}, a, \mathrm{w} \rangle$ and label $\forall x.A$, then $\mathrm{F}|_{\mathrm{V}-\{\mathrm{v}\}}$ is a correct proof forest, obtained from F by removing the vertex v, the edge $\langle v, - \rangle w$, and any dependencies $\langle v, - \rangle w \to \mathrm{e}$. Let the sequent translation of this proof forest be the proof Π with conclusion sequent Γ, A. Then F translates to the following proof.

  $$\dfrac{\begin{matrix}\Pi \\ \vdots \\ \vdash \Gamma, A[a/x]\end{matrix}}{\vdash \Gamma,\ \forall x.A}\,\forall\mathrm{R}$$

  The side-condition of the $\forall$R rule, that the eigenvariable a may not occur free in Γ, is satisfied by condition 3 of Definition 5.4.3, by which a may not occur free in the label of any root node of F.

- If F contains an existential root node v with no edges, labelled $\exists x.A$, let the sequent translation of the correct proof forest $\mathrm{F}|_{\mathrm{V}-\{\mathrm{v}\}}$ be the proof Π with conclusion sequent Γ. Then F translates to the following proof.

  $$\dfrac{\begin{matrix}\Pi \\ \vdots \\ \vdash \Gamma\end{matrix}}{\vdash \Gamma, \exists x.A}\,\mathrm{W}\exists$$

- If F contains an existential root node v with exactly one edge $\langle \mathrm{v}, t, \mathrm{w} \rangle$, and this edge is minimal in the dependency ($\mathrm{e} \not\lesssim \langle \mathrm{v}, \mathrm{w} \rangle$ for all edges e), let $lab(\mathrm{v}) = \exists x.A$ and let the sequent translation of the correct proof forest $\mathrm{F}|_{\mathrm{V}-\{\mathrm{v}\}}$ be the proof Π with conclusion sequent Γ. Then F translates to the following proof.

  $$\dfrac{\begin{matrix}\Pi \\ \vdots \\ \vdash \Gamma, A[t/x]\end{matrix}}{\vdash \Gamma,\ \exists x.A}\,\exists\mathrm{R}$$

- If F contains an existential root node v with $n \geq 2$ edges $\langle \mathrm{v}, t_1, \mathrm{w}_1 \rangle, \ldots, \langle \mathrm{v}, t_n, \mathrm{w}_n \rangle$ and label $\exists x.A$. Let F′ be the proof forest obtained from F by distributing the edges of v over v and a fresh vertex v′, where both end up with at least one edge, as follows. For some i $(0 < i < n)$, replace the edges $\langle \mathrm{v}, t_1, \mathrm{w}_1 \rangle, \ldots, \langle \mathrm{v}, t_i, \mathrm{w}_i \rangle$ by edges $\langle \mathrm{v}', t_1, \mathrm{w}_1 \rangle, \ldots, \langle \mathrm{v}', t_i, \mathrm{w}_i \rangle$, where v′ is a fresh vertex. If F′ translates to the proof Π with conclusion sequent $\Gamma, \exists x.A, \exists x.A$, then F translates to the following proof.

$$\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma, \exists x.A, \exists x.A\end{array}}{\vdash \Gamma, \exists x.A}\,\mathrm{C}\exists$$

- If the proof forest F consists purely of a collection of propositional vertices $\mathrm{v}_1, \ldots, \mathrm{v}_n$ labelled $P_1, \ldots, P_m$, then F translates to the following proof.

$$\dfrac{}{\vdash P_1, \ldots, P_n}\,\mathrm{Taut}$$

Acyclicity of the dependency guarantees that to any proof forest at least one of the above steps applies. In particular, if a proof forest has only existential root nodes with a single edge, one of these must be minimal in the dependency.

The two translation procedures are almost inverse, but not quite. To ensure that the translation from proof forests to proofs $(\Mapsto)$ terminates, it is prevented from generating successive contractions and weakenings on the same existential formula, as illustrated below.

$$\dfrac{\dfrac{\dfrac{\vdash \Gamma, \exists x.A}{\vdash \Gamma, \exists x.A, \exists x.A}\,\mathrm{W}\exists}{\vdash \Gamma', \exists x.A, \exists x.A}}{\vdash \Gamma', \exists x.A}\,\mathrm{C}\exists$$

Such constructions of successive contractions and weakenings may occur in the sequent calculus, but are generally considered bureaucracy.

Proposition 5.5.2. *For a proof* Π *in the strict calculus of Figure 5.4, without successive contractions and weakenings,* $[\![\Pi]\!] \Mapsto \Pi$.

Proof. By inspection of the two translation procedures. □

As highlighted in Section 5.3, the translations of the cut are not inverse to one another. Firstly, how proof forests are decomposed is formalised in the lemma below.

Lemma 5.5.3. *Given a correct proof forest* F *with a cut edge* $\langle\bot,\mathrm{c}\rangle$ *such that* $\mathrm{e}\not\lesssim\langle\bot,\mathrm{c}\rangle$ *for all edges* e*, and with conjunction edges* $\langle\mathrm{c},\mathrm{L},\mathrm{x}\rangle$ *and* $\langle\mathrm{c},\mathrm{R},\mathrm{y}\rangle$*, the subforests* $\mathrm{F}|_X$ *and* $\mathrm{F}|_Y$ *are correct proof forests, where* X *and* Y *are as follows.*

$$X = \{\mathrm{v}\in\mathrm{V} \mid \mathrm{y}\not\lesssim\mathrm{v},\, \mathrm{c}\neq\mathrm{v}\} \qquad Y = \{\mathrm{v}\in\mathrm{V} \mid \mathrm{x}\not\lesssim\mathrm{v},\, \mathrm{c}\neq\mathrm{v}\}$$

Proof. It is easily seen that $\mathrm{F}|_X$ and $\mathrm{F}|_Y$ are proof forests. For correctness, for any switching ς for $\mathrm{F}|_X$ there is a switching $\varsigma\cup\{\mathrm{c}\mapsto\mathrm{L}\}$ for F that switches on the exact same propositional vertices. Then $\mathrm{F}|_X$ is correct if F is, and by symmetry so is $\mathrm{F}|_Y$. ☐

Then to complete the description of the translation procedure, a correct proof forest F with cuts translates to a sequent proof Π in the calculus of Figure 5.4, written $\mathrm{F}\Mapsto\Pi$, with the translation steps for cut-free proof forests above, plus the following one.

- If $\langle\bot,A,\mathrm{c}\rangle$ is a cut edge in F that is minimal in the dependency ($\mathrm{e}\not\lesssim\langle\bot,\mathrm{c}\rangle$ for all edges e), let $\langle\mathrm{c},L,\mathrm{x}\rangle$ and $\langle\mathrm{c},R,\mathrm{y}\rangle$ be the edges of the vertex c. Let the proof forests $\mathrm{F}|_X$ and $\mathrm{F}|_Y$, where

  $$X = \{\mathrm{v}\in\mathrm{V} \mid \mathrm{y}\not\lesssim\mathrm{v},\, \mathrm{c}\neq\mathrm{v}\} \qquad Y = \{\mathrm{v}\in\mathrm{V} \mid \mathrm{x}\not\lesssim\mathrm{v},\, \mathrm{c}\neq\mathrm{v}\}$$

  translate to Π with conclusion Γ,A and Π' with conclusion $\Gamma,A^\bot$ respectively. Then F translates to the following proof.

  $$\dfrac{\dfrac{\begin{matrix}\Pi\\ \vdots\\ \vdash\Gamma,A\end{matrix}\qquad\begin{matrix}\Pi'\\ \vdots\\ \vdash A^\bot,\Gamma\end{matrix}}{\vdash\Gamma,\Gamma}\,\mathrm{Cut}}{\vdash\Gamma}\,\mathrm{CR}$$

First, it will be argued that the translation relation is never empty.

Proposition 5.5.4. *If* F *is a correct proof forest then there is at least one sequent proof* Π *such that* $\mathrm{F}\Mapsto\Pi$.

Proof. Firstly, as was argued above, the acylicity of the dependency ensures that to every forest at least one step applies. Secondly, the translation procedure must be well-defined, in the sense that at each point the induction step is applied to a correct proof forest. That induction steps are applied to proof forests follows by an easy inspection of the translation steps, and that these are correct is immediate for all but the translation of the cut, which follows by follows from Lemma 5.5.3. Finally, the procedure must terminate. This follows from the observation that each translation step reduces the following measure: the multiset of the number of edges of each vertex in the forest, ordered by the standard multiset ordering. ☐

Cuts, permutations, and dependencies

Some of the main differences between proof forests and sequent proofs arise from the nature of the cut in both formalisms.

The translation step for cuts, from proof forests to sequent proofs, is essentially the translation from the additive cut to the multiplicative cut in the sequent calculus. This gives the formal side of the point made in Section 5.3, that cuts in proof forests are of an additive nature, but that composition of proof forests uses them in a multiplicative sense. The important technical difference between the additive cut in sequent proofs and the cut in proof forests is that the sequent cut strictly separates the two proofs it combines, Π and Π' in the translation step above, while the proof forests $\mathrm{F}|_X$ and $\mathrm{F}|_Y$ may have a common, shared part. Also, the difference between the correctness condition of proof forests, in Definition 5.4.6, and the tautology axioms of sequent calculus, disappears in the light of the translation procedure: the values of a proof forest, under all its switchings, are precisely the tautology axioms of its sequent proof translation.

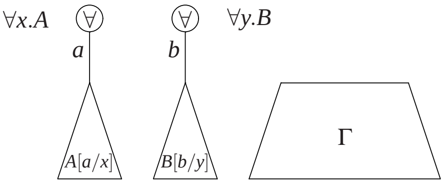
In Figure 5.10 it is illustrated how proof forests factor out the permutations in the sequent calculus. The first of the examples pictured shows the permutation of two $\forall$R-inferences; both translate to the same forest, pictured below them. The second example shows the permutation of an $\exists$R-inference with a cut. In this way the translation $[\![-]\!]$, from proofs in the strict calculus of Figure 5.4 plus cut to proof forests, factors out any permutation that the sequent calculus admits.

The dependants of an edge in a proof forest then correspond, morally, to the notion of a smallest subproof under permutations in the sequent calculus. However, in the presence of cuts the correspondence is not precise: it occurs that inferences may not permute, while their corresponding edges in the forest translation are nonetheless not dependent. Such impermutabilities occur, for example, in the following way.

$$\dfrac{\dfrac{\vdash \Gamma, B, A \qquad \vdash A^{\perp}, B, \Gamma'}{\vdash \Gamma, B, B, \Gamma'}\,\text{Cut}}{\vdash \Gamma, B, \Gamma'}\,\text{CR}$$

In the above example, the cut and the contraction cannot be permuted, because the two contracted formulae end up each in a different subproof. In proof forests, there is no corresponding dependency. This has the consequence that in a proof forest translated from a sequent proof, a the dependants of an edge may be strictly smaller than the minimal subproof of the inference it is a translation of.

$$\dfrac{\dfrac{\vdash A[a/x], B[b/x], \Gamma}{\vdash A[a/x],\ \forall y.B,\ \Gamma}\,\forall\mathrm{R}}{\vdash\ \forall x.A,\ \ \forall y.B,\ \Gamma}\,\forall\mathrm{R} \qquad \dfrac{\dfrac{\vdash A[a/x], B[b/x], \Gamma}{\vdash\ \forall x.A,\ B[b/x], \Gamma}\,\forall\mathrm{R}}{\vdash\ \forall x.A,\ \ \forall y.B,\ \Gamma}\,\forall\mathrm{R}$$



$$\dfrac{\vdash \Gamma, A \qquad \dfrac{\vdash A^{\perp}, B[t/x], \Gamma'}{\vdash A^{\perp},\ \exists x.B,\ \Gamma'}\,\exists\mathrm{R}}{\vdash \Gamma, \exists x.B, \Gamma'}\,\mathrm{Cut} \qquad \dfrac{\dfrac{\vdash \Gamma, A \qquad \vdash A^{\perp}, B[t/x], \Gamma'}{\vdash \Gamma, B[t/x], \Gamma'}\,\mathrm{Cut}}{\vdash \Gamma,\ \exists x.B,\ \Gamma'}\,\exists\mathrm{R}$$

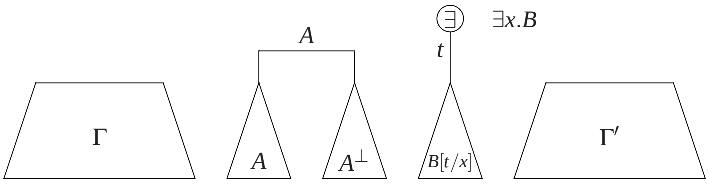


Figure 5.10: Permutations are factored out in proof forests

To summarise, in the absence of the cut, proof forests abstract over the linear order of inferences in a sequent proof, the translations back and forth are essentially inverse to one another, and dependency corresponds exactly to non-permutability. The addition of cuts increases the differences between proof forests and sequent proofs: translations are not inverse, and not all causes of non-permutability are captured in the dependency.

Chapter 6

Cut-elimination in classical proof forests

6.1 Introduction

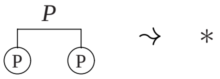
In this chapter, cut-elimination for classical proof forests will be discussed. The cut-reduction steps for classical proof forests, presented in Section 6.2, will be based on a natural notion of composition of strategies, and correspond closely to reduction steps in the sequent calculus. However, these reduction steps turn out to be far from well-behaved. A first hint of this, in Section 6.2, is the existence of cuts configured in such a way that they cannot be reduced; such cuts will be called *unsafe*. Then in Section 6.3, a problematic proof forest will be presented, dubbed the *universal counterexample*. Though it may arise in the translation of a sequent proof, or by composition, it has infinite reduction paths, and reducing it introduces unsafe cuts. (That it is also non-confluent is shown in Section 7.4.)

To obtain weak normalisation, in Section 6.4 two modifications to the reduction relation are proposed. The problem of unsafe cuts is addressed by a simple operation called *pruning*, which may be added to rewrite steps. A further modification groups together the reduction steps on the same cut. The modified reduction relation thus obtained is then shown to be weakly normalising, and conjectured to be strongly normalising.

6.2 Reductions

The cut-reduction steps in proof forests will come in four kinds: for propositional cuts, and for first-order cuts with zero, with one, and with more existential branches. The reduction steps are natural from a game-theoretic perspective, and similar in spirit to those in the sequent calculus, although of course there will be technical differences. However, it will turn out that reduction steps are not naturally well-behaved, and that certain cuts cannot be reduced. The four reduction steps will first be introduced informally, omitting in part how the dependency is treated, but with enough detail to show where the problems arise.

I. Propositional reduction steps Firstly, a propositional cut is reduced in a *propositional reduction step*, which simply removes the cut from the proof forest. In the illustration below, the asterisk on the right indicates that nothing remains of the cut itself; the unaffected parts of the proof forest are not displayed.

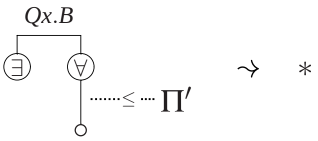


The corresponding reduction in the sequent calculus, on a cut with a propositional cut-formula, is illustrated below.

$$\dfrac{\dfrac{}{\vdash \Gamma, P}\,\text{Taut} \qquad \dfrac{}{\vdash P^{\perp}, \Gamma'}\,\text{Taut}}{\vdash \Gamma, \Gamma'}\,\text{Cut} \quad \Rightarrow \quad \dfrac{}{\vdash \Gamma, \Gamma'}\,\text{Taut}$$

After permuting the cut upwards until on both sides only a tautology axiom remains above it, the cut is removed, and the two tautology axioms replaced by a single one.

II. Disposal steps Next, a cut on a first-order formula with no existential branches is reduced in a *disposal step*, pictured below.

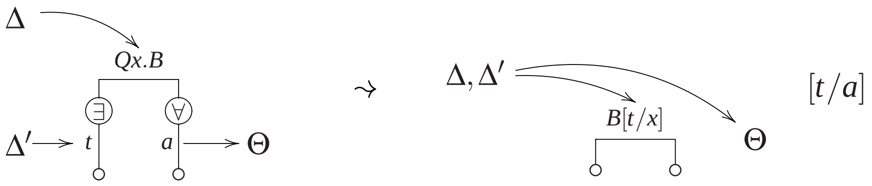


The reduction step removes the cut plus all its dependants; in the above illustration the dependants of the universal edge of the cut are represented as Π'. This is similar to what happens in the corresponding reduction step in the sequent calculus, for a cut on a weakened formula, depicted below.

$$\dfrac{\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma\end{array}}{\vdash \Gamma, A}\,\mathrm{WR} \qquad \begin{array}{c}\Pi'\\ \vdots\\ \vdash A^{\perp}, \Gamma'\end{array}}{\vdash \Gamma, \Gamma'}\,\mathrm{Cut} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma\end{array}}{\vdash \Gamma, \Gamma'}\,\mathrm{WR}$$

The reduction step removes the subproof Π', on the opposite side of the weakening. The other formulae in the removed subproof, depicted by Γ', are introduced by weakenings in the result. A disposal step may remove individual branches of an existential node, while leaving other branches and the node itself untouched. That this can be seen as similar to introducing weakenings becomes explicit when a disposal step removes all the remaining branches of an existential node, leaving it as a leaf.

III. Logical reduction steps The reduction step for a cut with exactly one existential branch, a *logical reduction step*, implements the external interpretation of the cut, as two strategies playing against each other, described in Section 5.3. In this interpretation ∀belard's choice on one side of the cut mirrors ∃loise's move on the other side. The reduction step, depicted below, makes this identification at a syntactic level, by substituting all occurrences of ∀belard's eigenvariable with ∃loise's witnessing term.

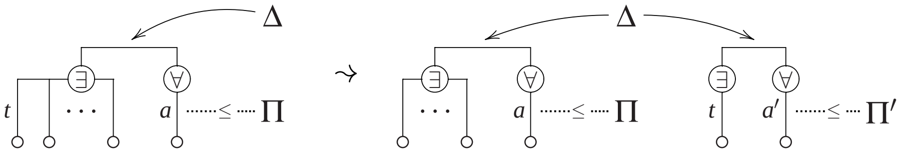


In the diagram, the dependency is adjusted according to the global substitution $[t/a]$, while preserving existing dependencies. For the dependencies from Δ': any eigenvariable b that is free in t will, in the result, be free in the new cut-formula $B[t/x]$ and in any witnessing term or cut-formula where a was free before. For those from Δ: any eigenvariable free in $Qx.B$ will be free in $B[t/x]$, and the dependencies from Δ to Θ are preserved. The corresponding reduction in sequent calculus is similar.

$$\dfrac{\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma, B[t/x]\end{array}}{\vdash \Gamma, \exists x.B}\,\exists\mathrm{R} \qquad \dfrac{\begin{array}{c}\Pi'\\ \vdots\\ \vdash \Gamma', B^{\perp}[a/x]\end{array}}{\vdash \Gamma', \forall x.B^{\perp}}\,\forall\mathrm{R}}{\vdash \Gamma, \Gamma'}\,\mathrm{Cut} \quad \Rightarrow \quad \dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \Gamma, B[t/x]\end{array} \qquad \begin{array}{c}\Pi'[t/a]\\ \vdots\\ \vdash \Gamma', B^{\perp}[t/x]\end{array}}{\vdash \Gamma, \Gamma'}\,\mathrm{Cut}$$

The reduction step applies to a cut on first-order formulae introduced by logical rules (∀R and ∃R). After permuting the two inference rules to be immediately above the cut, the two logical inferences are removed, the cut is replaced by one on the premises of the logical rules, and the substitution $[t/a]$ is applied to all (relevant) occurrences of the eigenvariable a.

IV. Structural reduction steps In the game interpretation, for a cut with two or more existential branches there are several moves by ∃loise, and just one for ∀belard. To allow these to be identified, the natural approach is to make copies of ∀belard's move, until there is one for each of ∃loise's moves. Along with ∀belard's move, the minimum that must be duplicated is its dependants: these are the moves that respond, directly or indirectly, to ∀belard's move, and for each different choice by ∀belard a different response must be permitted. A *structural reduction step*, on a first-order cut with two or more existential branches, is then as follows.



The reduction step duplicates the cut and all its dependants on the universal side, represented by Π, and moves one existential branch, the one assigning the witness t above, from the original cut to the duplicated one. The eigenvariables of the duplicated dependants Π' are renamed, in the way that a' is. The duplicated cut is dependent on the same edges and vertices that the original was, and likewise dependencies towards the existential branches of the cut, including that assigning t, are preserved.

A corresponding proof transformation in the sequent calculus, for a cut on a contracted formula, is depicted below.

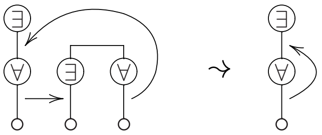
$$\dfrac{\dfrac{\begin{array}{c}\Theta\\ \vdots\\ \vdash \Gamma, A, A\end{array}}{\vdash \Gamma, A}\,\text{CR} \qquad \begin{array}{c}\Pi\\ \vdots\\ \vdash A^{\perp}, \Gamma'\end{array}}{\vdash \Gamma, \Gamma'}\,\text{Cut} \quad \Rightarrow \quad \dfrac{\dfrac{\dfrac{\begin{array}{c}\Theta\\ \vdots\\ \vdash \Gamma, A, A\end{array} \qquad \begin{array}{c}\Pi\\ \vdots\\ \vdash A^{\perp}, \Gamma'\end{array}}{\vdash \Gamma, \Gamma', A}\,\text{Cut} \qquad \begin{array}{c}\Pi\\ \vdots\\ \vdash A^{\perp}, \Gamma'\end{array}}{\vdash \Gamma, \Gamma', \Gamma'}\,\text{Cut}}{\vdash \Gamma, \Gamma'}\,\text{CR}$$

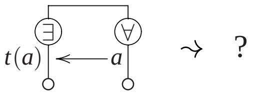
The subproof Π, on the other side of the cut than the contraction, is duplicated, and to remove the contraction each of its premises A are connected to one of the subproofs Π with a cut. The contractions on Γ' correspond, in proof forests, to the duplication of the edges on an existential node, but not the node itself. It should be noted that the above sequent proof transformation is not strongly normalising when both cut-formulae are contracted—see, e.g., [28, Section 3].

The reduction steps follow naturally from the interpretation of the cut as strategies playing against each other: witnesses and eigenvariables on either side of a cut are identified, and when backtracking occurs on one side, the other strategy is modified to respond, uniformly, to each witness it is presented with. The reduction steps are also closely related to their counterparts in sequent calculus, with the removal and duplication of dependants corresponding to removal and duplication of (smallest) subproofs. However, there is one caveat, discussed at the end of Section 5.5: in the presence of cuts, the correlation between a set of dependants and a smallest subproof is imprecise, and the former may be strictly smaller than the latter. As a consequence, the reduction behaviour of both formalisms will be significantly different—this will be addressed in Section 6.3.

Safety

With logical and structural reduction steps, problems occur when there are dependencies between the universal and the existential edges of a cut. Below on the left, if the unique existential edge of a cut depends on the universal edge, reducing the cut with a logical reduction step creates a cycle in the dependency.





Above right, the eigenvariable a of the universal edge of a cut occurs free in the witness $t(a)$ of the existential edge. Semantically, reducing this cut would require ∀belard's witness a and ∃loise's witness $t(a)$ to be identified. Resolving the cut with a substitution $[t(a)/a]$, which leaves free occurrences of the variable a in the substituted terms $t(a)$, is clearly undesirable.

A structural reduction step on a cut with a dependency between its universal edge and an existential edge is also problematic. From the informal description of the reduction step it is not immediately obvious how the different elements, duplicating the cut and moving one existential edge to the duplicate, should be applied. The illustration in Figure 6.1 explores the three options that conform to the following, reasonable, constraints: the dependent edge should be duplicated, with one copy dependent on the original universal edge, and the other on the duplicated edge; and in the result the original cut and its copy should each have at least one existential edge. In the two upper central diagrams, the cut that is being reduced, the *primary* cut, is indicated by the black token. The first two possible reduction steps pictured above return, in one logical

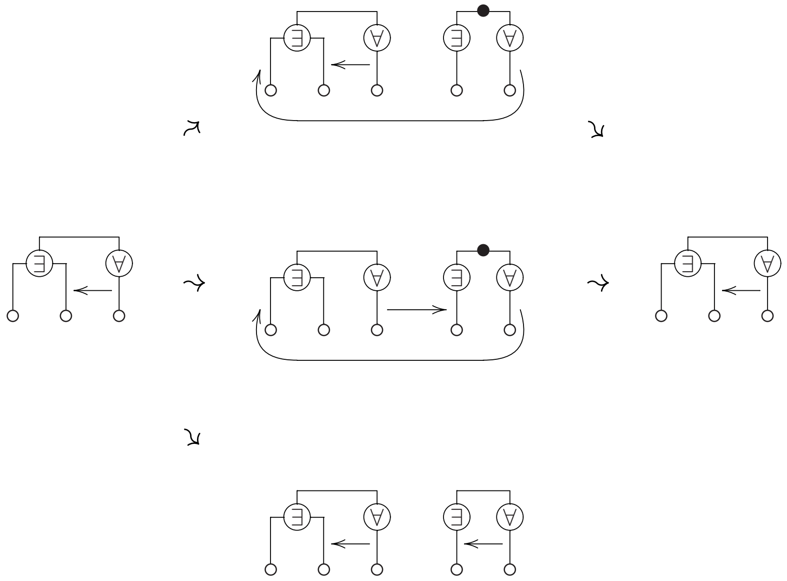


Figure 6.1: Structural steps on unsafe cuts create reduction cycles

reduction step, to the original configuration, creating a cyclic reduction path. The third possibility leaves the original cut intact, while its duplication creates a problematic logical cut.

As the above illustrates, the configuration where an existential edge of a cut depends on the universal edge of that same cut creates serious problems for cut reduction. It is also an unnatural configuration: it does not arise from composition—and hence not from the translation of sequent proofs—since there will be no dependencies between the two proof forests that are composed. This observation provides a reasonable constraint to impose on cut reduction.

Definition 6.2.1 (Safety)**.** A cut c is *safe* if its dependants on both sides are disjoint. That is, let c have the edges $\langle c, L, x\rangle$ and $\langle c, R, y\rangle$; then c is safe if

$$\neg\exists v \in V.\ x \leq v \ \wedge\ y \leq v\ .$$

A proof forest is *safe* if all its cuts are safe.

The reduction steps, as they are defined below, will apply only to safe cuts. The restriction thus imposed on reduction is intentionally weak. A stronger criterion would be to confine reduction steps to forests that are the translation of a sequent proof—ideas in this direction are explored in Chapter 7. However, one aim of the present approach is to investigate proof reductions in a general setting, independent of those in the sequent calculus, and for this reason the present, weaker constraint is employed.

Formal definitions

Before defining the reduction steps formally, it will be explained how the duplication in structural reduction steps is implemented. Briefly, duplication proceeds as follows: the vertices in the part in a proof forest that is to be duplicated are first renamed using a substitution; then the renamed forest and the original forest are combined by taking their union, which is defined pointwise over their components.

First, let the *union* of two pre-proof forest be given as follows.

$$\mathrm{F_A} \cup \mathrm{F_B} \triangleq \big(\,\mathrm{V_A} \cup \mathrm{V_B},\ \ \bot_\mathrm{A},\ \ lab_\mathrm{A} \cup lab_\mathrm{B},\ \ \mathrm{E_A} \cup \mathrm{E_B},\ \ (\rightarrow_\mathrm{A}) \cup (\rightarrow_\mathrm{B})\,\big)\ [\bot_\mathrm{A}/\bot_\mathrm{B}]$$

The special $\bot$-vertex in the union is obtained by merging the $\bot$-vertices of the component forests by a substitution. Then Figure 6.2 illustrates how substitution and union are used to implement duplication. To copy the dependants of the node v in the forest $\mathrm{F_A}$ (these are the vertices v, y, and z), first the forest $\mathrm{F_B}$ is created by applying the substitutions $[\mathrm{v'/v}]$, $[\mathrm{y'/y}]$, and $[\mathrm{z'/z}]$. In addition, the eigenvariable b is renamed to b', because it belongs to an edge that is duplicated. Then $\mathrm{F_A}$ and $\mathrm{F_B}$ are combined by taking their union.

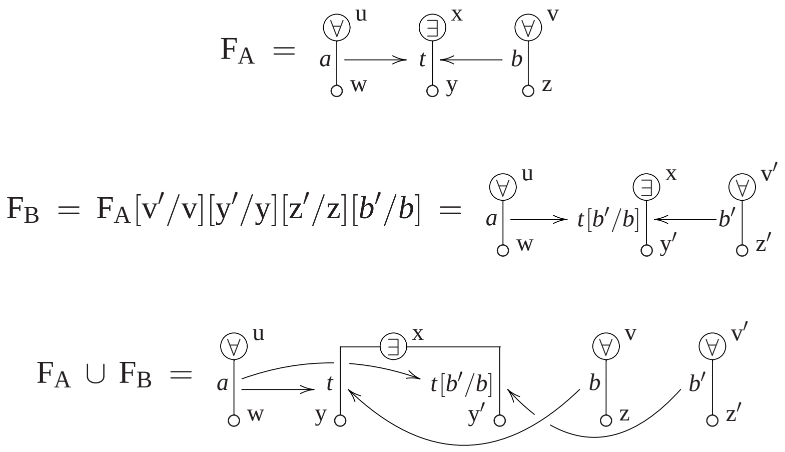
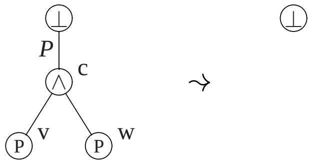

$\mathrm{F_A} =$
u
x
v
a
t
b
w
y
z
$\mathrm{F_B} = \mathrm{F_A}[\mathrm{v'/v}][\mathrm{y'/y}][\mathrm{z'/z}][b'/b] =$
u
x
v′
a
t[b′/b]
b′
w
y′
z′
$\mathrm{F_A} \cup \mathrm{F_B} =$
u
x
v
v′
a
t
t[b′/b]
b
b′
w
y
y′
z
z′

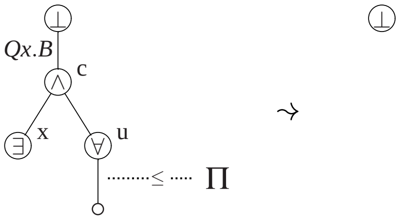
Figure 6.2: Duplication (of the node v and its dependants)

The formal definitions of the reduction steps, below, are accompanied by further illustrations.

Definition 6.2.2 (I. Propositional reduction steps). Let F be a proof forest with a cut $\langle \bot, P, \mathrm{c} \rangle$, where P is a propositional formula, and edges $\langle \mathrm{c}, \mathrm{L}, \mathrm{v} \rangle$ and $\langle \mathrm{c}, \mathrm{R}, \mathrm{w} \rangle$. Then $\mathrm{F} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F}|_X$ with a *propositional reduction step*, where $X = V - \{\mathrm{c}, \mathrm{v}, \mathrm{w}\}$.



Definition 6.2.3 (**II**. Disposal reduction steps)**.** Let F be a proof forest with a cut $\langle\bot,\mathrm{c}\rangle$ and edges $\langle\mathrm{c},\mathrm{u}\rangle$ and $\langle\mathrm{c},\mathrm{x}\rangle$, where the vertex x is an existential leaf. Then $\mathrm{F}\overset{\mathrm{c}}{\rightsquigarrow}\mathrm{F}|_X$ with a *disposal reduction step*, where $X=\{\mathrm{v}\in\mathrm{V}\mid\mathrm{c}\not\leq\mathrm{v}\}$.



In the illustration above, also the cut-formula is $Qx.B$ indicated, where Q is a quantifier, and the dependants Π of the universal edge of the cut. What is removed in the reduction step is the vertices c, x, u, and those in Π, plus their edges and dependencies.

Definition 6.2.4 (**III**. Logical reduction steps)**.** Let $\mathrm{F_A}$ be a proof forest with a safe cut $\langle\bot,Qx.B,\mathrm{c}\rangle$ where $Q\in\{\forall,\exists\}$, edges $\langle\mathrm{c},U,\mathrm{u}\rangle$ and $\langle\mathrm{c},X,\mathrm{x}\rangle$ where $\{U,X\}=\{\mathrm{L},\mathrm{R}\}$, and edges $\langle\mathrm{u},a,\mathrm{w}\rangle$ and $\langle\mathrm{x},t,\mathrm{y}\rangle$, where x is an existential vertex with exactly one edge, and $u\not\leq_{\mathrm{A}}y$. Then $\mathrm{F_A}\overset{\mathrm{c}}{\rightsquigarrow}\mathrm{F_B}$ with a *logical reduction step*, where $\mathrm{F_B}$ is defined as follows.

- $\mathrm{V_B}=\mathrm{V_A}-\{\mathrm{u},\mathrm{x}\}$
- $lab_{\mathrm{B}}(\mathrm{c})=B\wedge B^{\bot}[t/x]$; otherwise $lab_{\mathrm{B}}(\mathrm{v})=lab_{\mathrm{A}}(\mathrm{v})[t/a]$
- $\mathrm{E_B}$ is obtained from $\mathrm{E_A}$ by replacing the five edges

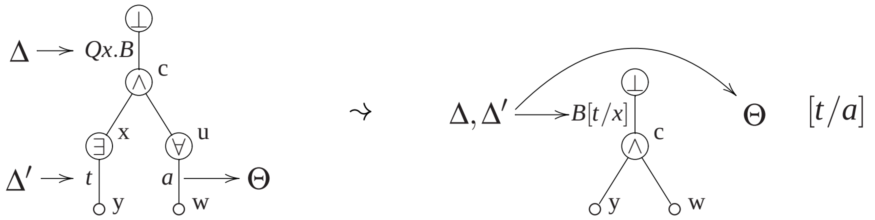
  $$\langle\bot,Qx.B,\mathrm{c}\rangle\qquad\langle\mathrm{c},U,\mathrm{u}\rangle\qquad\langle\mathrm{c},X,\mathrm{x}\rangle\qquad\langle\mathrm{u},a,\mathrm{w}\rangle\qquad\langle\mathrm{x},t,\mathrm{y}\rangle$$

  with the three edges

  $$\langle\bot,B[t/x],\mathrm{c}\rangle\qquad\langle\mathrm{c},U,\mathrm{w}\rangle\qquad\langle\mathrm{c},X,\mathrm{y}\rangle$$

  and replacing any other edge $\langle\mathrm{v}_1,Y,\mathrm{v}_2\rangle$ with $\langle\mathrm{v}_1,Y[t/a],\mathrm{v}_2\rangle$
- The relation $(\rightarrow_{\mathrm{B}})$ is the smallest relation on $\mathrm{E_B}$ such that

  $$\begin{array}{lll} \mathrm{e}_1\rightarrow_{\mathrm{B}}\mathrm{e}_2 & \text{if} & \mathrm{e}_1\rightarrow_{\mathrm{A}}\mathrm{e}_2,\quad\text{or}\\ & & \mathrm{e}_1\rightarrow_{\mathrm{A}}\langle\bot,\mathrm{c}\rangle\text{ and }\langle\mathrm{u},\mathrm{w}\rangle\rightarrow_{\mathrm{A}}\mathrm{e}_2,\quad\text{or}\\ & & \mathrm{e}_1\rightarrow_{\mathrm{A}}\langle\mathrm{x},\mathrm{y}\rangle\text{ and }\langle\mathrm{u},\mathrm{w}\rangle\rightarrow_{\mathrm{A}}\mathrm{e}_2,\quad\text{or}\\ & & \mathrm{e}_1\rightarrow_{\mathrm{A}}\langle\mathrm{x},\mathrm{y}\rangle\text{ and }\mathrm{e}_2=\langle\bot,\mathrm{c}\rangle \end{array}$$

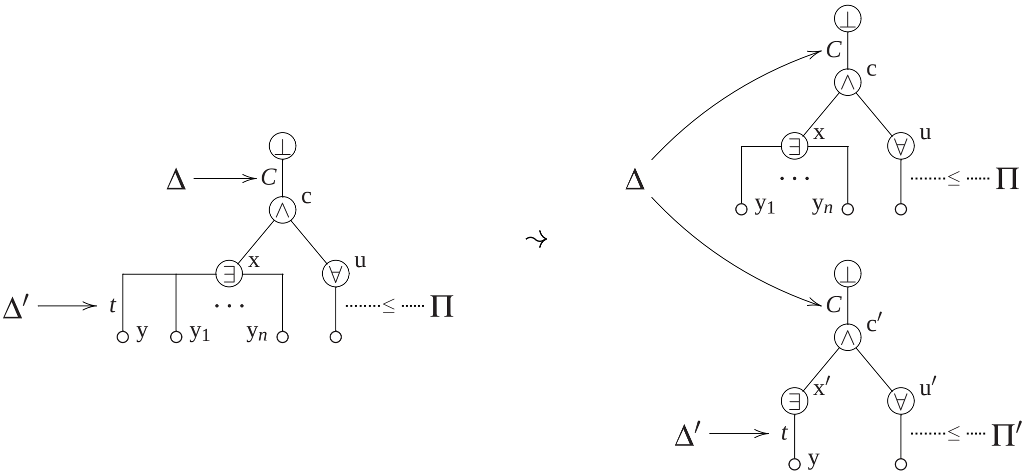


Definition 6.2.5 (**IV**. Structural reduction steps)**.** Let $\mathrm{F_A}$ be a proof forest with a safe cut $\langle\bot,\mathrm{c}\rangle$, edges $\langle\mathrm{c},\mathrm{u}\rangle$ and $\langle\mathrm{c},\mathrm{x}\rangle$, and existential edges $\langle\mathrm{x},\mathrm{y}\rangle$ and $\langle\mathrm{x},\mathrm{y}_1\rangle,\dots,\langle\mathrm{x},\mathrm{y}_n\rangle$. Then $\mathrm{F_A} \overset{\mathrm{c,y}}{\rightsquigarrow} \mathrm{F_B}$ with a *structural reduction step*, where $\mathrm{F_B}$ is defined as follows. Let X be following set of vertices, and let ρ and σ be the following substitution maps on nodes and (eigen)variables, respectively (where $<$ is the strict version of $\leq$).

$$\begin{array}{rcl} X & = & \{\mathrm{v}\in\mathrm{V}\mid \mathrm{x}\not<_{\mathrm{A}}\mathrm{v}\} \\ \rho & = & \{\mathrm{v}\mapsto\mathrm{v}'\mid \mathrm{v}\in\{\mathrm{c},\mathrm{x}\}\ \vee\ \mathrm{u}\leq_{\mathrm{A}}\mathrm{v}\} \\ \sigma & = & \{a\mapsto a'\mid \langle\mathrm{v},a,\mathrm{w}\rangle\in\mathrm{E_A}(\forall)\ \wedge\ \mathrm{u}\leq_{\mathrm{A}}\langle\mathrm{v},\mathrm{w}\rangle\} \end{array}$$

where all v' and a' are fresh for $\mathrm{F_A}$ (and w.r.t. each other). Then $\mathrm{F_B}$ is as follows:

$$\mathrm{F_B} = \big(\,\mathrm{F_A}\ \cup\ \mathrm{F_A}|_X[\rho][\sigma]\,\big)\ [\langle\mathrm{x}',\mathrm{y}\rangle/\langle\mathrm{x},\mathrm{y}\rangle]\ .$$



Technically, a structural step proceeds as follows. The dependants of the existential side, of the vertex x, are removed, and the cut and its universal side are renamed, creating $\mathrm{F_A}|_X[\rho][\sigma]$. The effect of taking the union of this proof forest with the original $\mathrm{F_A}$ is to create a duplicate c' of the cut c, but without any existential branches. Then the substitution $[\langle\mathrm{x}',\mathrm{y}\rangle/\langle\mathrm{x},\mathrm{y}\rangle]$ moves the edge $\langle\mathrm{x},\mathrm{y}\rangle$ from the original cut to the duplicate.

The design of the reduction step depends on the assumption that c is safe. Otherwise, if some dependant of u depends also on x, it will be deleted in $\mathrm{F_A}|_X$. Then the subforest of u' is strictly smaller than that of u, while it should be an exact duplicate.

For a structural reduction step $\mathrm{F_A} \overset{\mathrm{c,y}}{\rightsquigarrow} \mathrm{F_B}$ the superscript y indicates the *primary edge* of the reduction step, and may be omitted. The superscript c in any reduction step $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_B}$, indicating the *primary cut*, may likewise be omitted.

Basic properties

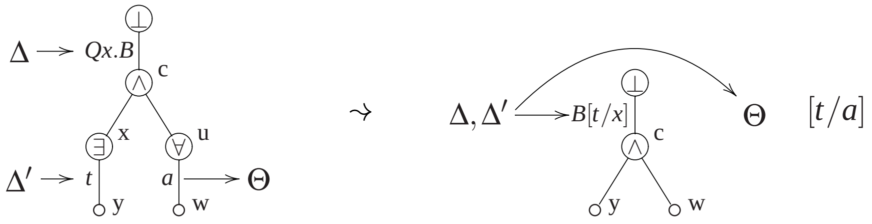
The first main properties of reductions to be established are that they preserve the axioms of proof forests, in Definition 5.4.3, and that they preserve correctness, Definition 5.4.6. For propositional and disposal steps, which only remove nodes and edges, this is mostly straightforward. On the other hand, logical and structural reduction steps involve adding and restructuring edges and dependencies, which makes in particular showing that they preserve the antisymmetry of the dependency ordering non-trivial. To provide a technical basis, the following two lemmata describe how logical and structural reduction steps modify the dependency $(\leq)$ on a forest.

Lemma 6.2.6. *In a logical reduction step* $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_B}$*, where* $\mathrm{c,u,w,x,y}$ *are as in Definition 6.2.4, for all* $\mathrm{v_1,v_2} \in \mathrm{V_B}$*,*

$$\begin{array}{rcrl} \mathrm{v_1} \leq_\mathrm{A} \mathrm{v_2} & \Rightarrow & & \mathrm{v_1} \leq_\mathrm{B} \mathrm{v_2} \\ & & \textit{or} & \mathrm{v_1} \in \{\bot, \mathrm{c}\} \;\wedge\; \exists \mathrm{e}.\ \langle \mathrm{u,w} \rangle \to_\mathrm{A} \mathrm{e} \leq_\mathrm{A} \mathrm{v_2} \end{array}$$

$$\begin{array}{rcrl} \mathrm{v_1} \leq_\mathrm{B} \mathrm{v_2} & \Rightarrow & & \mathrm{v_1} \leq_\mathrm{A} \mathrm{v_2} \\ & & \textit{or} & \mathrm{v_1} \leq_\mathrm{A} \langle \mathrm{x,y} \rangle \;\wedge\; \langle \mathrm{u,w} \rangle \leq_\mathrm{A} \mathrm{v_2} \\ & & \textit{or} & \mathrm{v_1} \leq_\mathrm{A} \langle \mathrm{x,y} \rangle \;\wedge\; \mathrm{v_2} = \mathrm{c}\ . \end{array}$$

Proof. For convenience, an illustration of the reduction step is reproduced below.



A dependency $\mathrm{v_1} \leq \mathrm{v}_n$ arises from a sequence $\mathrm{v_1}, \ldots, \mathrm{v}_n$ where for each $i \leq n$ either

$$\langle \mathrm{v}_{i-1}, \mathrm{v}_i \rangle \in \mathrm{E} \qquad \text{or} \qquad \langle \mathrm{v}_{i-1}, \mathrm{z} \rangle \;\to\; \langle \mathrm{z}', \mathrm{v}_i \rangle$$

for some vertices z, z'—note that no steps of the form $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to \mathrm{e} \to \langle \mathrm{z}', \mathrm{v}_i\rangle$ or similar are possible, since the same edge is never both a source and target in $(\to)$.

For the first statement, let $\mathrm{v}_1 \leq_{\mathrm{A}} \mathrm{v}_n$. Firstly, if no v_i is y or u, then also no v_i is x, since $\mathrm{v}_n \neq \mathrm{x}$ because $\mathrm{v}_n \in \mathrm{V_B}$ and otherwise v_{i+1} would have to be y. Then any edge $\langle \mathrm{v}_{i-1}, \mathrm{v}_i\rangle \in \mathrm{E_A}$ has a counterpart $\langle \mathrm{v}_{i-1}, \mathrm{v}_i\rangle \in \mathrm{E_B}$, and if $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \mathrm{z}', \mathrm{v}_i\rangle$ then also $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{B}} \langle \mathrm{z}', \mathrm{v}_i\rangle$. Next, if some v_i is u, then $\mathrm{v}_{i-1} = \mathrm{c}$ and either $\mathrm{v}_{i+1} = \mathrm{w}$ or $\langle \mathrm{u}, \mathrm{w}\rangle \to_{\mathrm{A}} \langle \mathrm{z}', \mathrm{v}_{i+1}\rangle$. In the former case, $\langle \mathrm{c}, \mathrm{w}\rangle = \langle \mathrm{v}_{i-1}, \mathrm{v}_{i+1}\rangle \in \mathrm{E_B}$. In the latter case, if $\mathrm{v}_1 \in \{\bot, \mathrm{c}\}$ the second disjunct of the statement applies; otherwise the sequence $\mathrm{v}_1, \ldots, \mathrm{v}_n$ contains a section $\mathrm{v}_{i-2}, \ldots, \mathrm{v}_{i+1}$ where

$$\langle \mathrm{v}_{i-2}, \mathrm{z}\rangle \ \to_{\mathrm{A}}\ \langle \bot, \mathrm{c}\rangle,\ \langle \mathrm{c}, \mathrm{u}\rangle,\ \langle \mathrm{u}, \mathrm{w}\rangle \ \to_{\mathrm{A}}\ \langle \mathrm{z}', \mathrm{v}_{i+1}\rangle\ ,$$

in which case $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{B}} \langle \mathrm{z}', \mathrm{v}_{i+1}\rangle$. Finally, if some v_i is y, then either $\mathrm{v}_{i-1} = \mathrm{x}$ and $\mathrm{v}_{i-2} = \mathrm{c}$, in which case $\langle \mathrm{c}, \mathrm{y}\rangle = \langle \mathrm{v}_{i-2}, \mathrm{v}_i\rangle \in \mathrm{E_B}$, or $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y}\rangle$, in which case $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{B}} \langle \bot, \mathrm{c}\rangle$ and $\langle \mathrm{c}, \mathrm{y}\rangle \in \mathrm{E_B}$.

For the second statement, let $\mathrm{v}_1 \leq_{\mathrm{B}} \mathrm{v}_n$. Firstly, for each edge $\langle \mathrm{v}_{i-1}, \mathrm{v}_i\rangle$ in $\mathrm{E_B}$ there is also an edge $\langle \mathrm{v}_{i-1}, \mathrm{v}_i\rangle$ in $\mathrm{E_A}$, except for $\langle \mathrm{c}, \mathrm{y}\rangle$ and $\langle \mathrm{c}, \mathrm{w}\rangle$, which have corresponding paths $\langle \mathrm{c}, \mathrm{x}\rangle$, $\langle \mathrm{x}, \mathrm{y}\rangle$ and $\langle \mathrm{c}, \mathrm{u}\rangle$, $\langle \mathrm{u}, \mathrm{w}\rangle$. Next, for $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{B}} \langle \mathrm{z}', \mathrm{v}_i\rangle$ Definition 6.2.4 gives four options.

1. $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \mathrm{z}', \mathrm{v}_i\rangle$

2. $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \bot, \mathrm{c}\rangle$ and $\langle \mathrm{u}, \mathrm{w}\rangle \to_{\mathrm{A}} \langle \mathrm{z}', \mathrm{v}_i\rangle$
   Then $\mathrm{v}_{i-1} \leq_{\mathrm{A}} \mathrm{v}_i$ because also $\langle \mathrm{c}, \mathrm{u}\rangle \in \mathrm{E_A}$.

3. $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y}\rangle$ and $\langle \mathrm{u}, \mathrm{w}\rangle \to_{\mathrm{A}} \langle \mathrm{z}', \mathrm{v}_i\rangle$
   Then $\mathrm{v}_{i-1} \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y}\rangle$ and $\langle \mathrm{u}, \mathrm{w}\rangle \leq_{\mathrm{A}} \mathrm{v}_i$, and the second disjunct of the statement applies.

4. $\langle \mathrm{v}_{i-1}, \mathrm{z}\rangle \to_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y}\rangle$ and $\langle \mathrm{z}', \mathrm{v}_i\rangle = \langle \bot, \mathrm{c}\rangle$
   Unless $\mathrm{v}_n = \mathrm{c}$, in which case the third disjunct of the statement applies, v_{i+1} is either y or w. In the former case it is immediate that $\mathrm{v}_{i-1} \leq_{\mathrm{A}} \mathrm{y}$; in the latter the second disjunct of the statement applies, since both $\mathrm{v}_{i-1} \to_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y}\rangle$ and $\langle \mathrm{u}, \mathrm{w}\rangle \leq_{\mathrm{A}} \mathrm{w} = \mathrm{v}_{i+1}$.

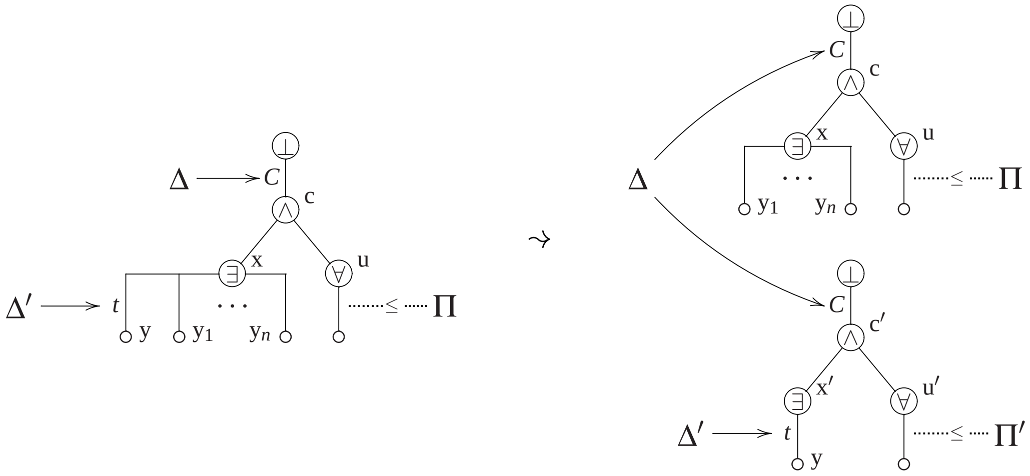
(Note that the last two cases cannot apply for more than one $i \leq n$ without there being a cycle in $(\leq_{\mathrm{A}})$ or c being unsafe.) ☐

For structural reduction steps, there is the following lemma.

Lemma 6.2.7. *In a structural reduction step* $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_B}$, *where the nodes* $\mathrm{c}, \mathrm{u}, \mathrm{x}, \mathrm{y}_i$ *and the renaming convention* $\mathrm{v} \mapsto \mathrm{v}'$ *are as in Definition 6.2.5, for all* $\mathrm{v}, \mathrm{w} \in \mathrm{V_A}$,

$$\begin{array}{lcl} \mathrm{v} \leq_{\mathrm{B}} \mathrm{w} & \Rightarrow & \mathrm{v} \leq_{\mathrm{A}} \mathrm{w} \\ \mathrm{v} \leq_{\mathrm{B}} \mathrm{w}' & \Rightarrow & \mathrm{v} \leq_{\mathrm{A}} \mathrm{w} \ \wedge \ \mathrm{u} \not\leq_{\mathrm{A}} \mathrm{v} \\ \mathrm{v}' \leq_{\mathrm{B}} \mathrm{w} & \Rightarrow & \mathrm{v} \leq_{\mathrm{A}} \mathrm{w} \ \wedge \ \mathrm{v} \in \{\mathrm{c}, \mathrm{x}\} \ \wedge \ \mathrm{y}_i \leq_{\mathrm{A}} \mathrm{w} \\ \mathrm{v}' \leq_{\mathrm{B}} \mathrm{w}' & \Rightarrow & \mathrm{v} \leq_{\mathrm{A}} \mathrm{w} \end{array}$$

Proof. For convenience, an illustration of the reduction step is reproduced below.



It is immediate from the way duplication is implemented that the dependencies

$$\mathrm{v} \leq_{\mathrm{B}} \mathrm{w} \qquad \mathrm{v} \leq_{\mathrm{B}} \mathrm{w}' \qquad \mathrm{v}' \leq_{\mathrm{B}} \mathrm{w} \qquad \mathrm{v}' \leq_{\mathrm{B}} \mathrm{w}'$$

are mirrored by a dependency $\mathrm{v} \leq_{\mathrm{A}} \mathrm{w}$. For the remaining parts of the statement, if $\mathrm{u} \leq_{\mathrm{A}} \mathrm{v} \leq_{\mathrm{A}} \mathrm{w}$ then $\mathrm{v} \leq_{\mathrm{B}} \mathrm{w}$ and $\mathrm{v}' \leq_{\mathrm{B}} \mathrm{w}'$, but not $\mathrm{v}' \leq_{\mathrm{B}} \mathrm{w}$ or $\mathrm{v} \leq_{\mathrm{B}} \mathrm{w}'$. Firstly, this means that if $\mathrm{v} \leq_{\mathrm{B}} \mathrm{w}'$ then v cannot be a dependant of u in $\mathrm{F_A}$. Secondly, if $\mathrm{v}' \leq_{\mathrm{B}} \mathrm{w}$ then, since neither v nor w can depend on u in $\mathrm{F_A}$ but v is still a duplicated vertex, v must be c or x; moreover, the dependants of c' and x' in $\mathrm{F_B}$ include, besides c' and x', only those of y_i and those of u'; then w must be among the former. □

With the description of how dependencies are modified by logical and structural reduction steps complete, it can now be shown that reductions preserve the axioms of proof forests.

Proposition 6.2.8. *If* $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_B}$ *then* $\mathrm{F_B}$ *is a proof forest.*

Proof. For all four kinds of reduction step, $\mathrm{F_B}$ is straightforwardly seen to obey most conditions of Definition 5.4.3. The following details are treated explicitly.

1. All nodes in V are in legal configurations.

The requirements in of legal configurations concerning labels and witnesses are easily verified. The other requirements fix the arity (the number of edges) of universal nodes $(\mathrm{V}(\forall))$ and cut nodes $(\mathrm{V}(\wedge))$. Removal and duplication in disposal and structural steps (Definitions 6.2.3 and 6.2.5) affects only the arity of existential positions and $\bot$, since by condition 4 only edges in $\mathrm{E}(\exists)$ or $\mathrm{E}(\bot)$ are targets in $(\rightarrow)$; other edges are removed or duplicated only along with their source nodes.

3. For a universal edge $\langle \mathrm{v}, a, \mathrm{w} \rangle \in \mathrm{E}(\forall)$ the following conditions hold:

   - a is not free in any formula in Γ,
   - $a \neq b$ for any other universal edge $\langle \mathrm{x}, b, \mathrm{y} \rangle \in \mathrm{E}(\forall)$,
   - $\langle \mathrm{v}, a, \mathrm{w} \rangle \rightarrow \langle \mathrm{x}, l, \mathrm{y} \rangle$ if $\langle \mathrm{x}, \mathrm{y} \rangle \in \mathrm{E}(\exists) \cup \mathrm{E}(\bot)$ and $a \in fv(l)$.

In a logical step (Definition 6.2.4) the reorganisation of the dependency traces the substitution $[t/a]$, as follows. Let the edges $\langle \mathrm{x}, t, \mathrm{y} \rangle$ and $\langle \mathrm{u}, a, \mathrm{w} \rangle$ be as in Definition 6.2.4. If the eigenvariable of an edge $\mathrm{e}_1 \in \mathrm{E}_\mathrm{A}(\forall)$ is free in t then $\mathrm{e}_1 \rightarrow_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle$; for an edge e_2 where t is to be substituted either $\mathrm{e}_2 = \langle \bot, \mathrm{c} \rangle$ or $\langle \mathrm{u}, \mathrm{w} \rangle \rightarrow_\mathrm{A} \mathrm{e}_2$, and after reduction $\mathrm{e}_1 \rightarrow_\mathrm{B} \mathrm{e}_2$. For a structural step the duplication of eigenvariables, along with vertices, ensures that their uniqueness is preserved, and that the dependency relation $(\rightarrow_\mathrm{B})$ traces their occurrences if $(\rightarrow_\mathrm{A})$ does.

5. The dependency $(\leq)$ is a partial order.

For a structural step it is immediate from Lemma 6.2.7 that $(\leq_\mathrm{B})$ is antisymmetric if $(\leq_\mathrm{A})$ is. For a logical step, let $\mathrm{v} \leq_\mathrm{B} \mathrm{v}'$ and $\mathrm{v}' \leq_\mathrm{B} \mathrm{v}$ for some $\mathrm{v} \neq \mathrm{v}'$. Lemma 6.2.6 gives three cases—(i), (ii), and (iii)—for $\mathrm{v} \leq_\mathrm{B} \mathrm{v}'$ and three—(a), (b), and (c)—for $\mathrm{v}' \leq_\mathrm{B} \mathrm{v}$.

$$\begin{array}{llllll}
\text{(i)} & \mathrm{v} \leq_\mathrm{A} \mathrm{v}' & \text{(ii)} & \mathrm{v} \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle \ \wedge\ \langle \mathrm{u}, \mathrm{w} \rangle \leq_\mathrm{A} \mathrm{v}' & \text{(iii)} & \mathrm{v} \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle \ \wedge\ \mathrm{v}' = \mathrm{c} \\
\text{(a)} & \mathrm{v}' \leq_\mathrm{A} \mathrm{v} & \text{(b)} & \mathrm{v}' \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle \ \wedge\ \langle \mathrm{u}, \mathrm{w} \rangle \leq_\mathrm{A} \mathrm{v} & \text{(c)} & \mathrm{v}' \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle \ \wedge\ \mathrm{v} = \mathrm{c}
\end{array}$$

In case (i) and (a) hold, $\leq_\mathrm{A}$ is antisymmetric; if (i) and (b) hold then $\langle \mathrm{u}, \mathrm{w} \rangle \leq_\mathrm{A} \mathrm{v}' \leq_\mathrm{A} \mathrm{v} \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle$, which means that the cut c is unsafe in F_A, a contradiction. If (i) and (c) hold then $\mathrm{v}' \neq \mathrm{c}$ since $\mathrm{v}' \neq \mathrm{v}$, and $\mathrm{v}' \notin \{\mathrm{x}, \mathrm{u}\}$ since $\mathrm{v}' \in \mathrm{V}_\mathrm{B}$. Then since $\mathrm{c} \leq_\mathrm{A} \mathrm{v}'$ also $\mathrm{y} \leq_\mathrm{A} \mathrm{v}'$ or $\mathrm{w} \leq_\mathrm{A} \mathrm{v}'$, giving the inequalities below, respectively; the former breaks antisymmetry of $(\leq_\mathrm{A})$, while the latter makes c unsafe in F_A.

$$\mathrm{y} \leq_\mathrm{A} \mathrm{v}' \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle \leq_\mathrm{A} \mathrm{y} \qquad \langle \mathrm{u}, \mathrm{w} \rangle \leq_\mathrm{A} \mathrm{w} \leq_\mathrm{A} \mathrm{v}' \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle$$

Next, the case (ii–a) is symmetric to that of (i–b), and if (ii) and (b) hold then $\langle \mathrm{u},\mathrm{w}\rangle \leq_{\mathrm{A}} \mathrm{v} \leq_{\mathrm{A}} \langle \mathrm{x},\mathrm{y}\rangle$, and c is unsafe. Similarly, in the case (ii–c) $\langle \mathrm{u},\mathrm{w}\rangle \leq_{\mathrm{A}} \mathrm{v}' \leq_{\mathrm{A}} \langle \mathrm{x},\mathrm{y}\rangle$. Finally, the cases (iii–a) and (iii–b) are symmetric to (i–c) and (ii–c) respectively, and (iii–c) requires $\mathrm{v} = \mathrm{c} = \mathrm{v}'$, a contradiction. ☐

Proposition 6.2.9. *If* $\mathrm{F}_{\mathrm{A}} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F}_{\mathrm{B}}$ *and* F_{A} *is correct, then so is* F_{B}.

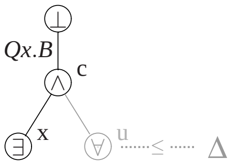
Proof. Let $\langle \bot, C, \mathrm{c}\rangle$ be the primary cut and let ς be a switching for F_{B}. The four types of reduction step will be addressed in turn. For each of the three first-order reduction steps a switching ς' for F_{A} will be given such that if $\mathit{val}(\mathrm{F}_{\mathrm{A}}, \varsigma')$ is a tautology so is $\mathit{val}(\mathrm{F}_{\mathrm{B}}, \varsigma)$.

I. Propositional steps If $\mathrm{F}_{\mathrm{A}} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F}_{\mathrm{B}}$ is a propositional step (Definition 6.2.2), there are two switchings for F_{A}, with the following values of the switched forests:

$$\begin{aligned} \varsigma' &= \varsigma \cup \{\mathrm{c} \mapsto \mathrm{L}\}; & \mathit{val}(\mathrm{F}_{\mathrm{A}}, \varsigma') &= \mathit{val}(\mathrm{F}_{\mathrm{B}}, \varsigma) \vee C \\ \varsigma'' &= \varsigma \cup \{\mathrm{c} \mapsto \mathrm{R}\}; & \mathit{val}(\mathrm{F}_{\mathrm{A}}, \varsigma'') &= \mathit{val}(\mathrm{F}_{\mathrm{B}}, \varsigma) \vee C^{\bot} \end{aligned}$$

If both values are tautologies, so is $\mathit{val}(\mathrm{F}_{\mathrm{B}}, \varsigma)$.

II. Disposal steps If $\mathrm{F}_{\mathrm{A}} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F}_{\mathrm{B}}$ is a disposal step (Definition 6.2.2), let ς' agree with ς on all cuts in F_{B}, and switch off the universal side of the primary cut c, as illustrated below, where the dependants of $\langle \mathrm{c},\mathrm{u}\rangle$ are greyed out.



Formally, choose $\varsigma' = \varsigma \cup \{\mathrm{c} \mapsto X\}$, so that $\langle \mathrm{c},\mathrm{u}\rangle \in \mathrm{E}$; then a propositional vertex $\mathrm{v} \in \mathrm{V}_{\mathrm{A}}(\mathrm{P})$ is switched on in $\mathrm{F}_{\mathrm{A}}^{\varsigma'}$ if and only if it is switched on in $\mathrm{F}_{\mathrm{B}}^{\varsigma}$. It follows immediately that $\mathit{val}(\mathrm{F}_{\mathrm{A}}, \varsigma')$ is a tautology if and only if $\mathit{val}(\mathrm{F}_{\mathrm{B}}, \varsigma)$ is.

III. Logical steps If $\mathrm{F}_{\mathrm{A}} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F}_{\mathrm{B}}$ is a logical reduction step, let the five edges

$$\langle \bot, Qx.B, \mathrm{c}\rangle \qquad \langle \mathrm{c}, U, \mathrm{u}\rangle \qquad \langle \mathrm{c}, X, \mathrm{x}\rangle \qquad \langle \mathrm{u}, a, \mathrm{w}\rangle \qquad \langle \mathrm{x}, t, \mathrm{y}\rangle$$

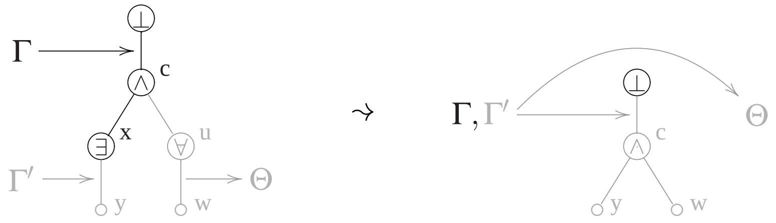
be as in Definition 6.2.4. There are two cases to consider.

1. Suppose the cut c is not switched off by ς, i.e. no $e \leq_B c$ is in E^{ς}_{B}. Let $\varsigma' = \varsigma$, let $v \in V_A$ be a propositional vertex, and assume that $e \leq_B v$ for some $e \in E^{\varsigma}_{B}$. For $e \leq_B v$ Lemma 6.2.6 gives three options; however, two are ruled out because $e \leq_A \langle x, y\rangle$ would imply $e \leq_B c$, contrary to assumption. The remaining option gives $e \leq_A v$; since is immediate that also $e \in E^{\varsigma'}_{A}$, it then follows that $\mathit{val}(F_B, \varsigma)$ is a tautology if $\mathit{val}(F_A, \varsigma')$ is.

2. Suppose c is switched off by some $e_0 \leq_B c$ in E^{ς}_{B}. Again let $v \in V_A$ be a propositional vertex and assume that $e \leq_B v$ for some $e \in E^{\varsigma}_{B}$, but this time let ς' agree with ς on all cuts except c, where it switches on the existential branch,

   $$\varsigma' \;=\; \{v \mapsto \varsigma(v) \mid v \in V_A(\wedge) \wedge v \neq c\} \;\cup\; \{c \mapsto X\}\ ,$$

   so that $\langle c, u\rangle \in E^{\varsigma'}_{A}$. The idea of the proof is that also in $F^{\varsigma'}_{A}$ all propositional nodes depending on c are switched off, since either $e_0 \leq_A \langle \bot, c\rangle$ or $e_0 \leq_A \langle x, y\rangle$, the latter of which is illustrated below.

   It will be shown that v depends on e while $e \in E^{\varsigma'}_{A}$, or that v depends on e_0 or $\langle c, u\rangle$, both of which are in $E^{\varsigma'}_{A}$. It then follows that v is switched off in $F^{\varsigma'}_{A}$, and since $V_A(\mathrm{P}) = V_B(\mathrm{P})$, that $\mathit{val}(F_B, \varsigma)$ is a tautology if $\mathit{val}(F_A, \varsigma')$ is. Firstly, for $e \leq_B v$ Lemma 6.2.6 gives three options, one of which is ruled out because v is propositional, so that $v \neq c$. This means that either

   $$e \leq_A v \qquad \text{or} \qquad e \leq_A \langle x, y\rangle \text{ and } \langle u, w\rangle \leq_A v\ .$$

   In the latter case, $\langle c, u\rangle \leq_A v$; also the former is immediate if $e \in E^{\varsigma'}_{A}$, which is the case unless $e = \langle c, x\rangle$, since ς' and ς agree on all cuts other than c. Then since $v \neq x$ (v is propositional) and $\langle c, x\rangle$ is not a source in $(\rightarrow_A)$, it follows that $y \leq_A v$. For $e_0 \leq_B c$, Lemma 6.2.6 gives three options, but since $\langle u, w\rangle \leq_A c$ would violate the antisymmetry of $(\leq_A)$ only two remain:

   $$e_0 \leq_A c \qquad \text{or} \qquad e_0 \leq_A \langle x, y\rangle\ .$$

   In both cases, $e_0 \leq_A y \leq_A v$, and v is switched off in $F^{\varsigma'}_{A}$.

IV. Structural steps If $\mathrm{F_A} \overset{\mathrm{c,y}}{\rightsquigarrow} \mathrm{F_B}$ is a structural reduction step, let the existential edges $\langle \mathrm{x},\mathrm{y}\rangle$ and $\langle \mathrm{x},\mathrm{y}_1\rangle,\ldots,\langle \mathrm{x},\mathrm{y}_n\rangle$ be as in Definition 6.2.5, as well as the set X and the substitution maps ρ and σ. Three cases are distinguished, depending on the choice the switching ς for $\mathrm{F_B}$ makes on c and c'; the second and third case overlap.

1. If ς on both c and c' selects the existential branch $\varsigma(\mathrm{c}) = \varsigma(\mathrm{c}') = X$, let ς' on $\mathrm{F_A}$ agree with ς:
$$\varsigma' = \{\mathrm{v} \mapsto \varsigma(\mathrm{v}) \mid \mathrm{v} \in \mathrm{V_A}(\wedge)\}\ .$$

2. If ς on c selects the universal branch, $\varsigma(\mathrm{c}) = U$, again let ς and ς' agree:
$$\varsigma' = \{\mathrm{v} \mapsto \varsigma(\mathrm{v}) \mid \mathrm{v} \in \mathrm{V_A}(\wedge)\}\ .$$

3. If ς on c' selects the universal branch, $\varsigma(\mathrm{c}') = U$, let $\varsigma' : \mathrm{V_A}(\wedge)$ be as follows:
$$\varsigma'(\mathrm{v}) = \begin{cases} \varsigma(\mathrm{v}') & \text{if } \mathrm{v}' \in \mathrm{V_B} \\ \varsigma(\mathrm{v}) & \text{otherwise.} \end{cases}$$

Let $\mathrm{v} \in \mathrm{V_A}(\mathrm{P})$ be a propositional node switched on by ς' in $\mathrm{F_A}$. First it will be shown for cases 1 and 2 that v is switched on by ς in $\mathrm{F_B}$. Since $\mathrm{V_A}(\mathrm{P}) \subseteq \mathrm{V_B}(\mathrm{P})$ this requires only the following:

$$\exists \mathrm{e}_1 \in \mathrm{E}^{\varsigma}_{\mathrm{B}}.\ \mathrm{e}_1 \leq_{\mathrm{B}} \mathrm{v} \quad \Rightarrow \quad \exists \mathrm{e}_2 \in \mathrm{E}^{\varsigma'}_{\mathrm{A}}.\ \mathrm{e}_2 \leq_{\mathrm{A}} \mathrm{v}\ .$$

The edge e_1 is either an original one or a duplicated one. If it is original, then $\mathrm{e}_1 \in \mathrm{E}^{\varsigma'}_{\mathrm{A}}$ and $\mathrm{e}_1 \leq_{\mathrm{A}} \mathrm{v}$. If it is a duplicate, then by Lemma 6.2.7 it can only be $\langle \mathrm{c}',\mathrm{x}'\rangle$, since v is an original node. In case 1, $\langle \mathrm{c},\mathrm{x}\rangle \in \mathrm{E}^{\varsigma}_{\mathrm{A}}$ and $\langle \mathrm{c},\mathrm{x}\rangle \leq_{\mathrm{A}} \mathrm{v}$; in case 2, $\langle \mathrm{c}',\mathrm{x}'\rangle \notin \mathrm{E}^{\varsigma}_{\mathrm{B}}$.

For case 3 it will be shown, for every propositional vertex v switched on by ς' in $\mathrm{F_A}$, that ς in $\mathrm{F_B}$ switches on v' if $\mathrm{u} \leq_{\mathrm{A}} \mathrm{v}$ and v otherwise. If v does not depend on u it is not duplicated, and the argument is the same as above. Otherwise, if $\mathrm{u} \leq_{\mathrm{A}} \mathrm{v}$, let $\mathrm{e}_1 \leq_{\mathrm{B}} \mathrm{v}'$ for some $\mathrm{e}_1 \in \mathrm{E}^{\varsigma}_{\mathrm{B}}$. Then these are the possibilities.

- $\mathrm{e}_1 \in \mathrm{E_A}$. Then $\mathrm{e}_1 \leq_{\mathrm{A}} \mathrm{v}$, and, by Lemma 6.2.7, $\mathrm{u} \not\leq \mathrm{e}_1$. Thus, there is no duplicate of the source of e_1, and ς' agrees with ς in such cases, which gives $\mathrm{e}_1 \in \mathrm{E}^{\varsigma'}_{\mathrm{A}}$, a contradiction.

- $\mathrm{e}_1 = \mathrm{e}'$ for some $\mathrm{e} \in \mathrm{E_A}$. Then $\mathrm{e} \in \mathrm{E}^{\varsigma'}_{\mathrm{A}}$ and $\mathrm{e} \leq_{\mathrm{A}} \mathrm{v}$, a contradiction.

Since if $u \not\leq_A v$ the label of v contains no eigenvariables substituted by σ, the label is unaffected by it: $lab(v)[\sigma] = lab(v)$. Thus, for every propositional vertex v with label $lab_A(v)$ switched on by ς' in F_A, there is a vertex switched on by ς in F_B with label $lab_A(v)[\sigma]$, which is v' if $u \leq_A v$ and v otherwise. Then $val(F_B, \varsigma)$ is a tautology if $val(F_A, \varsigma')$ is. □

6.3 The universal counterexample

Figure 6.3 displays the *universal counterexample*, a proof forest consisting solely of two cuts. It may be obtained by composing the example in Figure 5.3 with two instances of the proof forest for the drinker's formula in Figure 5.1 (in this composition, the universal counterexample would be accompanied by a context of two additional trees). Labels and witnesses are omitted; naturally, in isolation, there is no choice of labels that makes the universal counterexample correct, since it is a proof forest for the empty sequent.

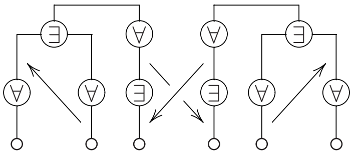


Figure 6.3: The universal counterexample

The universal counterexample is reduced in Figure 6.4 on page 168, until a single, unsafe cut remains. Throughout the reduction, the dependencies that contribute to the loss of safety are drawn in black, while other dependencies are drawn in grey. In places several reduction steps have been taken at once; such multi-steps are indicated by $(\leadsto^*)$. The example shows the following.

Proposition 6.3.1. *The reduction relation* $(\leadsto)$ *is not strongly normalising. This holds even for the class of forests that arise from cut-free forests by composition with cut.*

In addition to creating cuts that are unsafe, the universal counterexample may exhibit infinite reduction paths. An example of such a *reduction cycle* is shown in Figure 6.5.[1] The diagram at the bottom right of the reduction of the universal counterexample in Figure 6.4 gives rise to a reduction cycle similar to the one in Figure 6.5.

---

[1] The observation that reduction cycles may exist without passing through unsafe cuts, as happens in Figure 6.1, is due to Richard McKinley, via private communication.

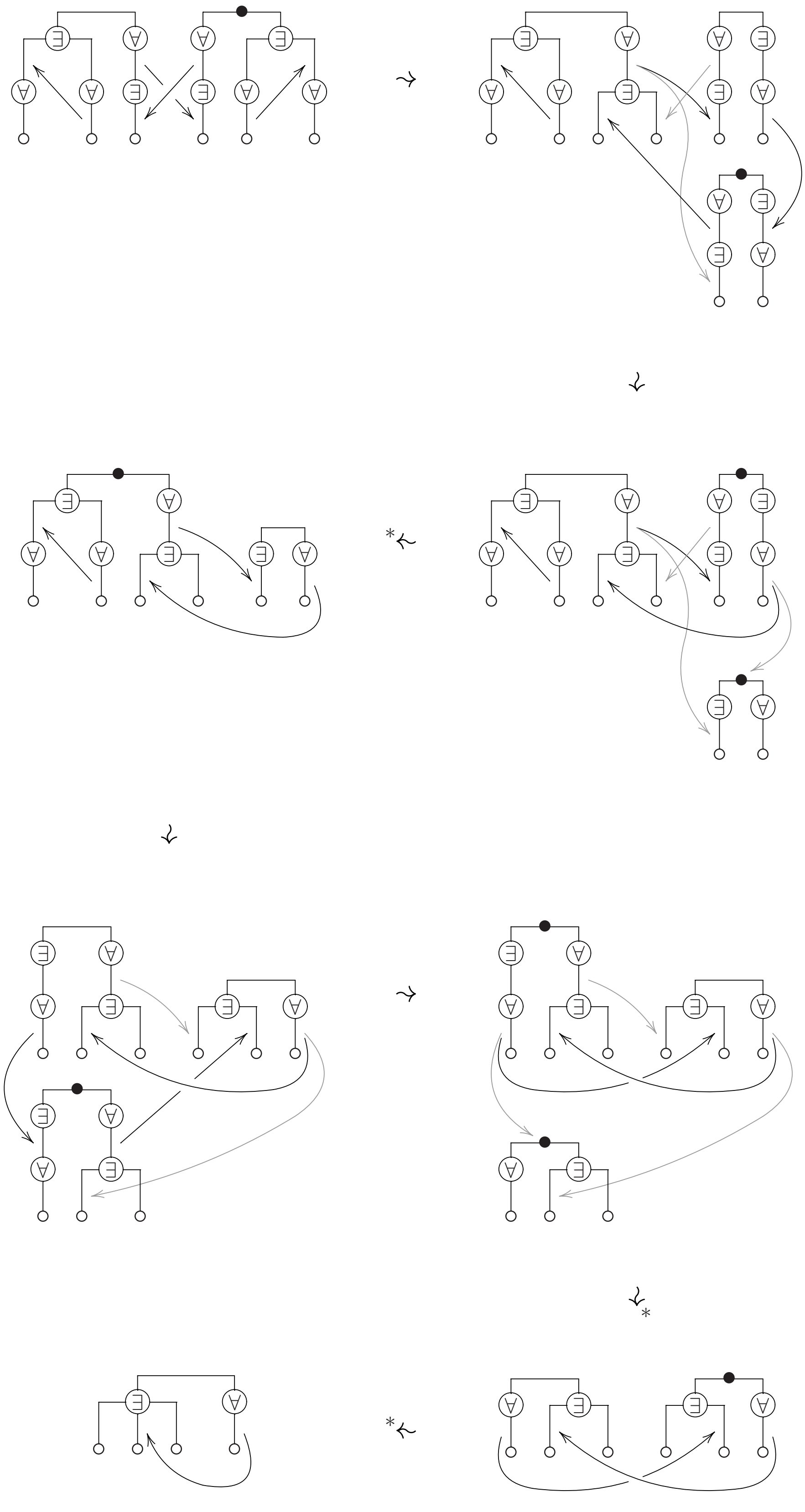


Figure 6.4: Reducing the universal counterexample to an unsafe forest

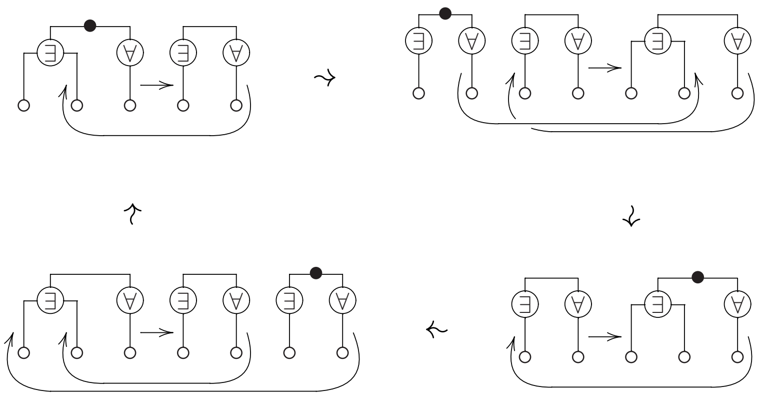
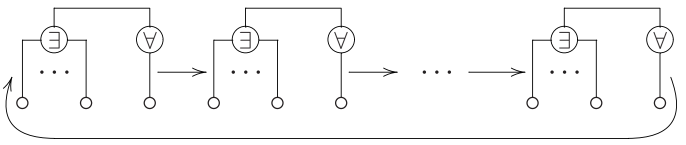


Figure 6.5: A reduction cycle

On weak normalisation

The universal counterexample does have normalising reduction paths, one of which is displayed in Figure 6.6 on page 170. The reduction path first reduces the cut on the right in its entirety, before reducing that on the left. Any such path, and only such paths, where one of the cuts is reduced before the other, are normalising. Weak normalisation of ($\rightsquigarrow$) is thus not ruled out—at least for proof forests that arise by composition.

Figure 6.7 on page 171 explores where in the reduction of the universal counterexample weak normalisation is lost. The left column shows the first four steps of the normalising reduction path of Figure 6.6, while the three other reduction steps ($\overset{\mathrm{c}}{\rightsquigarrow}$) each produce a proof forest that does not normalise. What these three proof forests have in common is a configuration of the kind below, where several *distinct* cuts are 'chained' together into a circle by dependencies between their branches (this will be referred to as a *circle of cuts*). Such a configuration is not weakly normalising, and although a similar one already exists in the universal counterexample itself, the crucial difference is that there, the circle passes through the same cuts twice.



While the universal counterexample may arise (in context) from the translation of a sequent proof, the unsafe proof forests that it reduces to, and the configuration above, do not. The important observation to be made here is that in Figure 6.7, the five proof forests in the left column are translations of sequent proofs, while the three proof

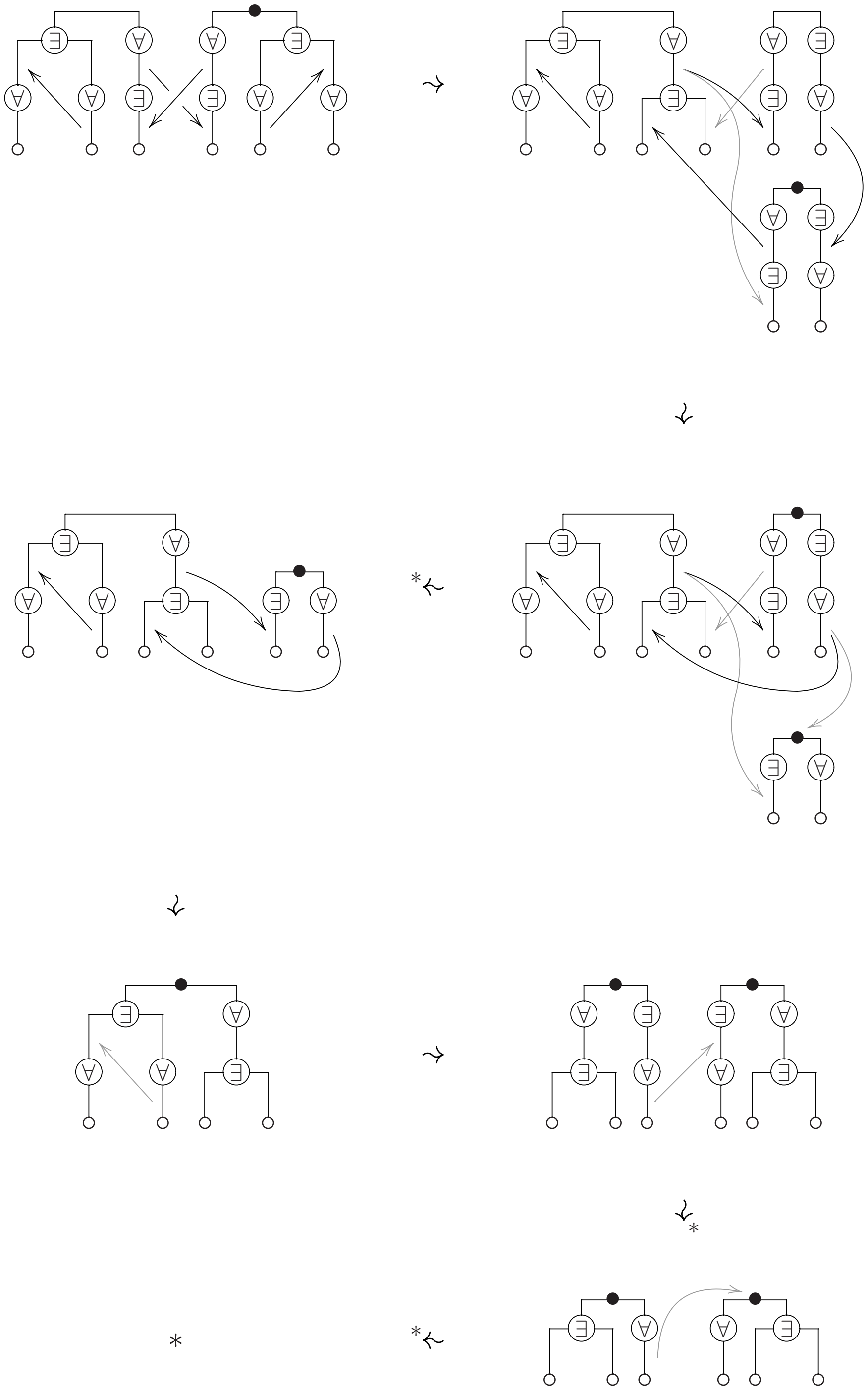


Figure 6.6: Normalising the universal counterexample

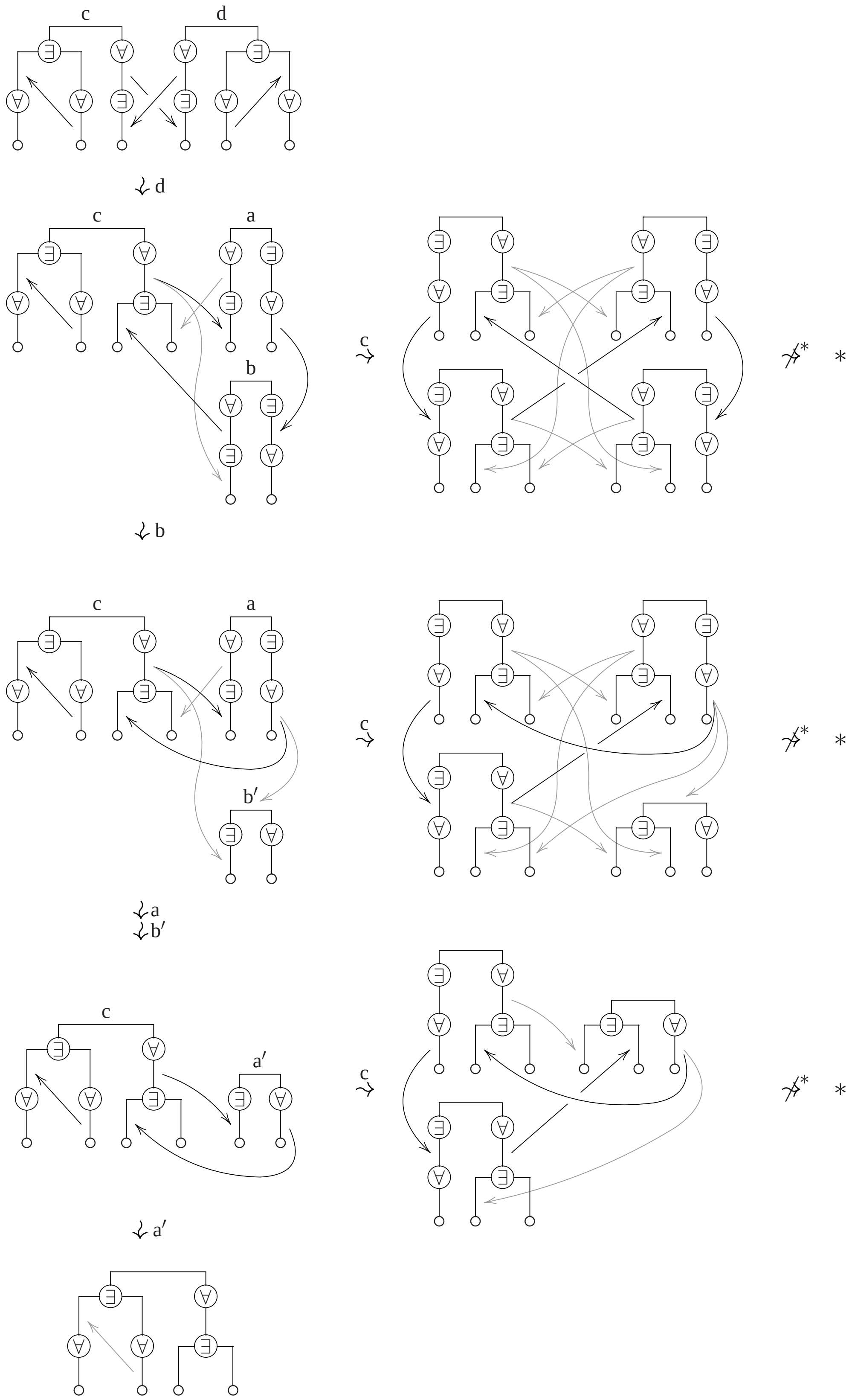
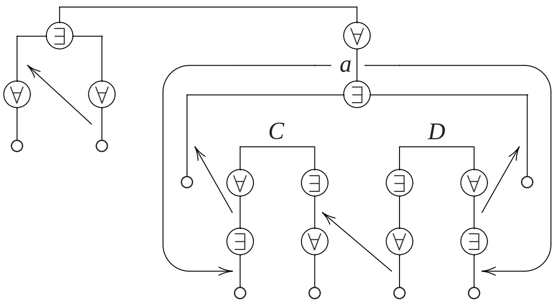


Figure 6.7: Losing weak normalisation

forests in the right column are not: the three reduction steps $(\overset{\mathsf{c}}{\rightsquigarrow})$ that cause the loss of weak normalisation, are also precisely the ones that take the proof forest outside the image of the sequent calculus translation. More specifically, the vertical steps in the left column *simulate* reduction in the sequent calculus, up to permutations, while the horizontal steps between the columns do not. The mechanism by which this happens is as follows. In each of the three horizontal steps $(\overset{\mathsf{c}}{\rightsquigarrow})$, the collection of dependants that is duplicated is strictly smaller than the subproof that would be duplicated in the corresponding reduction step in sequent calculus, for the reason explored at the end of Section 5.5. To illustrate this, the universal counterexample after one reduction step is depicted in Figure 6.8 in a 'planar' fashion, suggestive of the topology of a corresponding sequent proof, part of which is depicted below the proof forest. In the partial sequent proof, the contraction cannot permute above either cut, since its two premises originate in different subproofs, one in Π and one in Π''. In which subproofs the occurrences of $\exists y.A(a)$ must occur is determined by the dependencies in the proof forest.



$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\Pi\\ \vdots\\ \vdash \exists y.A(a), C\end{array} \qquad \dfrac{\begin{array}{c}\Pi'\\ \vdots\\ \vdash C^{\perp}, D\end{array} \qquad \begin{array}{c}\Pi''\\ \vdots\\ \vdash D^{\perp}, \exists y.A(a)\end{array}}{\vdash C^{\perp}, \exists y.A(a)}\,\text{Cut}}{\vdash \exists y.A(a), \exists y.A(a)}\,\text{Cut}}{\vdash \quad \exists y.A(a)}\,\text{C}\exists}{\vdash \quad \forall x \exists y.A(x)}\,\forall\text{R}$$

Figure 6.8: A subproof larger than the corresponding set of dependants

In the remainder of this chapter, a first solution to the problem of weak normalisation will be presented. In Section 6.4 the reduction relation is modified by adding a *pruning* operation, to make unsafe proof forests safe, and grouping reduction steps on the same cut, to avoid reduction cycles as in Figure 6.5. A second solution is presented in Chapter 7. It is based on a formalisation of the above observation, that the problematic reduction steps that cause the loss of weak normalisation are those that duplicate dependants which don't correspond exactly to any subproof in a corresponding sequent proof. It will be shown that it is possible to avoid such reduction steps, to obtain weak normalisation for $(\rightsquigarrow)$.

Reducing the universal counterexample is also non-confluent, yielding both normal forms and unsafe forests. Non-confluence will be explored in more detail in Section 7.4.

6.4 The modified reduction relation

The two main obstacles to obtaining weak normalisation are the occurrence of unsafe proof forests in reductions, and cyclic reduction paths of the kind shown in Figure 6.5. Both will be addressed in turn, below, resulting in a modified version $(\mathrel{\approx\!\!>})$ of the reduction relation $(\rightsquigarrow)$, that will be shown to be weakly normalising, and conjectured to be strongly normalising.

The notion of safety, defined in Section 6.2, was motivated by the observation that dependencies between the two branches of a cut may result in cuts that cannot be reduced, while such dependencies may never arise from composition or translation. This motivation explains why the concept of safety is needed. That it is also a natural concept, closely related to correctness, becomes clear from the game-theoretic idea of a cut consisting of two consecutive moves, explored in Section 5.3. In that view, the second of the two moves is a binary choice by ∀belard, who chooses exactly one branch of the cut in any particular game, after which the dependants of the other branch become unreachable. The observation that a vertex depending on both sides is then unreachable in any game, yields a simple solution to the problem of unsafe cuts: the offending vertices may be removed from the proof forest altogether, in an operation called *pruning*. Formalising this idea starts with the following definition.

Definition 6.4.1 (Conflict)**.** The symmetric *conflict* relation $(\#)$ holds between nodes that depend on different branches of the same cut:

$$v_1 \# v_2 \quad \overset{\Delta}{\iff} \quad \exists \langle c,u \rangle, \langle c,w \rangle \in E(\wedge).\ u \neq w \ \wedge \ u \leq v_1 \ \wedge \ w \leq v_2 \ .$$

The conflict relation indicates, precisely, when two vertices are never both reachable in any particular game. This gives an alternative approach to defining correctness.

Proposition 6.4.2. *In a proof forest* F *the maximal conflict-free subsets of* V *are precisely the sets of vertices switched on by the switchings of* F*. The values of* F *are the disjunctions over the labels of the propositional vertices in its maximal conflict-free subsets of* V.

Proof. For each cut c with children u and w, a maximal conflict-free set must contain exactly one of u and w (or a vertex that conflicts with both). If it contains u, it cannot contain the dependants of w, which are all in conflict with u; this corresponds to a switching that switches off w. The details are straightforward. Note, however, that it would be incorrect to use the slightly different characterisation of correctness as a tautology requirement over the maximal conflict-free sets of propositional variables. The reason is that to account for a switching that selects an existential leaf, such vertices must be considered in the maximality requirement. ☐

In addition, safety can be characterised as follows.

Proposition 6.4.3. *A proof forest is safe if and only if* $(\#)$ *is irreflexive.*

An interesting observation is that safe proof forests are *event structures* [96]. An event structure $(\mathrm{V}, \leq, \#)$ consists of:

- a set of *events* V;
- a partial *dependency* order $(\leq)$ on V, such that for any event v the down-closure $\{\mathrm{x} \mid \mathrm{x} \leq \mathrm{v}\}$ is finite;
- a symmetric, irreflexive *conflict* relation $(\#)$ on V, satisfying

$$\mathrm{u} \# \mathrm{v} \leq \mathrm{w} \;\Rightarrow\; \mathrm{u} \# \mathrm{w} \,.$$

Event structures model concurrent computation as a collection of events V, with the relation $(\leq)$ representing their causal dependency, and the conflict relation $(\#)$ expressing the incompatibility of certain events. It is easily verified that the vertices, dependency, and conflict relation of a proof forest F form an event structure $(\mathrm{V}, \leq, \#)$.

Since the correctness condition is based on which positions ∃loise can reach in any particular game, safety can be enforced by removing self-conflicting vertices.

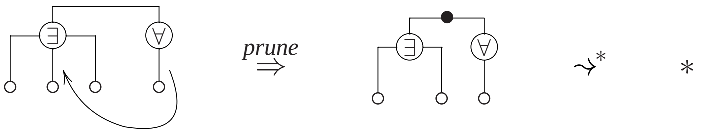
Definition 6.4.4 (Pruning)**.** The *pruning* function removes all self-conflicting nodes from a proof forest: $\mathit{prune}(\mathrm{F}) = \mathrm{F}|_X$, where $X = \{\mathrm{v} \in \mathrm{V} \mid \neg(\mathrm{v} \# \mathrm{v})\}$.

A pruned proof forest is by definition safe. Below, it is established that pruning a correct proof forest yields a correct subforest.

Proposition 6.4.5. *Pruning preserves the axioms of proof forests, and correctness.*

Proof. Most conditions of Definition 5.4.3 are preserved straightforwardly, though it should be noted that the branching condition on universal and cut vertices is preserved because their edges are never targets in $(\rightarrow)$; if such an edge $\langle \mathrm{u}, \mathrm{v}\rangle \in \mathrm{E}(\forall) \cup \mathrm{E}(\wedge)$ is removed, so is u, since v # v only if u # u. For correctness, the maximal, conflict-free subsets of vertices in F and in *prune*(F) are identical, since exactly the vertices that show up in no such subset in F are removed by pruning. It then follows from proposition 6.4.2 that the values of *prune*(F) are precisely those of F, and that pruning preserves correctness. ☐

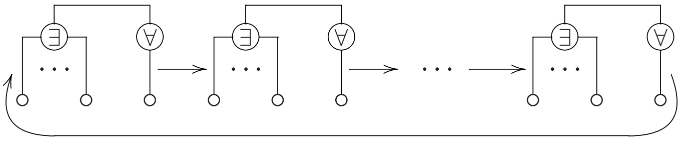
The final, unsafe cut in the reduction of the universal counterexample in Figure 6.4 is pruned, and then reduced, as follows.



Compound reduction steps

The second problem is that of infinite reduction paths of the kind shown in Figure 6.5, where cuts with mutually dependent branches can duplicate each other's existential branches. This problem is addressed by grouping reduction steps together in a *compound reduction step*, written $(\rightsquigarrow)$, which performs all the possible structural reduction steps on a given cut, one after another, and reduces the newly formed logical cuts, by one step each. A compound reduction step is depicted in Figure 6.9—the illustration omits the details of renaming nodes and eigenvariables in the contexts Π_1 through Π_n, which are the duplicates of Π.

The problem of infinite reduction paths on a configuration of the kind below, where cuts are chained together in a circle by dependencies between their branches, is then resolved as follows. Using compound reduction steps the number of cuts in the circle will strictly reduce, until only one, unsafe, cut remains, which can then be pruned.



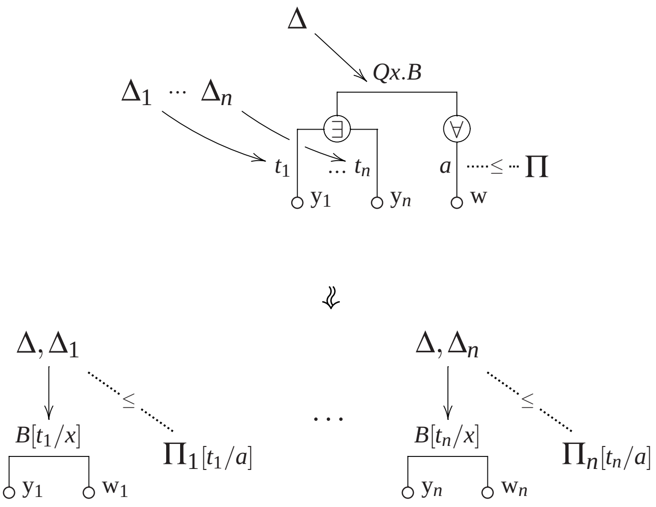


Figure 6.9: A compound reduction step

Compound reduction steps have the following good properties. Since they consist of a sequence of reduction steps in ($\rightsquigarrow$), plus pruning, compound steps inherit the preservation properties of ($\rightsquigarrow$), e.g. the preservation of the axioms of proof forests and of correctness. Moreover, the order in which the reduction steps in ($\rightsquigarrow$) that make up a compound reduction step are performed, is irrelevant: for a safe cut c in a proof forest F_A, there will be exactly one compound reduction step $F_A \overset{c}{\Rrightarrow} F_B$ (in the sense that if also $F_A \overset{c}{\Rrightarrow} F_C$, then $F_C = F_B$). As a consequence, the result of a compound reduction step can be defined directly, as is done in Definition 6.4.7. For these reasons compound steps may be viewed as a proper reduction relation—rather than a (local) strategy for ($\rightsquigarrow$)—consisting of one, uniform reduction rule for first-order cuts, and one for propositional cuts.

To establish these properties, compound reduction steps will be defined in two ways, which are then proven equal. The first will define the relation ($\Rrightarrow^-$) as a series of steps in ($\rightsquigarrow$); the second will define a relation ($\Rrightarrow^!$) that computes the outcome of a compound step directly. Weak normalisation will be shown for the relation ($\Rrightarrow$), which adds a concluding pruning step.

Definition 6.4.6 (Compound reduction steps)**.** A *compound reduction step* $F_A \overset{c}{\Rrightarrow} F_B$ on a safe cut c in a proof forest F_A, is inductively defined as follows.

The relation ($\Rrightarrow^-$) is the smallest such that $F_A \overset{c}{\Rrightarrow}{}^- F_D$ if:

- $F_A \overset{c}{\rightsquigarrow} F_D$ by a propositional step (Definition 6.2.2), or a disposal step (Definition 6.2.3), or a logical step (Definition 6.2.4); or

- $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_B}$ by a structural step (Definition 6.2.5), where c' is the duplicate of c,
  $\mathrm{F_B} \overset{\mathrm{c}'}{\rightsquigarrow} \mathrm{F_C}$ by a logical step, and
  $\mathrm{F_C} \overset{\mathrm{c}}{\approx\!\!>}{}^{-} \mathrm{F_D}$.

If $\mathrm{F_A} \overset{c}{\approx\!\!>}{}^{-} \mathrm{F_B}$ then $\mathrm{F_A} \overset{c}{\approx\!\!>} \mathit{prune}(\mathrm{F_B})$.

To illustrate the definition, for a cut c with existential edges $\langle \mathrm{x}, \mathrm{y}_1\rangle, \ldots, \langle \mathrm{x}, \mathrm{y}_n\rangle$ a compound reduction step consists of the following series of alternating structural and logical reduction steps, plus a pruning step at the end in the case of $(\approx\!\!>)$:

$$\mathrm{F}_1 \overset{\mathrm{c,y}_1}{\rightsquigarrow} \mathrm{F}_2 \overset{\mathrm{c}_1}{\rightsquigarrow} \mathrm{F}_3 \overset{\mathrm{c,y}_2}{\rightsquigarrow} \mathrm{F}_4 \overset{\mathrm{c}_2}{\rightsquigarrow} \ldots \overset{\mathrm{c}_{n-2}}{\rightsquigarrow} \mathrm{F_{2n-3}} \overset{\mathrm{c,y}_{n-1}}{\rightsquigarrow} \mathrm{F_{2n-2}} \overset{\mathrm{c}_{n-1}}{\rightsquigarrow} \mathrm{F_{2n-1}} \overset{\mathrm{c}}{\rightsquigarrow} \mathrm{F_{2n}}$$

where in each structural step $(\overset{\mathrm{c,y}_i}{\rightsquigarrow})$ the renaming substitution σ of Definition 6.2.5 assigns fresh vertices v_i (rather than v'). Note that there is one fewer structural step than there are logical steps in this sequence, since after $n-1$ structural steps the cut c will have only one existential branch remaining.

The second, direct, definition of compound reduction steps is as follows.

Definition 6.4.7. For a safe first-order cut c in a proof forest $\mathrm{F_A}$, the reduction step $\mathrm{F_A} \overset{\mathrm{c}}{\approx\!\!>}{}^{!} \mathrm{F_T}$ yields the pre-proof forest $\mathrm{F_T}$, as follows. Let c have edges $\langle \mathrm{c}, U, \mathrm{u}\rangle$ and $\langle \mathrm{c}, X, \mathrm{x}\rangle$, with universal edge $\langle \mathrm{u}, a, \mathrm{w}\rangle$ and existential edges $\langle \mathrm{x}, t_1, \mathrm{y}_1\rangle, \ldots, \langle \mathrm{x}, t_n, \mathrm{y}_n\rangle$. Let $\mathrm{F_R}$ be as follows.

$$\mathrm{F_R} \;=\; \Big(\mathrm{F_A} \cup \bigcup_{1\le i\le n} (\mathrm{F_A}|_X\,[\rho_i]\,[\sigma_i]\,)\Big)\;[\tau]$$

where for all i ($1 \le i \le n$), with all v_i and a_i fresh w.r.t. $\mathrm{F_A}$ and each other,

$$\begin{aligned}
X &= \{\mathrm{v} \in \mathrm{V_A} \mid \mathrm{x} \not<_{\mathrm{A}} \mathrm{v}\}\\
\rho_i &= \{\mathrm{v} \mapsto \mathrm{v}_i \mid \mathrm{v} \in \{\mathrm{c}, \mathrm{x}\} \;\vee\; \mathrm{u} \le_{\mathrm{A}} \mathrm{v}\}\\
\sigma_i &= \{a \mapsto a_i \mid \langle \mathrm{v}, a, \mathrm{w}\rangle \in \mathrm{E_A}(\forall) \;\wedge\; \mathrm{u} \le_{\mathrm{A}} \langle \mathrm{v}, \mathrm{w}\rangle\}\\
\tau &= \{\langle \mathrm{x}, \mathrm{y}_i\rangle \mapsto \langle \mathrm{x}_i, \mathrm{y}_i\rangle \mid 1 \le i < n\}\;.
\end{aligned}$$

Let $\mathrm{F_S} = \mathrm{F_R}|_Y$ where $Y = \{\mathrm{v} \in \mathrm{V_R} \mid \mathrm{c} \not\le_{\mathrm{R}} \mathrm{v}\}$. Let $\mathrm{F_T}$ be as follows. Firstly, $\mathrm{V_T}$ is $\mathrm{V_S}$ minus the vertices x_i and u_i for all $i \le n$. Secondly, $\mathit{lab}_{\mathrm{T}}(\mathrm{c}_i) = B \wedge B^{\perp}[t_i/x]$ and, for other vertices, $\mathit{lab}_{\mathrm{T}}(\mathrm{v}) = \mathit{lab}_{\mathrm{S}}(\mathrm{v})[t_i/a_i]$. Next, $\mathrm{E_T}$ is obtained from $\mathrm{E_S}$ by replacing for every $i \le n$ the edges

$$\langle \perp, Qx.B, \mathrm{c}_i\rangle \qquad \langle \mathrm{c}_i, U, \mathrm{u}_i\rangle \qquad \langle \mathrm{c}_i, X, \mathrm{x}_i\rangle \qquad \langle \mathrm{u}_i, a_i, \mathrm{w}_i\rangle \qquad \langle \mathrm{x}_i, t_i, \mathrm{y}_i\rangle$$

with the edges

$$\langle \bot, B[t_i/x], \mathrm{c}_i \rangle \qquad \langle \mathrm{c}_i, U, \mathrm{w}_i \rangle \qquad \langle \mathrm{c}_i, X, \mathrm{y}_i \rangle\ ,$$

and any other edge $\langle \mathrm{v}, Y, \mathrm{z} \rangle$ with $\langle \mathrm{v}, Y[t/a], \mathrm{z} \rangle$. Finally, $(\rightarrow_{\mathrm{T}})$ is the smallest relation on $\mathrm{E_T}$ such that

$$\begin{array}{lll} \mathrm{e}_1 \rightarrow_{\mathrm{T}} \mathrm{e}_2 & \text{if} & \mathrm{e}_1 \rightarrow_{\mathrm{S}} \mathrm{e}_2, \quad \text{or} \\ & & \mathrm{e}_1 \rightarrow_{\mathrm{S}} \langle \bot, \mathrm{c}_i \rangle \text{ and } \langle \mathrm{u}_i, \mathrm{w}_i \rangle \rightarrow_{\mathrm{S}} \mathrm{e}_2 \text{ for some } i, \quad \text{or} \\ & & \mathrm{e}_1 \rightarrow_{\mathrm{S}} \langle \mathrm{x}_i, \mathrm{y}_i \rangle \text{ and } \langle \mathrm{u}_i, \mathrm{w}_i \rangle \rightarrow_{\mathrm{S}} \mathrm{e}_2 \text{ for some } i, \quad \text{or} \\ & & \mathrm{e}_1 \rightarrow_{\mathrm{S}} \langle \mathrm{x}_i, \mathrm{y}_i \rangle \text{ and } \mathrm{e}_2 = \langle \bot, \mathrm{c}_i \rangle\ . \end{array}$$

The above definition, which combines features of the definitions of disposal, logical, and structural steps (Definitions 6.2.3, 6.2.4, and 6.2.5), proceeds as follows. The proof forest $\mathrm{F_R}$ results from duplicating the cut c and its dependants on the universal side as many times as there are existential branches of c, and moving each existential branch to its own copy. The cut c, with no existential branches left, is removed in the proof forest $\mathrm{F_S}$, in the way it would be in a disposal step. The proof forest $\mathrm{F_S}$ is what would be the result of applying all possible structural steps, or one disposal step, to the proof forest $\mathrm{F_A}$—plus one renaming substitution ρ_i and one σ_i, where $\langle \mathrm{x}, \mathrm{y}_i \rangle$ is the last existential edge remaining on the cut c. Then the proof forest $\mathrm{F_T}$ is the proof forest $\mathrm{F_S}$ after all duplicated cuts c_i have been reduced by a logical step.

That $(\rightsquigarrow^{-})$ and $(\rightsquigarrow^{!})$ are the same reduction relation is established below.

Proposition 6.4.8. *For a proof forest* $\mathrm{F_A}$ *with a safe first-order cut* c, *if* $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow}^{-} \mathrm{F_D}$ *and* $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow}^{!} \mathrm{F_T}$ *then* $\mathrm{F_D} = \mathrm{F_T}$ *(up to the naming of vertices and eigenvariables).*

Proof. Let $\mathrm{F_R}$, $\mathrm{F_S}$ and $\mathrm{F_T}$ be as in Definition 6.4.7. A main observation is that the set X in Definition 6.4.7 contains all the dependants of u: since c is safe, $\mathrm{u} \leq \mathrm{v}$ implies $\mathrm{x} \not\leq \mathrm{v}$. There is the following statement.

1. For v in $\mathrm{F_A}$, the vertex v_i is in $\mathrm{F_S}$ if and only if $1 \leq i \leq n$ and either $\mathrm{u} \leq_{\mathrm{A}} \mathrm{v}$ or $\mathrm{v} \in \{\mathrm{c}, \mathrm{x}\}$.

A second observation is that, since τ moves each existential edge $\langle \mathrm{x}, \mathrm{y}_i \rangle$ from x to the vertex x_i in $\mathrm{F_R}$, the cut c in $\mathrm{F_R}$ has no existential branches. Removing the cut c, in $\mathrm{F_S}$, removes also the dependants of u, but no dependants of x in $\mathrm{F_A}$. There is the following statement.

2. For v in $\mathrm{F_A}$, the vertex v is also in $\mathrm{F_S}$ if and only if v is different from x and c, and not a dependant of u (i.e., $\mathrm{u} \not\leq_{\mathrm{A}} \mathrm{v}$).

To show $F_D = F_T$, firstly, if c has no existential branches, $F_A \overset{c}{\rightsquigarrow}^{-} F_D$ consists of a single disposal step. It is easily verified that $F_R = F_A$ and $F_S = F_T = F_B$.

Secondly, if c is a logical cut with existential branch $\langle x, y_1 \rangle$, then $F_A \overset{c}{\rightsquigarrow}^{-} F_D$ consists of a single logical step. Observe that by **1.** and **2.** above, F_S is just $F_S = F_A[\rho_1][\sigma_1]$. Then $F_T = F_D[\rho_1][\sigma_1]$.

For the third case, of a cut c with two or more existential branches, the reduction step $F_A \overset{c}{\rightsquigarrow}^{-} F_D$ consists of a structural step $F_A \overset{c}{\rightsquigarrow} F_B$, a logical step $F_A \overset{c'}{\rightsquigarrow} F_C$, and a compound step $F_C \overset{c}{\rightsquigarrow}^{-} F_D$. It will be shown that $V_D = V_T$ (up to the same simple renaming as in the logical step above). Along the way, it is established that the cuts c' and c are safe in F_B and F_C respectively; this shows that there is at least one proof forest F_D such that $F_A \overset{c}{\rightsquigarrow}^{-} F_D$.

Let the existential edges of c be $\langle x, y_1 \rangle, \ldots, \langle x, y_n \rangle$, and let $\langle x, y_j \rangle$ be the primary branch of the structural step $F_A \overset{c, y_j}{\rightsquigarrow} F_B$. To align the notation of the structural step with that of $F_A \rightsquigarrow^{!} F_T$, let F_B be the following proof forest, where X, ρ_j, and σ_j are as in Definition 6.4.7.

$$F_B \;=\; (F_A \cup F_A|_X[\rho_j][\sigma_j]\,)\;[\langle x, y_j \rangle / \langle x_j, y_j \rangle]$$

It follows that the vertices of F_B are those of F_A plus the set $\{v_j \mid v \in \{c, x\} \lor u \leq_A v\}$. Moreover, $u \leq_A v \iff u \leq_B v$: from left to right, in the definition of F_B only the substitution $[\langle x, y_j \rangle / \langle x_j, y_j \rangle]$ removes dependencies, and $u \nleq_A x$; from right to left is immediate from Lemma 6.2.7 (which relates dependencies in F_B to those in F_A). From this lemma it is also immediate that in F_B both c and c_j are safe.

Then after the logical step $F_B \overset{c_j}{\rightsquigarrow} F_C$ removes u_j and x_j, the vertices in F_C are

$$V_A \;\cup\; \{v_j \mid v = c \;\lor\; u <_A v\}\ .$$

In addition, $u \nleq_C v_j$, and $u \leq_C v$ if and only if $u \leq_A v$, for all v in F_a, as follows. Firstly, if $u \leq_C v_j$ then by Lemma 6.2.6 either $u \leq_B v_j$ or $u \leq_B \langle x_j, y_j \rangle$. If $u \leq_B v_j$, by Lemma 6.2.7 $u \nleq_A v$, but then by the above v_j should not exist as a vertex in F_A, a contradiction. If $u \leq_B \langle x_j, y_j \rangle$ then by Lemma 6.2.7 $u \leq_A y_j$, contradicting safety of c. Then $u \nleq_C v_j$ for all v in F_A.

Above, it was shown that $u \leq_B v \iff u \leq_A v$. To show that $u \leq_C v \iff u \leq_B v$, first let $u \leq_C v$. By Lemma 6.2.6, $u \leq_B v$; the other cases, where $u \leq_B \langle x_j, y_j \rangle$, were ruled out above. For the converse, let $u \leq_B v$. By Lemma 6.2.6, $u \leq_A v$ unless u is one of c and x, which clearly cannot transpire. Then $u \leq_C v \iff u \leq_A v$.

Next, it is shown that in F_C the cut c is safe. Suppose $x \leq_C v$ and $u \leq_C v$; by the above, $u \leq_A v$, which means v is in F_A (it is not a duplicated node v'_j). Then for $x \leq_C v$

Lemma 6.2.6 gives $\mathrm{x} \leq_{\mathrm{B}} \mathrm{v}$, unless $\langle \mathrm{u}_j, \mathrm{w}_j \rangle \leq_{\mathrm{B}} \mathrm{v}$ or $\mathrm{v} = \mathrm{c}_j$, which are ruled out since $\mathrm{v} \neq \mathrm{v}'_j$. For $\mathrm{x} \leq_{\mathrm{B}} \mathrm{v}$ Lemma 6.2.7 gives $\mathrm{x} \leq_{\mathrm{A}} \mathrm{v}$; then c is unsafe in $\mathrm{F_A}$, a contradiction.

In $\mathrm{F_C}$ the cut c has $n-1$ existential edges, $\langle \mathrm{x}, \mathrm{y}_i \rangle$ for $1 \leq i \leq n$ such that $i \neq j$. By induction on the number of existential edges of c, the compound reduction step $\mathrm{F_C} \overset{\mathrm{c}}{\rightsquigarrow}{}^{-} \mathrm{F_D}$ is computed by $\mathrm{F_C} \overset{\mathrm{c}}{\rightsquigarrow}{}^{!} \mathrm{F_D}$ (up to a renaming of vertices). Recall that the vertices in $\mathrm{F_C}$ are

$$\mathrm{V_A} \cup \{\mathrm{v}_j \mid \mathrm{v} = \mathrm{c} \vee \mathrm{u} <_{\mathrm{A}} \mathrm{v}\} \ .$$

Applying **1.** and **2.** to $\mathrm{F_C} \overset{\mathrm{c}}{\rightsquigarrow}{}^{!} \mathrm{F_D}$, the vertices of $\mathrm{F_D}$ are the duplicated ones,

$$\{\mathrm{v}_i \mid \mathrm{v} \in \mathrm{V_C},\ 1 \leq i \leq n,\ i \neq j,\ \mathrm{u} <_{\mathrm{C}} \mathrm{v} \vee \mathrm{v} = \mathrm{c}\} \ ,$$

plus the original ones that are not removed,

$$\{\mathrm{v} \in \mathrm{V_C} \mid \mathrm{v} \notin \{\mathrm{x}, \mathrm{c}\},\ \mathrm{u} \nleq_{\mathrm{C}} \mathrm{v}\} \ .$$

Applying the characterisation of $\mathrm{V_C}$, above, to these sets, while using the earlier established fact that $\mathrm{u} \leq_{\mathrm{C}} \mathrm{v} \iff \mathrm{u} \leq_{\mathrm{A}} \mathrm{v}$, gives the following two sets, respectively.

$$\{\mathrm{v}_i \mid \mathrm{v} \in \mathrm{V_A},\ 1 \leq i \leq n,\ i \neq j,\ \mathrm{u} <_{\mathrm{A}} \mathrm{v} \vee \mathrm{v} = \mathrm{c}\}$$

$$\{\mathrm{v} \in \mathrm{V_A} \mid \mathrm{v} \notin \{\mathrm{x}, \mathrm{c}\},\ \mathrm{u} \nleq_{\mathrm{A}} \mathrm{v}\} \cup \{\mathrm{v}_j \mid \mathrm{v} \in \mathrm{V_A},\ \mathrm{v} = \mathrm{c} \vee \mathrm{u} <_{\mathrm{A}} \mathrm{v}\}$$

Their union, the following set, are the vertices of $\mathrm{F_D}$:

$$\mathrm{V_D} \ = \ \{\mathrm{v} \in \mathrm{V_A} \mid \mathrm{v} \notin \{\mathrm{x}, \mathrm{c}\},\ \mathrm{u} \nleq_{\mathrm{A}} \mathrm{v}\} \cup \{\mathrm{v}_i \mid \mathrm{v} \in \mathrm{V_A},\ 1 \leq i \leq n,\ \mathrm{u} <_{\mathrm{A}} \mathrm{v} \vee \mathrm{v} = \mathrm{c}\}$$

By **1.** and **2.** it is then immediate that $\mathrm{V_D} = \mathrm{V_T}$. From this, to show that $\mathrm{F_D} = \mathrm{F_T}$ is straightforward. ☐

Weak normalisation

A compound step replaces a cut with cut-formula $C = \forall x.B$ by a number of cuts each with a cut-formula $B[t/x]$ for some term t. The strict reduction in formula complexity allows an easy proof of weak normalisation. Let the *complexity* $compl(c)$ of a cut $\langle \bot, C, \mathrm{c} \rangle$ be the number of quantifiers in C, and let the *complexity* of a forest be the multiset of the complexities of all its cuts.

Theorem 6.4.9 (Weak normalisation)**.** *For any safe proof forest* $\mathrm{F_A}$ *there is a finite reduction path* $\mathrm{F_A} \rightsquigarrow^{*} \mathrm{F_B}$ *such that* $\mathrm{F_B}$ *is cut-free.*

Proof. Given a forest F_A that is not cut-free, select a cut $c \in V_A(\bot)$ that has no cut with equal or higher complexity amongst its dependants:

$$\forall d \in V_A(\bot).\ c < d \ \Rightarrow\ \mathit{compl}(c) > \mathit{compl}(d)\ ;$$

by the acyclicity of the dependency such a cut exists. Then if $F_A \overset{c}{\rightsquigarrow} F_B$ the complexity of F_B is strictly smaller, in the usual multiset ordering, than that of F_A: the primary cut c is replaced by several cuts of smaller complexity, and no cut of same or higher complexity is duplicated. The smallest value in the complexity measure, $\varnothing$, applies to forests that are cut-free. □

The proof is similar to Gentzen's original proof of weak normalisation for the sequent calculus, and in that sense, standard. The condition imposed on reductions in the above proof is simple and general: any cut whose dependants include only cuts of lower complexity may be reduced. Moreover, using the modified reduction algorithm the original counterexample in Figure 6.3 now strongly normalises, and no other mechanism has been found that may generate infinite reduction paths. For these reasons the following conjecture is put forward.

Conjecture 6.4.10. *The relation* $(\rightsquigarrow)$ *is strongly normalising.*

To conclude this section, Figure 6.10 shows a normalising reduction path in $(\rightsquigarrow)$ for the universal counterexample. The path uses pruning and is consistent with the condition in Theorem 6.4.9 that a cut may not be reduced if it has dependants of higher complexity. This illustrates that for the theorem, both pruning and the use of compound steps is necessary.

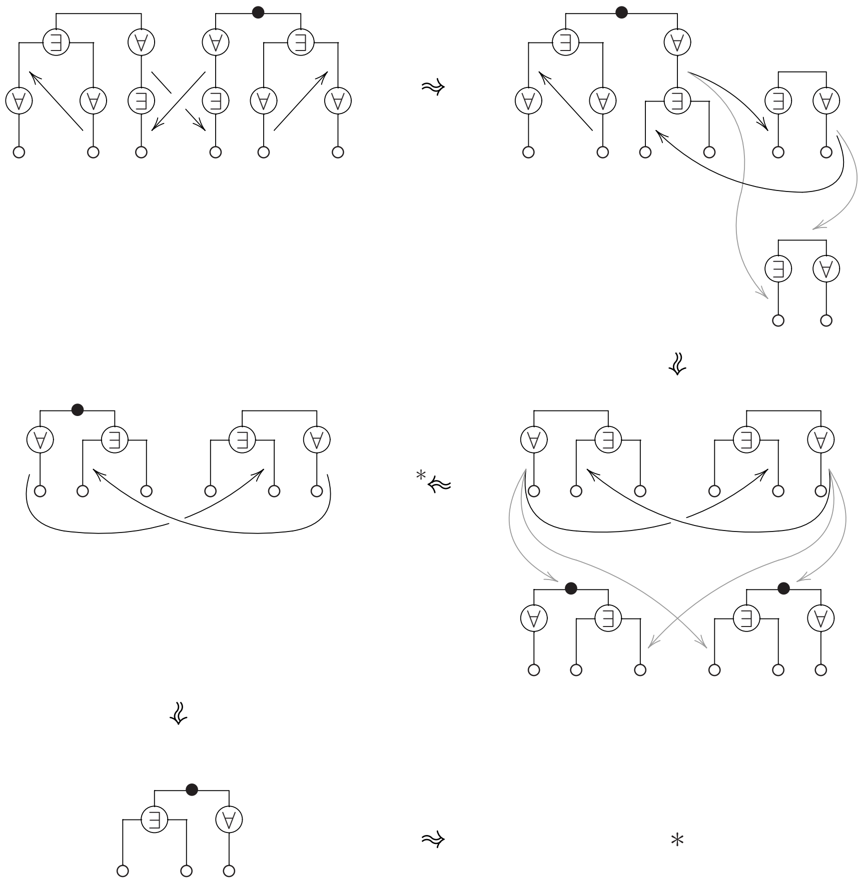


Figure 6.10: Reducing the universal counterexample with compound steps

Chapter 7

Exploring reduction

7.1 Introduction

This chapter will further explore the behaviour of cut-reduction for classical proof forests. The main body of the chapter is formed by Section 7.2, in which it is shown that the original reduction relation $(\rightsquigarrow)$, without pruning, is already weakly normalising. The approach is based on an analysis of the reduction of the universal counterexample, and the connections with the sequent calculus. By prohibiting certain reduction steps, reductions can be forced to stay within the image of the translation of sequent proofs (this translation was defined in Section 5.5).

The treatment of reductions in Section 7.2 suggests further approaches to cut-elimination in proof forests. Their discussion in Section 7.3 explores the differences between reductions in proof forests and those in the sequent calculus, ending with an examination of McKinley's closely related notion of Herbrand nets [74]. The final subject discussed, in Section 7.4, is that of confluence. For the different reduction variants discussed in this and the previous chapter, confluence fails in a variety of ways; however, interestingly, the universal counterexample is universally non-confluent.

7.2 Weak normalisation without pruning

A remaining question is whether the original reduction relation, $(\rightsquigarrow)$, might be weakly normalising. This is left open by the universal counterexample, which does have terminating reduction paths to a normal form (one is illustrated in Figure 6.6). In this section a proof of weak normalisation of $(\rightsquigarrow)$ will be constructed, without the need for pruning. The core idea of the proof is to avoid reduction steps that duplicate a subforest that is strictly smaller than a corresponding subproof in sequent calculus would be. In the discussion of the universal counterexample in Section 6.3 such steps were pinpointed as the cause of the loss of weak normalisation in its reduction paths.

In Section 5.5 in the previous chapter it was shown how the smallest subproof of an inference may contain more than just the dependants of the corresponding edge in a proof forest. It was demonstrated, by the example below left, that an inference cannot permute above a cut when it has premises in both subproofs of the cut, while no corresponding dependency need exist in the proof forest translation of the proof. Below right a similar impermutability is depicted; here, the premise of a universal quantifier introduction is used in one subproof of a cut, while its eigenvariable is used in the other.

$$\dfrac{\dfrac{\vdash \Gamma,B,A \qquad \vdash A^{\perp},B,\Gamma'}{\vdash \Gamma,B,B,\Gamma'}\text{Cut}}{\vdash \Gamma,B,\Gamma'}\text{CR} \qquad\qquad \dfrac{\dfrac{\vdash A(a),C \qquad \dfrac{\vdash C^{\perp},\ \ B(a)}{\vdash C^{\perp},\exists y.B(y)}\exists\text{R}}{\vdash \ \ A(a),\ \ \exists y.B(y)}\text{Cut}}{\vdash \forall x.A(x),\exists y.B(y)}\forall\text{R}$$

The first thing that will be addressed is to formalise a notion of *separation* in proof forests, that corresponds, morally, to ‘being in separate subproofs’ in a sequent proof. For the above two examples, where the subproofs are generated by a single cut, the conflict notion (Definition 6.4.1) would be adequate: the dependants on either side of a cut correspond to (smallest) subproofs, and the conflict generated by the cut indicates when vertices depend on different sides. However, in Section 6.3 the reduction of the universal counterexample provided an example, in Figure 6.8, where the premises of an inference are in subproofs separated not by one, but by two cuts. This can be generalised to the configuration below, where two formulae, A and B, are separated by a number of cuts, that may be interspersed among other inferences.

$$\begin{array}{ccccc} \Pi_1 & \Pi_2 & \Pi_3 & & \Pi_n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdash A,\Gamma_1,C_1 & \vdash C_1^{\perp},\Gamma_2,C_2 & \vdash C_2^{\perp},\Gamma_3,C_3 & & \vdash C_n^{\perp}\Gamma_n,B \\ \hline\hline \multicolumn{5}{c}{\vdash A,\Gamma'_1,\ldots,\Gamma'_n,B} \end{array}\text{Cut,?R}$$

The notion of separation to be established will thus need a certain measure of transitivity. But full transitivity is too much, since separation must also be symmetric; together this would mean that a vertex v that is separated from any other, is immediately separated from itself. This would render the notion useless: a self-separated vertex is precisely what should indicate that a proof forest is not the translation of any sequent proof. A notion of separation that captures the right amount of transitivity is defined below. It allows vertices to be separated by a series of cuts, as long as those cuts are distinct. The separation of vertices v and w by a set of cuts $C = \{\mathrm{c}_1, \ldots, \mathrm{c}_n\}$ is written $\mathrm{v} \mathbin{\#\#^{C}} \mathrm{w}$, illustrated in Figure 7.1 (note the annotation of the abbreviated cut with the name of the cut vertex, rather than the cut-formula).

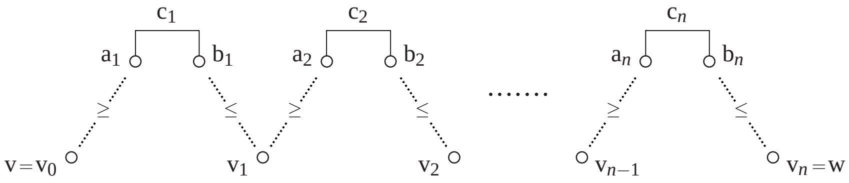


Figure 7.1: Separation

Definition 7.2.1 (Separation)**.** In a forest F the ternary *separation* relation

$$- \mathbin{\#\#^{-}} - \quad \subseteq \quad \mathrm{V} \times \mathcal{P}(\mathrm{V}(\bot)) \times \mathrm{V}$$

is the smallest relation satisfying the following.

$$\begin{array}{ll}
\mathrm{v} \ \#\#^{\{\mathrm{c}\}} \ \mathrm{w} & \text{if } \langle \mathrm{c}, \mathrm{v} \rangle, \langle \mathrm{c}, \mathrm{w} \rangle \in \mathrm{E}(\wedge) \text{ and } \mathrm{v} \neq \mathrm{w} \\
\mathrm{v} \ \#\#^{C \cup D} \ \mathrm{w} & \text{if } \mathrm{v} \ \#\#^{C} \ \mathrm{u} \text{ and } \mathrm{u} \ \#\#^{D} \ \mathrm{w} \text{ for some } \mathrm{u} \in \mathrm{V}, \text{ and } C \cap D = \varnothing \\
\mathrm{v} \ \#\#^{C} \ \mathrm{w} & \text{if } \mathrm{v}' \leq \mathrm{v}, \ \mathrm{w}' \leq \mathrm{w}, \text{ and } \mathrm{v}' \ \#\#^{C} \ \mathrm{w}'
\end{array}$$

If $\mathrm{v} \mathbin{\#\#^{C}} \mathrm{w}$ it is said that C *separates* v and w. The notation $(\#\#)$ denotes the union over all relations $(\#\#^{C})$ for all sets of cuts $C \subseteq \mathrm{V}(\bot)$ in F. The conflict relation $(\#)$ is recovered as the union of $(\#\#^{C})$ over all singletons C.

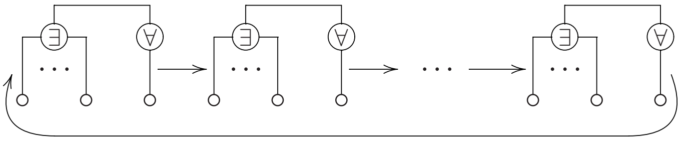
Definition 7.2.2 (Strong safety)**.** In a forest F, if $\mathrm{v} \mathbin{\#\#} \mathrm{v}$ for no vertex $\mathrm{v} \in \mathrm{V}$, then F is *strongly safe*.

Since $\mathrm{v} \mathbin{\#} \mathrm{w}$ implies $\mathrm{v} \mathbin{\#\#} \mathrm{w}$, if a proof forest is strongly safe, it is also safe. The translation of a sequent proof is strongly safe.

Proposition 7.2.3. *A forest* $[\![\Pi]\!]$ *translated from a sequent proof* Π *is strongly safe.*

Proof. The translation of an instance of the tautology axiom is strongly safe, and it is straightforward that strong safety is preserved by the translation steps for $\forall$R-inferences, $\exists$R-inferences, contractions, and weakenings, since these only add root nodes, or modify them. If two forests F_A and F_B are combined by a cut c, then there are no dependencies between them, and no vertices other than c have dependants in both. If $\mathrm{v} \mathbin{\#\#^{C}} \mathrm{w}$ in the composed forest, then either $C \subseteq \mathrm{V}_\mathrm{A}(\wedge)$ and $\mathrm{v} \mathbin{\#\#^{C}_{\mathrm{A}}} \mathrm{w}$, or $C \subseteq \mathrm{V}_\mathrm{B}(\wedge)$ and $\mathrm{v} \mathbin{\#\#^{C}_{\mathrm{B}}} \mathrm{w}$, or $\mathrm{v} \in \mathrm{V}_\mathrm{A}$, $\mathrm{w} \in \mathrm{V}_\mathrm{B}$, and $\mathrm{c} \in C$. ☐

The two causes of non-normalisation explored in Section 6.3 were, firstly, unsafe cuts, and secondly, circles of cuts, illustrated below. Where safety, based on the conflict relation, rules out unsafe cuts, strong safety prohibits the existence of a circle of cuts in a proof forest.



Of the proof forests arising when reducing the counterexample, those that are not weakly normalising are those that are not strongly safe. The reduction step where strong safety is lost is always the structural reduction step applied to the second original cut of the example, as in the exploration of the loss of weak normalisation in Figure 6.7.

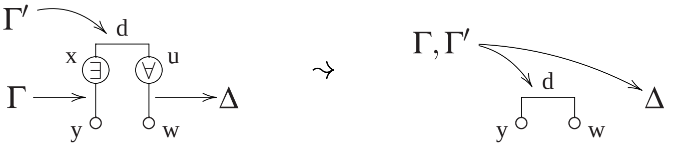
Before moving on, it will be proved that propositional, disposal, and logical steps preserve strong safety.

Lemma 7.2.4. *If* $\mathrm{F_A} \overset{\mathrm{d}}{\rightsquigarrow} \mathrm{F_B}$ *with a propositional step (**I**, Definition 6.2.2) or disposal step (**II**, Definition 6.2.3) then* $\mathrm{v} \mathbin{\#\#_\mathrm{B}} \mathrm{w}$ *only if* $\mathrm{v} \mathbin{\#\#_\mathrm{A}} \mathrm{w}$.

Proof. The statement is immediate from the fact that $\mathrm{F_B}$ is a subforest of $\mathrm{F_A}$, i.e. the fact that $\mathrm{F_B} = \mathrm{F_A}|_X$ for some $X \subseteq \mathrm{V_A}$. □

Lemma 7.2.5. *If* $\mathrm{F_A} \overset{\mathrm{d}}{\rightsquigarrow} \mathrm{F_B}$ *with a logical reduction step (**III**, Definition 6.2.4) and* $\mathrm{F_A}$ *is strongly safe, then* $\mathrm{v} \mathbin{\#\#^C_\mathrm{B}} \mathrm{w}$ *only if* $\mathrm{v} \mathbin{\#\#^D_\mathrm{A}} \mathrm{w}$ *for some* $D \subseteq C \cup \{\mathrm{d}\}$.

Proof. Let $\mathrm{v}_0 \mathbin{\#\#^C_\mathrm{B}} \mathrm{v}_n$ be witnessed as in Figure 7.1: let $C = \{\mathrm{c}_1, \dots, \mathrm{c}_n\}$, let each cut c_i have children a_i and b_i, and let $\mathrm{v}_0, \dots, \mathrm{v}_n$ be vertices such that $\mathrm{a}_i \leq \mathrm{v}_{i-1}$ and $\mathrm{b}_i \leq \mathrm{v}_i$, for $1 \leq i \leq n$. Let the primary cut d of the reduction step be configured as follows.



By Lemma 6.2.6, for a dependency $\mathrm{v} \leq_\mathrm{B} \mathrm{w}$ there are three options:

(i). there is a matching dependency $\mathrm{v} \leq_\mathrm{A} \mathrm{w}$, or

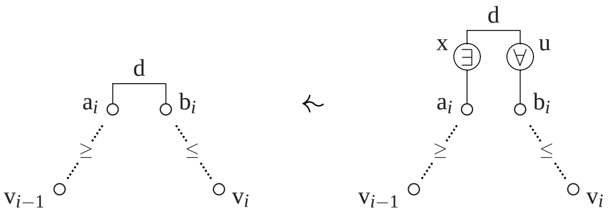
(ii). $\mathrm{v} \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle$ and $\langle \mathrm{u}, \mathrm{z} \rangle \leq_\mathrm{A} \mathrm{w}$, or

(iii). $\mathrm{v} \leq_\mathrm{A} \langle \mathrm{x}, \mathrm{y} \rangle$ and $\mathrm{w} = \mathrm{d}$.

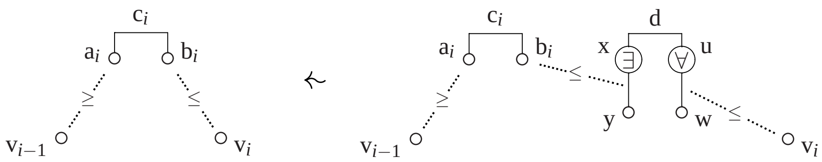
For each individual cut c_i in C, witnessing $\mathrm{v}_{i-1} \#\#_{\mathrm{B}}^{\{\mathrm{c}_i\}} v_i$, there are four possibilities.

(1). The cut is unaffected by the rewrite step: $\mathrm{c}_i \neq \mathrm{d}$ and (i) above applies to both the dependency $\mathrm{a}_i \leq_{\mathrm{B}} \mathrm{v}_{i-1}$ and to $\mathrm{b}_i \leq_{\mathrm{B}} \mathrm{v}_i$. Then $\mathrm{v}_{i-1} \#\#_{\mathrm{A}}^{\{\mathrm{c}_i\}} v_i$.

(2). The cut is itself reduced: $\mathrm{c}_i = \mathrm{d}$. Then $\mathrm{v}_{i-1} \#\#_{\mathrm{A}}^{\{\mathrm{d}\}} v_i$, as illustrated below (only one of two possible orientations for d is shown—the other has x and u swapped).



(3). For the dependence $\mathrm{b}_i \leq \mathrm{v}_i$ item (ii) above applies.



As illustrated, $\mathrm{v}_{i-1} \#\#_{\mathrm{A}}^{\{\mathrm{c}_i\}} y$ and $y \#\#_{\mathrm{A}}^{\{\mathrm{d}\}} v_i$, and hence $\mathrm{v}_{i-1} \#\#_{\mathrm{A}}^{\{\mathrm{c}_i,\mathrm{d}\}} v_i$.

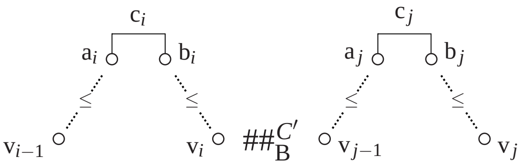
(4). For the dependence $\mathrm{a}_i \leq_{\mathrm{B}} \mathrm{v}_{i-1}$ item (ii) above applies. Then similarly to (3) above, $\mathrm{v}_{i-1} \#\#_{\mathrm{A}}^{\{\mathrm{d},\mathrm{c}_i\}} v_i$.

For both $\mathrm{a}_i \leq_{\mathrm{B}} \mathrm{v}_{i-1}$ and $\mathrm{b}_i \leq_{\mathrm{B}} \mathrm{v}_i$ item (i) subsumes item (iii). Considering the latter, item (iii) gives $\mathrm{b}_i \leq_{\mathrm{B}} \mathrm{d}$ in F_{B}, and $\mathrm{b}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$ in F_{A}. But since $\mathrm{d} \leq_{\mathrm{B}} \mathrm{y}$, there are also the dependencies $\mathrm{b}_i \leq_{\mathrm{B}} \mathrm{y}$ and $\mathrm{b}_i \leq_{\mathrm{A}} \mathrm{y}$; then y may be used instead of d as the v_i in the sequence $\mathrm{v}_1, \ldots, \mathrm{v}_n$.

Then the four options above are exhaustive, and for a single cut c_i they are mutually exclusive. That (1) excludes the others is immediate. Option (2) means that $\mathrm{a}_i = y$ or $\mathrm{b}_i = y$, (3) implies $\mathrm{b}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$, and (4) implies $\mathrm{a}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$. Having both $\mathrm{a}_i = y$ and $\mathrm{a}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$ would violate the antisymmetry of $\leq_{\mathrm{A}}$, while $\mathrm{b}_i = y$ and $\mathrm{a}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$ would mean $y \#\#_{\mathrm{A}}^{\{\mathrm{c}_i\}} y$. Then (2) excludes (3) and, symmetrically, (4); similarly, if both $\mathrm{a}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$ and $\mathrm{b}_i \leq_{\mathrm{A}} \langle \mathrm{x}, \mathrm{y} \rangle$ would hold, again $y \#\#_{\mathrm{A}}^{\{\mathrm{c}_i\}} y$, making (3) and (4) mutually exclusive.

For $\mathrm{v}_0 \#\#_{\mathrm{B}}^{C} \mathrm{v}_n$ the following is then immediate. If (1) applies to all cuts in C then $\mathrm{v}_0 \#\#_{\mathrm{A}}^{C} \mathrm{v}_n$, and if it applies to all but one cuts in C, then $\mathrm{v}_0 \#\#_{\mathrm{A}}^{C \cup \{\mathrm{d}\}} \mathrm{v}_n$. However, the general case is not immediate: if $\mathrm{v}_0 \#\#_{\mathrm{B}}^{C \cup D} \mathrm{v}_n$ because $\mathrm{v}_0 \#\#_{\mathrm{B}}^{C} \mathrm{v}_i$ and $\mathrm{v}_i \#\#_{\mathrm{B}}^{D} \mathrm{v}_n$ for some v_i, then even if $v_0 \, \#\#_A^{C\cup\{d\}} \, v_i$ and $v_i \, \#\#_A^{D\cup\{d\}} \, v_n$ it does not necessarily follow that $v_0 \, \#\#_A \, v_n$, because the sets of cuts are not disjoint.

First, the case will be considered when there are precisely two cuts in C to which (1) does not apply. Let c_i denote the first and c_j the second; they are configured as follows.



Here, $C' \subseteq C$ is the subset $\{c_{i+1}, \ldots, c_{j-1}\}$—note that if $i = j-1$ then $v_i = v_{j-1}$ instead of $v_i \, \#\#_B \, v_{j-1}$, which does not affect the argument below. Since (1) applies to all other cuts in C than c_i and c_j, in particular it applies to all cuts in C', so that $v_i \, \#\#_A^{C'} \, v_{j-1}$ (or $v_i = v_{j-1}$) also before the reduction step, in F_A.

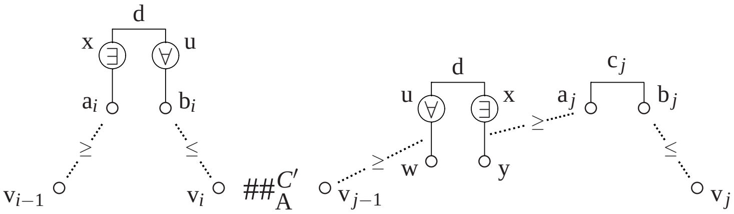
There are nine cases to be considered: one of (2), (3), and (4) applies to c_i, and simultaneously one of (2), (3), and (4) applies to c_j. For each case it will be shown that either

$$v_{i-1} \, \#\#_A^{\{c_i\}} \, v_j \qquad \text{or} \qquad v_{i-1} \, \#\#_A^{\{c_j\}} \, v_j \qquad \text{or} \qquad v_{i-1} \, \#\#_A^{\{c_i, c_j\}} \, v_j \;,$$

and hence that $v_0 \, \#\#_A^D \, v_n$ for some $D \subseteq C$, or that the case cannot transpire, for example because it would imply a separation $v \, \#\#_A \, v$ in F_A, contradicting the assumption of strong safety.

- If (2) applies to both c_i and c_j then $c_i = c_j = d$, while c_i and c_j were assumed to be distinct, a contradiction.

- If (2) applies to c_i and (3) to c_j, there are two ways in which c_i and d can be identified: a_i is on the existential side and b_i on the universal side of d, or the other way around. In the former case (for later reference, with the existential side 'on the left'), there is the following configuration in F_A.

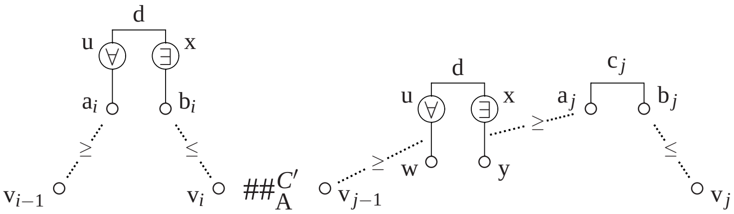


  Note that in the above illustration the distinct occurrences of the vertices x and u, and the cut d, must be identified. As d is a logical cut x has only one edge, and $a_i = y$. The remaining equalities and inequalities below are readily observed.

$$a_j \quad \leq_A \quad \langle x, y \rangle \quad \leq_A \quad y \quad = \quad a_i \quad \leq_A \quad v_{i-1}$$

A direct consequence of $a_j \leq v_{i-1}$ is then $v_{i-1} \; \#\#_A^{\{c_j\}} \; v_j$.

The other possible way of identifying c_i and d (with the existential side on the right) gives the following configuration.



From the following dependencies

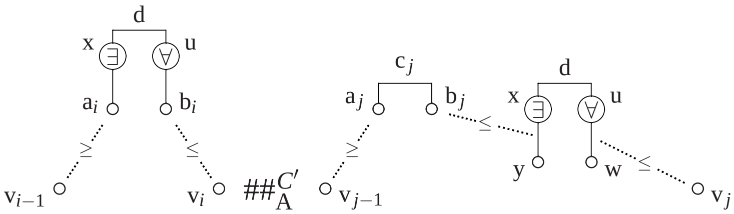
$$u \leq \langle u, w \rangle \leq v_{j-1} \qquad x \leq b_i \leq v_i$$

the first of the two separations below follows.

$$v_{j-1} \; \#\#_A^{\{d\}} \; v_i \; \#\#_A^{C'} \; v_{j-1}$$

Since $d \notin C'$, the vertex v_{j-1}, among others, is separated from itself, contradicting the assumption that F_A is strongly safe.

- If (2) applies to c_i and (4) to c_j, again d and c_i can be identified in two ways. One gives the following configuration in F_A.

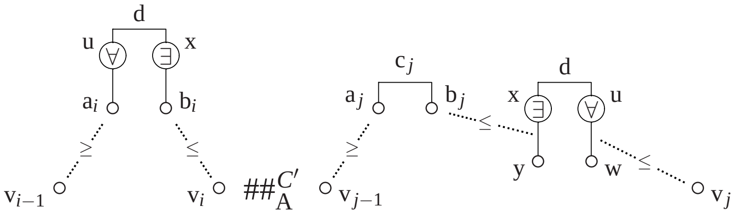


The three separations below are readily observed.

$$a_i \; \#\#_A^{\{d\}} \; v_i \; \#\#_A^{C'} \; v_{j-1} \; \#\#_A^{\{c_j\}} \; y$$

Because (2) does not apply to c_j, which is then distinct from d, the three sets of cuts involved are disjoint, and $a_i \; \#\#_A \; y$. Identifying both occurrences of x means $a_i = y$; it then follows that $y \; \#\#_A \; y$, a contradiction.

The other orientation of d gives the configuration below.

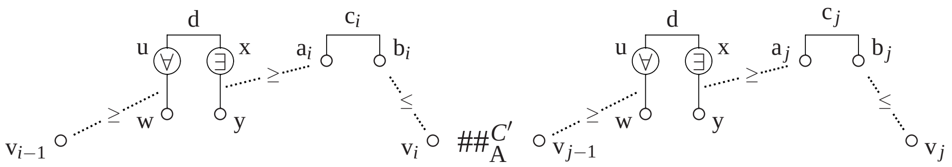


The following separations and (in)equalities can be observed; in particular, $y = b_i$ by identifying both occurrences of x.

$$v_i \ \#\#_A^{C'} \ v_{j-1} \ \#\#_A^{\{c_j\}} \ y \qquad\qquad y = b_i \leq v_i$$

Combining the equations above gives $v_i \ \#\#_A \ v_i$, a contradiction.

- The case where (3) applies to c_i and (2) to c_j is symmetrical to the above one, where (2) applies to c_i and (4) to c_j.
- If (3) applies to both c_i and c_j, there is the following configuration in F_A.

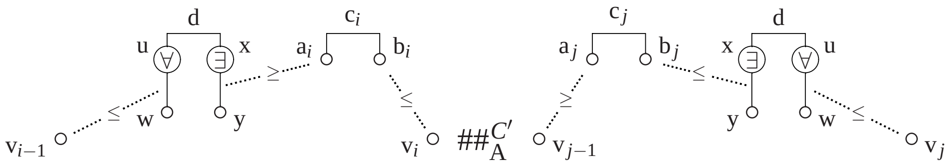


In the illustration, from left to right the following three separations can be observed.

$$y \ \#\#_A^{\{c_i\}} \ v_i \ \#\#_A^{C'} \ v_{j-1} \ \#\#_A^{\{d\}} \ y$$

Since all three sets of cuts are disjoint, $y \ \#\#_A \ y$, a contradiction.

- If (3) applies to c_i and (4) to c_j, there is the configuration below.

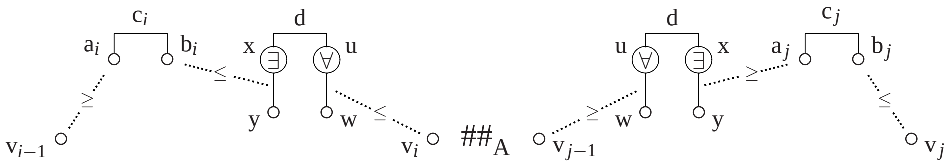


From left to right, the following separations can be observed.

$$y \ \#\#_A^{\{c_i\}} \ v_i \ \#\#_A^{C'} \ v_{j-1} \ \#\#_A^{\{c_j\}} \ y$$

Then $y \ \#\#_A \ y$, a contradiction.

- The case where (4) applies to c_i and (2) to c_j is symmetrical to the second case above, where (2) applies to c_i and (3) to c_j. That is, if d has the existential side on the right, then $v_{i-1} \ \#\#_A^{\{c_i\}} \ v_j$, otherwise the case leads to a contradiction.
- If (4) applies to c_i and (3) to c_j, then F_A contains the configuration below.



On the far left and far right, the following separations can be observed.

$$\mathrm{v}_{i-1}\ \#\#_{\mathrm{A}}^{\{\mathrm{c}_i\}}\ \mathrm{y} \qquad\qquad \mathrm{y}\ \#\#_{\mathrm{A}}^{\{\mathrm{c}_j\}}\ \mathrm{v}_j$$

Then $\mathrm{v}_{i-1}\ \#\#_{\mathrm{A}}^{\{\mathrm{c}_i,\mathrm{c}_j\}}\ \mathrm{v}_j$.

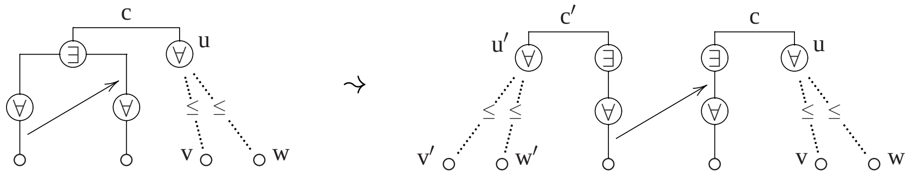
- The case where (4) applies to both c_i and c_j is symmetrical to the fourth case, where (3) applies to both cuts.

The case where three or more cuts in C do not satisfy (1) would follow by inductively taking fragments $\mathrm{v}_i\ \#\#_{\mathrm{B}}^{C'}\ \mathrm{v}_j$ of the separation $\mathrm{v}_0\ \#\#_{\mathrm{B}}^{C}\ \mathrm{v}_n$ such that $C' \subseteq C$ contains precisely two cuts to which (2), (3), or (4) apply, after which the statement would follow because $\mathrm{v}_i\ \#\#_{\mathrm{A}}^{D'}\ \mathrm{v}_j$ for some $D' \subseteq C'$. However, in fact it can be observed that having three or more cuts in C not satisfying (1) always leads to a contradiction. The three cases above that do not immediately prove a contradiction are those where (2) and (3), or (4) and (2), or (4) and (3) apply to c_i and c_j respectively. The first two of these are symmetric, which means that (2) applies with a different orientation in both: in the first, with the existential side on the left, and in the second, with the existential side on the right. □

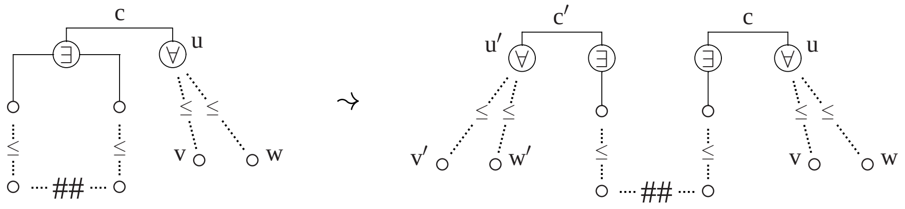
Straddling

Before, it was shown that forests translated from sequent proofs are strongly safe, and that strong safety is preserved in propositional, disposal, and logical reduction steps. Since strong safety is lost in the reduction of the counterexample, structural steps do not preserve it. The remainder of this section will explore the way structural reduction steps interact with separation, and use the findings to construct a class of reduction strategies that preserve strong safety.

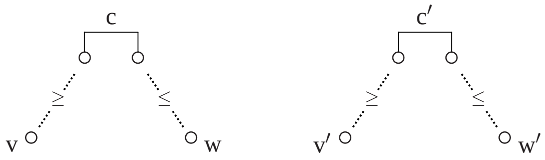
There are two ways in which a structural step may introduce a separation; strong safety is lost when both occur simultaneously. The first is pictured below, as it occurs in the reductions of the universal counterexample. A dependency between the existential branches of a cut c introduces a separation $\mathrm{v}\ \#\#^{\{\mathrm{c},\mathrm{c}'\}}\ \mathrm{w}'$ for every v and w such that $\mathrm{u} \leq \mathrm{v},\mathrm{w}$; that is, between every non-duplicated and every duplicated dependant of the universal side of c.



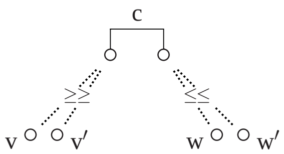
A generalisation of the above example replaces the dependency between the existential branches of the cut by a separation, as follows.



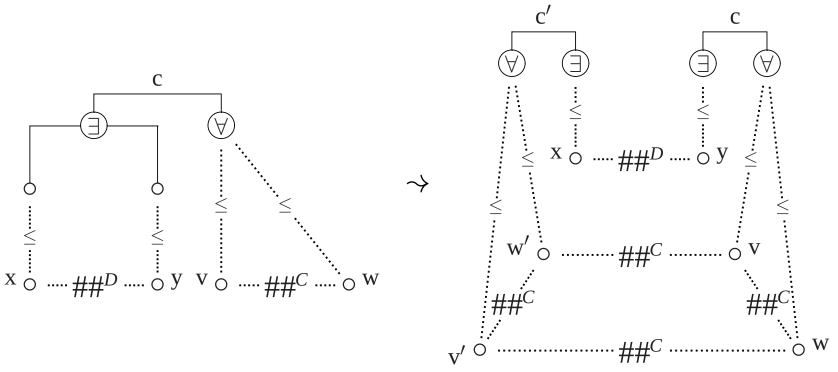
The above way that separation is introduced is unavoidable: cuts with dependencies between their existential branches do occur, and it must possible to reduced them. The second method by which structural steps create separation, which may be avoided, is as follows. If the cut below left is duplicated by a structural step, the cut below right is created; note how v ## w and v′ ## w′, but not v ## w′, or vice versa.



However, if only the vertices v and w are duplicated, but not the cut c itself, the result is as follows.



In this case not just v ## w and v′ ## w′, but also v ## w′ and v′ ## w. Again, this can be generalised, by replacing the single cut c separating v and w by an arbitrary separation ($\#\#^C$). The situation thus described is exactly that where the dependants duplicated in a structural step are strictly smaller than a corresponding subproof in the sequent calculus, described in Section 6.3. Then when also the existential branches of the cut contain a separation, the reduction step breaks strong safety, as illustrated below.



In the resulting proof forest, if $E = D \cup \{\mathrm{c}, \mathrm{c}'\}$, then $\mathrm{v} \mathrel{\#\#^{C}} \mathrm{w}'$ and $\mathrm{w}' \mathrel{\#\#^{E}} \mathrm{v}$, which taken together give $\mathrm{v} \mathrel{\#\#^{C\cup E}} \mathrm{v}$ and $\mathrm{w}' \mathrel{\#\#^{C\cup E}} \mathrm{w}'$ In the same way, v' and w are self-separated due to the separation $\mathrm{v}' \mathrel{\#\#^{E}} \mathrm{w}$, but note that E does not separate v from w, or v' from w'.

The mechanism thus described is precisely what causes the loss of strong safety in the universal counterexample. It will be shown that problematic reduction steps of the kind described above can be avoided, yielding a weak normalisation proof for $(\rightsquigarrow)$. First, in the situation where a vertex u has dependants v and w separated by C, while some c in C does not depend on u, it is said that u *straddles* c, written $\mathrm{u} \vartriangle \mathrm{c}$. Straddling is defined below, and $\mathrm{u} \vartriangle \mathrm{c}$ is illustrated in Figure 7.2.

Definition 7.2.6. A vertex u *straddles* a cut $\mathrm{c} \in \mathrm{V}(\perp)$, written $\mathrm{u} \vartriangle \mathrm{c}$, as follows.

$$\mathrm{u} \vartriangle \mathrm{c} \quad \overset{\Delta}{\iff} \quad \exists \mathrm{v}, \mathrm{w}, C. \quad \mathrm{u} \leq \mathrm{v}, \mathrm{w}, \quad \mathrm{u} \nleq \mathrm{c}, \quad \mathrm{v} \mathrel{\#\#^{C}} \mathrm{w}, \quad \text{and} \quad \mathrm{c} \in C$$

The set of cuts C is a *witness* for $\mathrm{u} \vartriangle \mathrm{c}$.

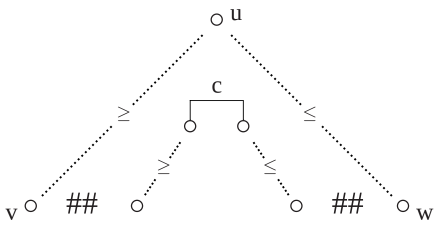


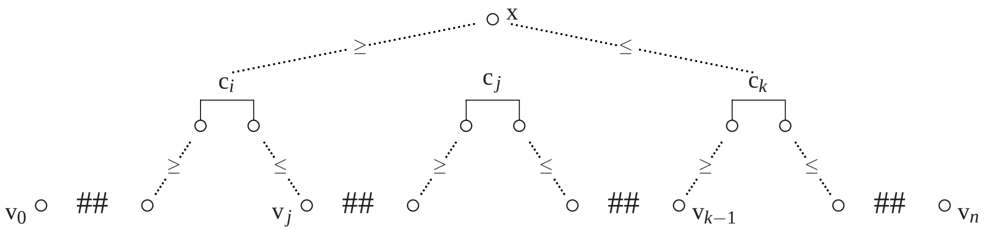
Figure 7.2: Straddling

For flexibility, if C is a witness for $\mathrm{u} \vartriangle \mathrm{c}$, the definition of straddling does allow other cuts in C than c to depend on u. By the following easy lemma a smaller witness $D \subseteq C$ can always be found such that u straddles all cuts in D.

Lemma 7.2.7. *If C is a witness for $\mathrm{x} \vartriangle \mathrm{c}$ then there is a witness $D \subseteq C$ for $\mathrm{x} \vartriangle \mathrm{c}$ such that $\mathrm{x} \nleq \mathrm{d}$ for all $\mathrm{d} \in D$.*

Proof. Let $\mathrm{x} \leq \mathrm{v}_0, \mathrm{v}_n$, let $C = \{\mathrm{c}_1, \ldots \mathrm{c}_n\}$, let $\mathrm{v}_{i-1} \mathrel{\#\#^{\{\mathrm{c}_i\}}} \mathrm{v}_i$ for $1 \leq i \leq n$, and let $\mathrm{c} = \mathrm{c}_j$. If $\mathrm{x} \leq \mathrm{c}_i$ then $\mathrm{x} \leq \mathrm{v}_{i-1}$ and $\mathrm{x} \leq \mathrm{v}_i$. Let i be the largest index smaller than j such that $\mathrm{x} \leq \mathrm{c}_i$, or $i = 0$ if no such c_i exists; and let k be the smallest index greater than j such that $\mathrm{x} \leq \mathrm{c}_k$, or $k = n+1$ if no such c_k exists. Then $\mathrm{x} \leq \mathrm{v}_i$ and $\mathrm{x} \leq \mathrm{v}_{k-1}$, while

$v_i \mathbin{\#\#}^D v_{k-1}$, where $D = \{c_{i+1}, \dots, c_{k-1}\}$. In particular, $c \in D$, while $x \not\leq d$ for all $d \in D$.



□

Straddling captures, in proof forests, an impermutability in the sequent calculus not accounted for in the dependency, between a cut and an inference with premises (or eigenvariable occurrences) in both subproofs of the cut. Since the ordering formed by the combination of the straddling relation and the dependency, $(\vartriangle \cup \leq)^*$, represents a non-permutable ordering of inferences in a sequent proof, it is natural to require it to be antisymmetric. In fact, this is already the case in any proof forest that is strongly safe, as will be established in Lemma 7.2.11. Then in a strongly safe proof forest there is always a cut that has no dependants of greater complexity, and does not straddle another. Showing that reducing this cut preserves strong safety, in Lemma 7.2.10, will then allow a weak normalisation proof along the lines of that of Theorem 6.4.9, again using compound reduction steps to obtain a simple reducing measure.

Formalising this proof idea starts with an easy, but convenient lemma.

Lemma 7.2.8. *In a structural reduction step* $F_A \overset{c}{\rightsquigarrow} F_B$ *(**IV**, Definition 6.2.5) no duplicated vertex* v' *depends on* c *in* F_B.

Proof. If u and x are respectively the universal and existential child of c, then v' cannot depend on u in F_B, because by Lemma 6.2.7 no dependants of u are duplicates, and must depend on x. At the same time, v' must be x' or c', or a dependant of u', because it is a duplicate. However, $c \leq_B c'$ (and also $c \leq_B x'$) would mean $x \leq_B c'$ or $u \leq_B c'$, and hence $c \leq_A x \leq_A c$ or $c \leq_A u \leq_A c$, contradicting antisymmetry of $\leq_A$. Then $u' \leq_B v'$ and hence $u \leq_A v$, and in F_A, the vertex v depends on both x and u, contradicting that c must be safe in F_A for the reduction step to apply. □

Next, it is shown that a reducing a cut that straddles no others preserves strong safety.

Lemma 7.2.9. *In a structural reduction step* $F_A \overset{c}{\rightsquigarrow} F_B$ *(**IV**, Definition 6.2.5) on a strongly safe proof forest* F_A*, where* c *straddles no cut* d*, then (**1**)* F_B *is strongly safe, and (**2**)* c *straddles no cuts in* F_B.

Proof. A separation $(\#\#^D)$ in $\mathrm{F_B}$ where D is a singleton takes one of the following eight forms, where v are w are vertices, and d is a cut, in $\mathrm{F_A}$.

$$\begin{array}{cccc} v \#\#_B^{\{d\}} w & v' \#\#_B^{\{d\}} w & v \#\#_B^{\{d\}} w' & v' \#\#_B^{\{d\}} w' \\ v \#\#_B^{\{d'\}} w & v' \#\#_B^{\{d'\}} w & v \#\#_B^{\{d'\}} w' & v' \#\#_B^{\{d'\}} w' \end{array}$$

These options will first be narrowed down. By Lemma 6.2.7, if duplicated vertex has a non-duplicated dependant, the duplicated vertex must be x' or c'. Then in two cases above, $v' \#\#_B^{\{d'\}} w$ and $v \#\#_B^{\{d'\}} w'$, the cut d' must be c', the duplicate of the primary cut, because an original vertex, w or v respectively, depends on it.

Two other cases are ruled out altogether. One is $v' \#\#_B^{\{d\}} w'$, top right, which could only be produced in the reduction step if $c \vartriangle_A d$ (a contradiction), by the following reasoning. Let u be the universal child of the primary cut c, and x the existential child; since v' and w' are duplicates, $u \leq_A v, w$. By Lemma 7.2.8, which states that c has no dependants in $\mathrm{F_B}$ that are duplicates, $u \not\leq_A d$; otherwise, $u \leq_B d$ would mean $c \leq_B v', w'$. Moreover, $x \leq_A d$ would imply $x \leq v$ and $u \leq v$, and hence $v \#\#_A v$. Then $c \not\leq d$ in $\mathrm{F_A}$, so that $c \vartriangle_A d$, a contradiction.

The second case ruled out is $v \#\#_B^{\{d'\}} w$, bottom left. Since $d' \leq_B v$, by Lemma 6.2.7 the cut d must be the primary cut c itself, and v must reside in the primary branch of the reduction step. Similarly, w must reside in the primary branch, but for the separation to exist in $\mathrm{F_B}$, it must also depend on the universal node u' of c', and on u in $\mathrm{F_A}$. Then $w \#\#_A^{\{c\}} w$, a contradiction.

In addition, in the case $v' \#\#_B^{\{d\}} w$ and the one symmetric to it, the cut d cannot be the primary cut c itself, since by Lemma 7.2.8 no duplicated nodes depend on c in $\mathrm{F_B}$. This leaves the following six possibilities.

$$\begin{array}{cccc} v \#\#_B^{\{d\}} w & v' \#\#_B^{\{d\}} w \; (d \neq c) & v \#\#_B^{\{d\}} w' \; (d \neq c) & \\ & v' \#\#_B^{\{c'\}} w & v \#\#_B^{\{c'\}} w' & v' \#\#_B^{\{d'\}} w' \end{array}$$

That in all of these cases $v \#\#_A^{\{d\}} w$ (or $v \#\#_A^{\{c\}} w$) is straightforward from Lemma 6.2.7. However, this does not mean that in general $v \#\#_B w$ (or $v' \#\#_B w'$, etc.) implies $v \#\#_A w$: if in $u \#\#_B^C v \#\#_B^{C'} w$ the set C contains a cut d while C' contains its duplicate d', then even if $u \#\#_A v$ and $v \#\#_A w$, not necessarily $u \#\#_A w$.

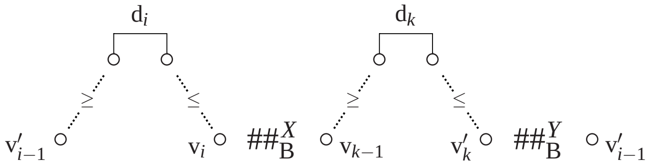
For (**1**), assume $v \#\#_B^C v$ or $v' \#\#_B^C v'$ for some vertex v in $\mathrm{F_A}$. Firstly, if C does not contain both a cut d and its duplicate d', then $v \#\#_A^D v$ where $D = \{d \mid d \in C \vee d' \in C\}$. Otherwise, C does contain two cuts d, d'. From the six cases above it can be observed that on one side of d' at least, all dependants are duplicates w', while on one side of d all dependants are originals w from $\mathrm{F_A}$. Then C contains at least one cut

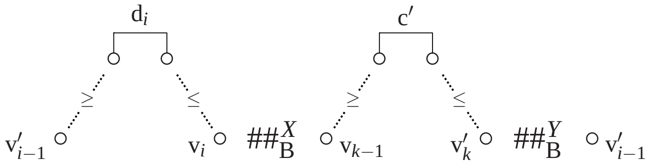
$$\mathrm{v}' \,\#\#_{\mathrm{B}}^{\{\mathrm{d}_i\}}\, \mathrm{w} \quad \text{or} \quad \mathrm{v}' \,\#\#_{\mathrm{B}}^{\{\mathrm{c}'\}}\, \mathrm{w}\ ,$$

where $\mathrm{d}_i \neq \mathrm{c}$, and one cut

$$\mathrm{v} \,\#\#_{\mathrm{B}}^{\{\mathrm{d}_k\}}\, \mathrm{w}' \quad \text{or} \quad \mathrm{v} \,\#\#_{\mathrm{B}}^{\{\mathrm{c}'\}}\, \mathrm{w}'\ ,$$

where $\mathrm{d}_k \neq \mathrm{c}$. This gives three possible configurations: the two illustrated below, and one symmetric to the second. The two illustrated cases will be treated; in the first, $C = \{\mathrm{d}_i, \mathrm{d}_k\} \cup X \cup Y$, in the second, $C = \{\mathrm{d}_i, \mathrm{c}'\} \cup X \cup Y$.





W.l.o.g. it may be assumed that X contains only cuts of the kind $\mathrm{v} \,\#\#_{\mathrm{B}}^{\{\mathrm{d}\}}\, \mathrm{w}$, where v, w, and d are originals. In $\mathrm{F_A}$, the vertices v_{i-1} and v_k both depend on u, the universal child of c. Also, d_i cannot depend on c by Lemma 7.2.8. In the first case,

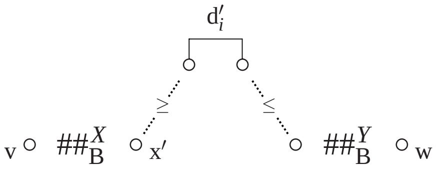
$$\mathrm{v}_{i-1} \ \#\#_{\mathrm{A}}^{X \cup \{\mathrm{d}_i, \mathrm{d}_k\}} \ \mathrm{v}_k\ ,$$

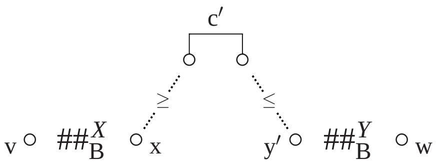
since X contains no duplicated cuts. In the second case,

$$\mathrm{v}_{i-1} \quad \#\#_{\mathrm{A}}^{X \cup \{\mathrm{d}_i\}} \quad \mathrm{v}_{k-1}\ ,$$

while $\mathrm{c} \leq_{\mathrm{A}} \mathrm{v}_{k-1}$ (the case differs from the first for the possibility that X contains c). In both cases, $\mathrm{c} \vartriangle_{\mathrm{A}} \mathrm{d}_i$, a contradiction.

For (**2**), assume $\mathrm{c} \vartriangle_{\mathrm{B}} \mathrm{d}$ for some cut d in $\mathrm{F_B}$. Let $\mathrm{c} \leq_{\mathrm{B}} \mathrm{v}, \mathrm{w}$ and $\mathrm{v} \,\#\#_{\mathrm{B}}^{C}\, \mathrm{w}$ with $\mathrm{d} \in C$; by Lemma 7.2.7 it may be assumed that no cut in C depends on c. If C contains no duplicate cuts then $\mathrm{v} \,\#\#_{\mathrm{A}}^{C}\, \mathrm{w}$. Otherwise, one of the following configurations pertains, where X contains no duplicated cuts.





In the first case, where d_i may be c itself, $\mathrm{c} \leq_{\mathrm{A}} \mathrm{x}$ and $\mathrm{v} \,\#\#_{\mathrm{A}}^{X}\, \mathrm{x}$. Since X cannot be empty ($\mathrm{v} \neq \mathrm{x}'$), and no cut in X depends on c by assumption, $\mathrm{c} \vartriangle_{\mathrm{A}} \mathrm{d}$ for some $\mathrm{d} \in X$, a contradiction. The second case follows similarly, unless X is empty, when $\mathrm{v} = \mathrm{x}$. In that case the argument may be repeated symmetrically, with w taking the place of v; then the second case cannot apply because c' has the wrong orientation, i.e., y' cannot take the place of x. □

It was established earlier that other reduction steps preserve strong safety. Therefore, the preservation of strong safety extends to compound reduction steps, as follows.

Lemma 7.2.10. *If* $\mathrm{F_A} \overset{\mathrm{c}}{\approx\!>}{}^{-} \mathrm{F_D}$ *with* $\mathrm{F_A}$ *strongly safe and no cut* d *s.t.* $\mathrm{c} \vartriangle_{\mathrm{A}} \mathrm{d}$, *then* $\mathrm{F_D}$ *is strongly safe.*

Proof. Since the forest $\mathrm{F_A}$ is strongly safe, it is also safe, and the compound step $(\overset{\mathrm{c}}{\approx\!>}{}^{-})$ may be applied. If the compound step $(\overset{\mathrm{c}}{\approx\!>}{}^{-})$ consists of a single propositional step (**I**) or disposal step (**II**), then $\mathrm{F_D}$ is a subforest of $\mathrm{F_A}$, and the statement is immediate. If it consists of a single logical step (**III**), the statement follows directly from Lemma 7.2.5. It remains to show, for the following successive structural and logical reduction steps,

$$\mathrm{F_A} \overset{\mathrm{c}}{\leadsto} \mathrm{F_B} \overset{\mathrm{c}'}{\leadsto} \mathrm{F_C} \overset{\mathrm{c}}{\approx\!>}{}^{-} \mathrm{F_D}$$

that the forest $\mathrm{F_C}$ is strongly safe and that $\mathrm{c} \vartriangle_{\mathrm{C}} \mathrm{d}$ for no d, after which the strong safety of $\mathrm{F_D}$ follows by induction.

Firstly, by Lemma 7.2.9 no $\mathrm{c} \leq_{\mathrm{B}} \mathrm{d}$ and no $\mathrm{v} \,\#\#_{\mathrm{B}}\, \mathrm{v}$ in the forest $\mathrm{F_B}$. Next, if $\mathrm{v} \,\#\#_{\mathrm{C}}\, \mathrm{v}$ in $\mathrm{F_C}$ then by Lemma 7.2.5 $\mathrm{u} \,\#\#_{\mathrm{B}}\, \mathrm{u}$ for some vertex u. That leaves $\mathrm{c} \vartriangle_{\mathrm{C}} \mathrm{d}$. Let $\mathrm{c} \leq_{\mathrm{C}} \mathrm{v}, \mathrm{w}$ while $\mathrm{v} \,\#\#_{\mathrm{C}}^{X}\, \mathrm{w}$ with $\mathrm{d} \in X$. By Lemma 7.2.7 it may be assumed that no cut x in X depends on c.

Then in $\mathrm{F_B}$, by Lemma 7.2.5 $\mathrm{v} \,\#\#_{\mathrm{B}}^{Y}\, \mathrm{w}$ for some $Y \subseteq X \cup \{\mathrm{c}'\}$. Also, $\mathrm{c} \nleq \mathrm{c}'$, by Lemma 7.2.8 (no dependants of c are duplicates). Then no cut in Y depends on c, and $\mathrm{c} \vartriangle_{\mathrm{B}} \mathrm{y}$ for some $\mathrm{y} \in Y$, a contradiction. □

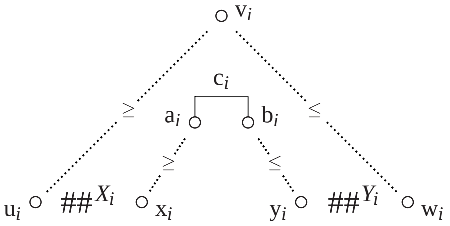
It remains to be shown that for strongly safe proof forests the (transitively closed) combination of the dependency and straddling forms a partial order.

Lemma 7.2.11. *In a strongly safe forest* $(\leq \cup \vartriangle)^{*}$ *is antisymmetric.*

Proof. Consider a series of dependencies and straddlings.

$$\mathrm{c}_0 \leq \mathrm{v}_1 \vartriangle \mathrm{c}_1 \leq \ldots \leq \mathrm{v}_n \vartriangle \mathrm{c}_n = \mathrm{c}_0 \,.$$

For $1 \leq i \leq n$, let each cut c_i have children a_i and b_i, and let $\mathrm{v}_i \vartriangle \mathrm{c}_i$ be witnessed as follows: $\mathrm{v}_i \leq \mathrm{u}_i, \mathrm{w}_i$, while $\mathrm{u}_i \,\#\#^{C_i}\, \mathrm{w}_i$ with $C_i = X_i \cup \{\mathrm{c}_i\} \cup Y_i$, as illustrated below.



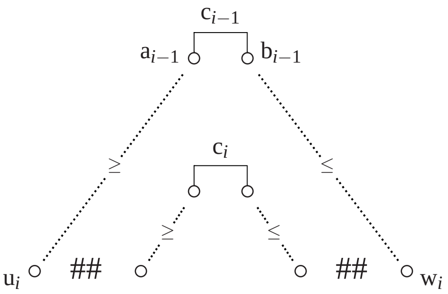
In the case $n = 1$ there are the dependencies $\mathrm{v}_1 \vartriangle \mathrm{c}_1$ and $\mathrm{c}_1 \leq \mathrm{v}_1$. Then either $\mathrm{a}_1 \leq \mathrm{v}_1$ or $\mathrm{b}_1 \leq \mathrm{v}_1$; w.l.o.g. assume the latter. Via the following separations and dependencies this gives $\mathrm{u}_1 \,\#\#\, \mathrm{u}_1$, a contradiction.

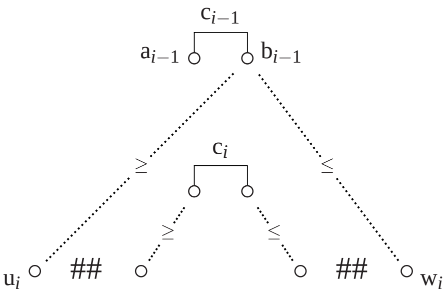
$$\mathrm{u}_1 \;\; \#\#^{X_1} \;\; \mathrm{x}_1 \;\; \#\#^{\{\mathrm{c}_1\}} \;\; \mathrm{b}_1 \;\leq\; \mathrm{v}_1 \;\leq\; \mathrm{u}_1$$

For the general case, since $\mathrm{c}_{i-1} \leq \mathrm{v}_i \leq \mathrm{u}_i, \mathrm{w}_i$, either $\mathrm{c}_{i-1} \vartriangle \mathrm{c}_i$ or $\mathrm{v}_{i-1} \leq \mathrm{c}_{i-1} \leq \mathrm{c}_i$; in this latter case, the cycle may be shortened by skipping c_{i-1} and v_i. Then assume the given cycle, reproduced below, is the shortest in $\vartriangle$ over the cuts $\mathrm{c}_1 \ldots \mathrm{c}_n = \mathrm{c}_0$; in particular, $\mathrm{c}_j \in C_i$ if and only if $i = j$, since otherwise $\mathrm{c}_{j-1} \vartriangle \mathrm{c}_i$.

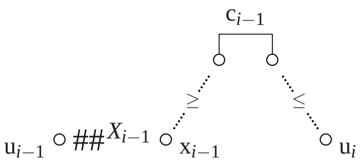
$$\mathrm{c}_0 \;\vartriangle\; \mathrm{c}_1 \;\vartriangle\; \ldots \;\vartriangle\; \mathrm{c}_n \;=\; \mathrm{c}_0$$

Next, in $\mathrm{c}_{i-1} \vartriangle \mathrm{c}_i$ the vertices u_i and w_i, which depend on c_{i-1}, must each depend on a_{i-1} or b_{i-1}. If one depends on a_{i-1} and the other on b_{i-1}, as illustrated below left, then $\mathrm{u}_i \,\#\#^{C_i \cup \{\mathrm{c}_{i-1}\}}\, \mathrm{u}_i$ (by Lemma 7.2.7 it may be assumed that $\mathrm{c}_{i-1} \notin C_i$) contradicting strong safety.





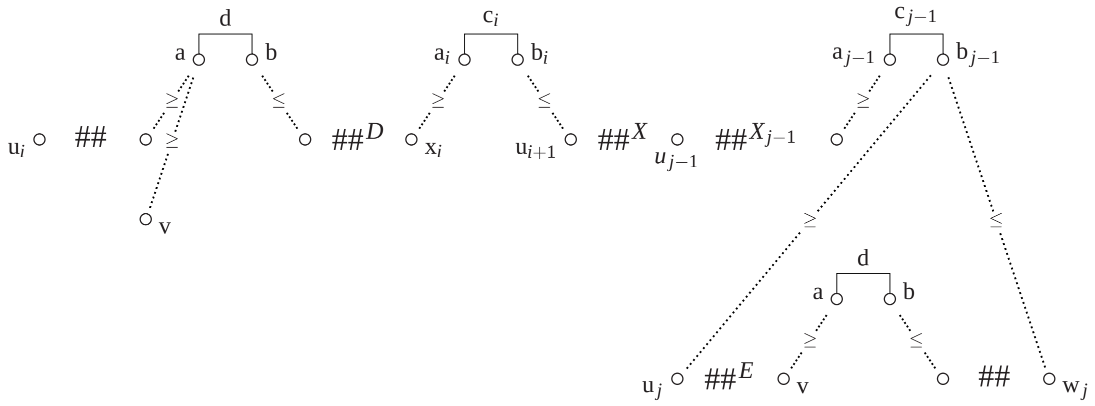
Without loss of generality let both u_i and w_i depend on b_i, for all $1 \leq i \leq n$, as illustrated above right. Then $\mathrm{u}_{i-1} \,\#\#\, \mathrm{u}_i$, as illustrated below, and $\mathrm{u}_n \,\#\#\, \mathrm{u}_1$.



This gives the following sequence.

$$\mathrm{u}_1 \quad \#\#^{X_1\cup\{\mathrm{c}_1\}} \quad \mathrm{u}_2 \quad \#\#^{X_2\cup\{\mathrm{c}_2\}} \quad \ldots \quad \#\#^{X_n\cup\{\mathrm{c}_n\}} \quad \mathrm{u}_1$$

To conclude that $\mathrm{u}_1 \; \#\# \; \mathrm{u}_1$, the sets $X_i \cup \{\mathrm{c}_i\}$ must be disjoint. Recall that $\mathrm{c}_i \notin X_j$ unless $i = j$. To show that also all sets X_i are disjoint, let X_i and C_j be such that $X_i \cap C_j \neq \varnothing$ with $i < j$, and i is the greatest index for which this holds, i.e. there is no $i < k < j$ with $X_k \cap C_j \neq \varnothing$. In X_i, illustrated below, from left to right let d be the last cut in X_i that also appears in C_j; i.e. let $D \subset X_i$ be disjoint from C_j, let d have children a and b, and let $\mathrm{a} \; \#\#^{D\cup\{\mathrm{d},\mathrm{c}_i\}} \; \mathrm{b}_i$.



Let v be a vertex such that $\mathrm{a} \leq \mathrm{v}$ and either $\mathrm{u}_j \; \#\#^E \; \mathrm{v}$ or $\mathrm{w}_j \; \#\#^E \; \mathrm{v}$ (the former is used in the illustration above), where $\mathrm{d} \notin E \subset C_j$. Then $\mathrm{v} \; \#\#^Z \; \mathrm{v}$, where

$$Z \;=\; \{\mathrm{d}\} \,\cup\, D \,\cup\, \{\mathrm{c}_i\} \,\cup\, X \,\cup\, X_{j-1} \,\cup\, \{\mathrm{c}_{j-1}\} \,\cup\, E \qquad\qquad X \;=\; \bigcup_{i<k<j-1} X_k \cup \{\mathrm{c}_k\}$$

—by construction, all components of Z as listed in the equations are disjoint. □

Finally, the previous lemmata combined allow weak normalisation to be proved.

Theorem 7.2.12 (Weak normalisation without pruning)**.** *For any strongly safe forest* $\mathrm{F_A}$ *there is a finite reduction path* $\mathrm{F_A}\ (\rightsquigarrow^-)^*\ \mathrm{F_C}$ *such that* $\mathrm{F_C}$ *is cut-free.*

Proof. Let $\mathrm{F_A}$ be strongly safe, and not cut-free. Select a cut c that does not straddle others and has no dependent cuts of same or higher complexity; i.e.

$$\forall \mathrm{d} \in \mathrm{V_A}(\bot).\ \mathrm{c} \not\vartriangle \mathrm{d} \,\wedge\, \big(\mathrm{c} < \mathrm{d} \;\Rightarrow\; \mathit{compl}(\mathrm{c}) > \mathit{compl}(\mathrm{d})\big)\ .$$

Such a cut c must exist, since the relation $(\leq \cup \vartriangle)^*$ on $\mathrm{F_A}$ is a partial order, by Lemma 7.2.11. Then let $\mathrm{F_A} \overset{\mathrm{c}}{\rightsquigarrow}{}^- \mathrm{F_B}$. By Lemma 7.2.10 the proof forest $\mathrm{F_B}$ is strongly safe, and its complexity is smaller than that of $\mathrm{F_A}$. By induction, $\mathrm{F_B}(\rightsquigarrow^-)^*\mathrm{F_C}$, and the statement follows. □

Since compound reduction steps consist of finitely many ordinary reduction steps, the following is immediate.

Corollary 7.2.13. *The relation* $(\rightsquigarrow)$ *is weakly normalising.*

7.3 Discussion and related work

The closest relatives of classical proof forests are Miller's expansion tree proofs [79], and the Herbrand nets investigated by McKinley [74]. This section will illustrate how these formalisms relate to proof forests, via a discussion of a number of modifications to various aspects of proof forests. These modifications include alternative correctness criteria, a further variation on reduction steps, and the addition of *tautology links*. A further variation, found in the literature, is the fragment of proof forests that disallows contraction and weakening [80]. Since the presence of contraction is a primary reason for the complexity of reductions in proof forests, this variant has a much better behaved cut-elimination procedure. However, the discussion here will be restricted to related formalisms in which contraction does occur, and the full complexities of classical logic are present.

Expansion tree proofs

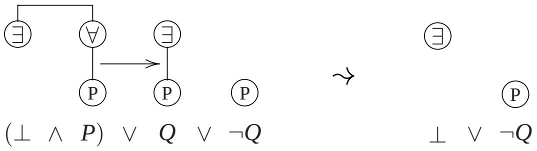
Two main distinctions between proof forests and Miller's expansion tree proofs as presented in [79] need no further explanation: expansion tree proofs allow higher-order and non-prenex formulae, but have not been given a normalisation procedure. A minor point is that expansion tree proofs require existential nodes to have at least one branch. A third, important difference is in the correctness criteria employed.

The correctness condition for expansion tree proofs used in [79], when stated for proof forests, assigns two formulae to a forest: the *shallow* formula and the *deep* formula. The shallow formula of a proof forest is the sequent formed by the labels of its root nodes. The deep formula is the propositional formula obtained by interpreting the trees in a proof forest as propositional formula trees, where the branching at an existential node is interpreted as disjunction, and cuts are treated as conjunctions. The shallow and deep formula of a propositional leaf node coincide, and the deep formula of a universal node is the same as that of its child. A proof forest would then be correct if its deep formula is a propositional tautology.

This correctness criterion corresponds directly to that of the Herbrand proofs by Buss [20], discussed in Section 5.2; for that reason, call it *Herbrand correctness*. Compared to the actual correctness criterion used in proof forests, of Definition 5.4.6, Herbrand correctness is equivalent to a switching condition that ignores the dependency. That is, a proof forest is Herbrand correct if for every possibility of deleting one child of each cut node, and recursively deleting its children (but not necessarily its dependants), the disjunction over the remaining propositional nodes is a tautology.

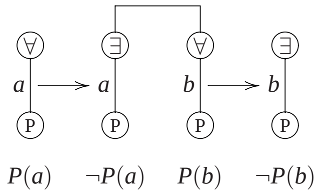
Several of the operations on proof forests used in normalisation do not preserve Herbrand correctness. First and foremost, Herbrand correctness is not preserved by the pruning operation. Given that pruning is essential to the modified reduction relation, and that reductions need to preserve correctness, the current correctness condition is a crucial component of the weak normalisation result in Theorem 6.4.9.

Disposal reduction steps also do not preserve Herbrand correctness, as is illustrated below: while the deep formula on the left is a tautology, the one on the right is not. The deep formula of a weakened existential node—which does not occur in expansion tree proofs—is taken to be $\bot$, the empty disjunction.



Operations that do preserve Herbrand correctness are composition via cut, and propositional, logical, and structural reduction steps. For the weak normalisation of $(\rightsquigarrow)$, Corollary 7.2.13, Herbrand correctness may then replace the correctness criterion of Definition 5.4.6, provided weakening (existential nodes without edges) is disallowed.

One drawback of Herbrand correctness is that it does not allow the current translation of the cut, from proof forests to sequent proofs, as presented in Section 5.5. The translation is easily amended: the two subproofs of a cut are the proof forests obtained by removing each of the two trees below the cut, ignoring the dependency. However, as the example in Figure 7.3 demonstrates, this introduces free variables into the sequent proof. These are eigenvariable occurrences whose universal introduction rule has been removed—and, technically, now resides in the other subproof. In the example a proof forest is translated according to the natural translation procedure supported by Herbrand correctness. The eigenvariable occurrence b on the rightmost edge in the proof forest is in the left subproof of the sequent proof no longer an occurrence of the eigenvariable b, which is introduced only in the right subproof. Note that although in this example, to obtain the subproofs for the sequent translation the proof forest could



$$\dfrac{\dfrac{\dfrac{\vdash \quad P(a), \quad \neg P(a), \quad \neg P(b)}{\vdash \forall x.P(x), \exists x.\neg P(x), \exists x.\neg P(x)}{\scriptstyle \forall\mathrm{R},\exists\mathrm{R}} \qquad \dfrac{\vdash \quad P(a), \quad P(b), \quad \neg P(b)}{\vdash \forall x.P(x), \forall x.P(x), \exists x.\neg P(x)}{\scriptstyle \forall\mathrm{R},\exists\mathrm{R}}}{\vdash \forall x.P(x), \exists x.\neg P(x), \forall x.P(x), \exists x.\neg P(x)}{\scriptstyle \mathrm{Cut}}}{\vdash \forall x.P(x), \exists x.\neg P(x)}{\scriptstyle \mathrm{CR}}$$

Figure 7.3: Translation based on Herbrand correctness

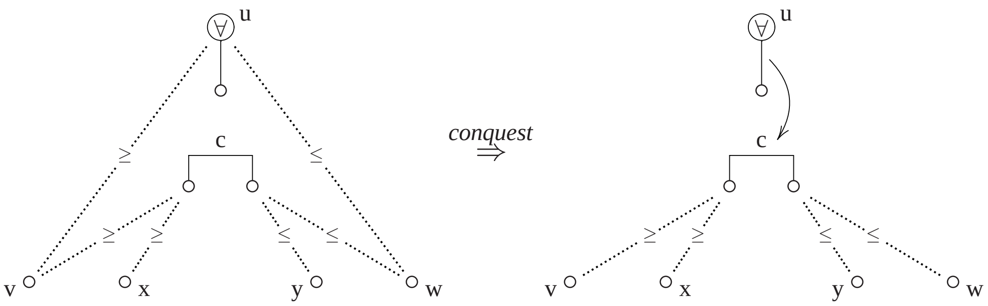
simply be split through the middle, in general the deep formula would be of the form $A \vee (B \wedge C) \vee D$, which only allows to conclude $A \vee B \vee D$ and $A \vee C \vee D$.

Other modifications

In Section 7.2 the straddling relation was defined (see Definition 7.2.6), to account for the impermutability of cuts with inferences that have premises or eigenvariable occurrences in both subproofs. Seeing that one interpretation of the dependency in proof forests is as an account of impermutability in the sequent calculus, it is natural to ask whether straddling could be incorporated into the dependency. Furthermore, straddling and strong safety are related to a multiplicative interpretation of the cut, whereas safety, correctness and pruning are related to an additive interpretation, as was argued in Section 5.5. In this light, a natural question is whether the notion of strong safety allows the construction of a translation from proof forests to sequent proofs that is inverse to $[\![-]\!]$. Pursuing these ideas, below, leads to a range of subtle modifications to the calculus of classical proof forests—some of which are more, some less semantically meaningful.

It was established in Section 7.2 that the only way that reductions cause the loss of strong safety is by a structural step on a cut c that straddles another cut d. The idea behind the first question, of incorporating straddling into the dependency, is that strong safety would not be lost if d were dependent on c instead. That straddling is a relation between nodes, while in proof forests the dependency is generated by the relation $(\rightarrow)$ from universal edges to existential edges and cut edges, is not an objection, because the loss of strong safety is caused by the duplication of the dependants of the universal branch of c; it is then sufficient to make the cut d dependent on the universal side of the cut c.

Implementing this idea, reductions can be augmented with the following *conquest*step, applied after each reduction step: for every universal edge $\langle u, w\rangle$ and every cut edge $\langle \bot, c\rangle$, if $u \vartriangle c$, add the dependency $\langle u, w\rangle \to \langle \bot, c\rangle$ to the proof forest. It is immediate from Lemma 7.2.11 that, for a strongly safe proof forest, the dependency remains antisymmetric after a conquest step, and it is not too difficult to see that strong safety itself is preserved. However, correctness is not preserved, as can be seen in the following example.



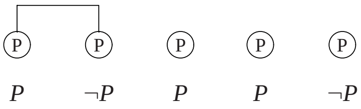
A switching that switches off the node u, switches off all four of v, w, x and y after the conquest step, but only v and w before, while c switches off only one of x and y. If the above configurations are part of a proof forest F on the left, and F′ on the right, with x and y propositional vertices, ς a switching that switches off u, and Γ is the value of F′ under the switching ς, then the value of F under ς is $\Gamma \vee \mathit{lab}(x)$ or $\Gamma \vee \mathit{lab}(y)$, depending on the choice of ς on c. For correctness to be preserved, it should be that $\Gamma \vee \mathit{lab}(x)$ and $\Gamma \vee \mathit{lab}(y)$ together imply Γ. This is not in general the case, because there is no obligation for $\mathit{lab}(x)$ and $\mathit{lab}(y)$ to be each other's negation.

However, dependencies are ignored in Herbrand correctness, which is therefore trivially preserved in conquest steps. The following is then put forward as a conjecture, for the informal nature of the arguments supporting its preservation properties.

Conjecture 7.3.1. *The reduction relation* $(\rightsquigarrow)$ *supplemented with conquest steps, on strongly safe, Herbrand correct proof forests without weakened existential nodes, is weakly normalising.*

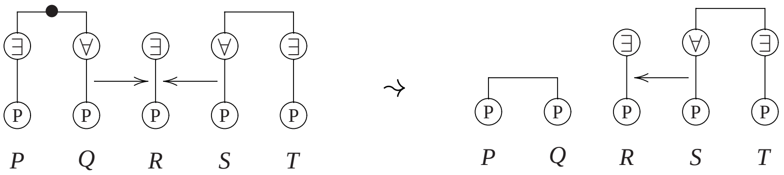
As weak normalisation for $(\rightsquigarrow)$ was proven by Corollary 7.2.13, conquest steps are redundant in the above conjecture. It is mentioned for the reason that the calculus with conquest is expected to have stronger normalisation properties—perhaps even strong normalisation if conquest is applied eagerly—than the calculus without. Still, the calculus described in this conjecture is an odd combination of semantically only tenuously related concepts: reduction steps plus conquest are based on the multiplicative cut in sequent calculus, while Herbrand correctness is based on a direct interpretation of Herbrand's Theorem. From this point, a natural direction to investigate is towards a correctness criterion that allows a purely multiplicative interpretation of the cut in proof forests.

This is precisely the question of a correctness condition that allows an inverse translation to $[\![-]\!]$. First and foremost, strictly speaking it is impossible to obtain such an inverse without adding additional structure to proof forests, by the following example.



There are two different sequent proofs that translate to this proof forest—and these are not equal up to permutations. Still, the example leaves open the possibility of a translation that is the inverse of $[\![-]\!]$ up to propositional contractions and tautology links—which is reasonable, given that propositional content is (supposed to be) ignored in both calculi. To find such a notion, an obvious direction follows the idea that the notion of separation (Definition 7.2.1) indicates, for a proof forest translated from a sequent proof, which vertices originated in different subproofs of the sequent proof. It is tempting to use separation to give a notion of *strong correctness*, by analogy to the way conflict may be used to define correctness, as was done in Proposition 6.4.2. In this notion, a proof forest would be strongly correct if for every maximal separation-free subset of V the labels over the propositional vertices form a tautology.

However, this notion of strong safeness does not bring the desired inverse translation procedure much closer, and moreover suffers from the fatal problem that it is not preserved under logical reduction steps. This follows from the example below.



On the left, under strong correctness there are the following three tautologies: $P \vee S$, $Q \vee T$, and $Q \vee R \vee S$. Crucially, the values under correctness are these three plus $P \vee T$; however, since the two outermost propositional vertices are separated, this is not one of the tautologies of strong correctness. Then on the right, the tautologies for both strong correctness and correctness are the following four: $P \vee T$, $P \vee R \vee S$, $Q \vee T$, and $Q \vee R \vee S$. Whereas correctness allows all four to be proved from the tautologies of the proof forest on the left, strong correctness cannot prove that $P \vee T$ is a tautology. The above also illustrates that 'strong correctness' is not an appropriate name, since for the proof forest on the left, it does not imply correctness.

A final modification of proof forests will be discussed, one that is more invasive than the previous, but perhaps not as much as it initially seems. The idea is to add the tautology rule of the sequent calculus of Figure 5.4 to proof forests as a *tautology link*, by analogy with the axiom links of MLL-nets. This is the direction taken by Richard McKinley, resulting in the Herbrand nets discussed below.

Herbrand nets

Like proof forests, the Herbrand nets developed by McKinley [74] are aimed at providing a canonical representation of first-order classical proof by removing bureaucracy, and share the same basic forest structure. Different from proof forests, in Herbrand nets the sequent calculus is taken as primary, and in particular the axiom rule—or in the first-order case, the tautology rule—is considered to contribute to the essential proof content. By adding *tautology links* corresponding to the tautology rule of Figure 5.4, Herbrand nets provide a notion of proof net for a first-order sequent calculus similar to the strict calculus in Figure 5.4 plus cut—specifically, it includes propositional contraction, but not existential weakening.

The technical distinctions between the two formalisms can mostly be ascribed to two properties, required of Herbrand nets in order to be a reasonable notion of proof net: one, translation from nets to sequent proofs should be invertible up to permutations; two, this translation should commute with reductions in either formalism. To achieve invertible translation the main ingredient, and the main distinction with proof forests, is invertible composition.

The tautology links of Herbrand nets are reminiscent of the axiom links found in other forms of proof net, but connect several propositional nodes that, by taking the disjunction over their labels, form a tautology. In addition, a propositional node can participate in multiple tautology links, which corresponds to contraction on propositional formulae in the sequent calculus. Tautology links are implemented as special vertices without parents or children, indexed by natural numbers, to which propositional nodes connect via pointers. These pointers are then incorporated into the dependency. Figure 7.4 illustrates a Herbrand net with two tautology links.

The correctness criterion of Herbrand nets is a Danos–Regnier-style switching condition [29], familiar from MLL nets, on the dependency graph of a forest. A switch-

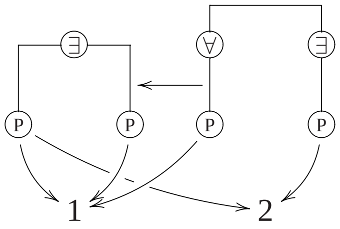


Figure 7.4: Herbrand nets are proof forests with tautology links

ing chooses: one edge of each existential node; one tautology link for each propositional node; and for each universal node u either its unique edge or one connection $\langle u, w\rangle \rightarrow \langle x, y\rangle$ from u to y. For each switching, after removing other such connections not chosen by the switching, the remaining graph is required to be connected and acyclic.

Reduction steps in Herbrand nets, which need to preserve this correctness condition, differ from those in proof forests in several respects. Omitting weakening, there is no equivalent to disposal steps in Herbrand nets, but logical steps in both formalisms are identical, modulo the presence of tautology links. Propositional steps in Herbrand nets unify two axiom links, in the way that is obvious from the reduction step in the sequent calculus, described in Section 6.2. A propositional cut with a child that connects to two or more links, rather than inducing a duplication from the implicit propositional contraction, is left unreduced pending the unification of these links.

A structural step in Herbrand nets duplicates the *kingdom* of its universal child. The notion of kingdom (see [13]) originated in the study of proof nets for multiplicative linear logic. There, and likewise in Herbrand nets, the kingdom of a node is the smallest subnet of which it is a root—where a subnet is a subgraph that is a proof net—and corresponds to the smallest possible subproof under permutations in a sequent proof. Kingdoms in Herbrand nets are precisely dependent subforests after a conquest step, i.e. after straddling is incorporated into the dependency—where the dependency differs from that of proof forests by including the pointers of tautology links.

To summarise, Herbrand nets are proof forests with tautology links, with reduction steps comparable to proof forest reduction with conquest. In [74] it is demonstrated that Herbrand nets have invertible composition and invertible translation with sequent proofs, and weakly normalising reduction steps that commute with those of the sequent calculus.

7.4 Non-confluence

In formalisms that respect the symmetry of classical logic, it is common for proof reduction to exhibit non-confluence. It is therefore perhaps not a great surprise that reduction in proof forests, too, is non-confluent. Nonetheless, it is interesting to look at the way non-confluence occurs here, firstly, because of the canonical nature of forests. A consequence of the strict focus on witness assignment to quantifiers in proof forests, while propositional content is ignored, is that reducing propositional cuts is trivially confluent—which means that any non-confluence in proof forests is due entirely to first-order proof content. In addition, forests are *polarised* in the sense that contraction and weakening are only applied to existential formulae. Thus two notorious sources of non-confluence in the standard sequent calculus, a cut on two weakenings and a cut on two contractions, are avoided (see e.g. [92, Appendix B1], [42], or [68]).

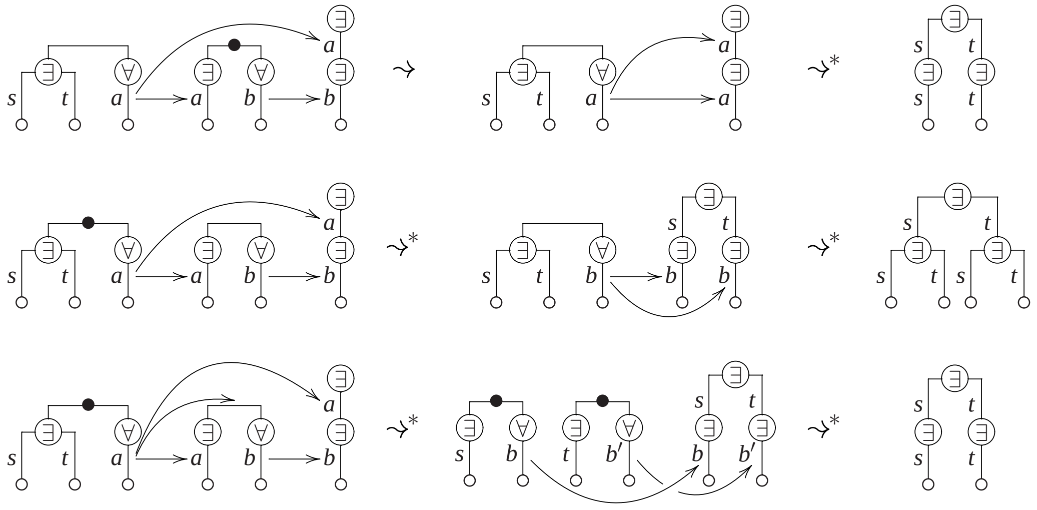


Figure 7.5: An example of non-confluence, made confluent by conquest

A second reason why non-confluence in proof forests is interesting is the way it appears in the universal counterexample, which may be seen as an instance of the familiar problem of two contractions interacting via cuts, rephrased for the current context of proof forests. Although many other examples of non-confluence exist, most are sensitive to modifications in the reduction relation. For example, in Figure 7.5 the first two reduction paths yield different normal forms, while a conquest step would modify the first forest to become that in the third reduction path, making the example confluent. (The particular example in Figure 7.5 is due to Richard McKinley.)

A second example, in Figure 7.6, is confluent in normal reduction $(\rightsquigarrow)$, but as shown

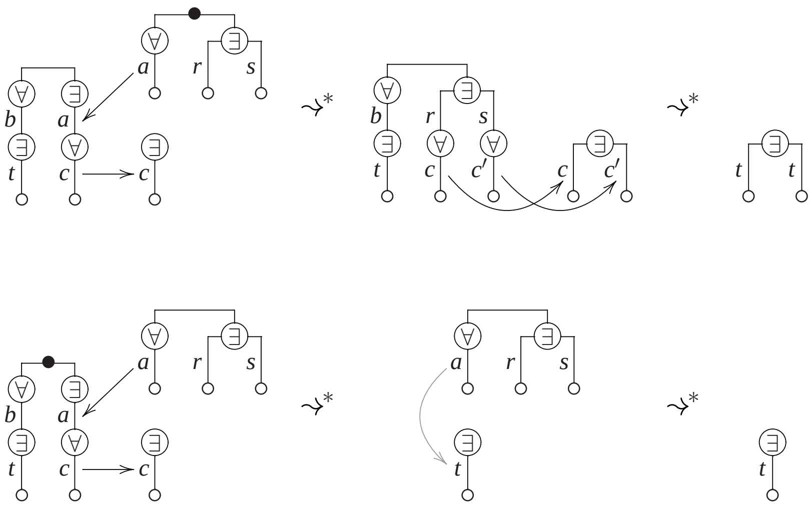


Figure 7.6: Non-confluence with minimal dependencies (where $b \notin fv(t)$)

becomes non-confluent when reduction is interleaved with minimisation (replacing the dependency with the minimal one, see Section 5.4). In the second reduction path of the example, the grey arrow results from the reduction steps, but is not part of the minimal dependency, since b is not free in t. The reduction also shows that minimality of the dependency is not preserved in reductions.

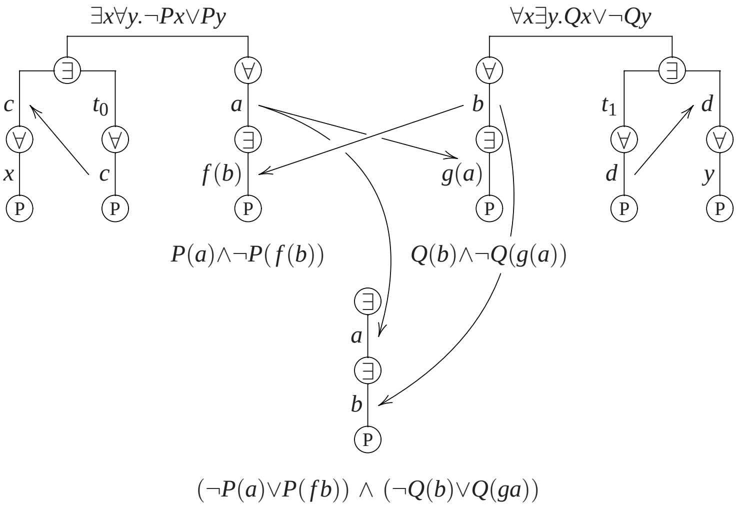
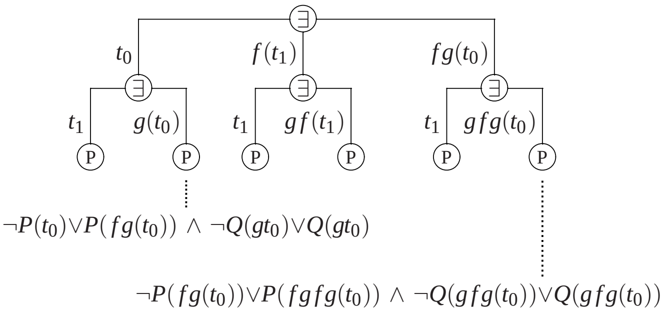


Figure 7.7: The universal counterexample in a context

The universal counterexample is the simplest example found that is non-confluent under any modification of the reduction relation described here—including reduction in Herbrand nets, a fact that is demonstrated in [74, Section 8]. In Figure 7.7 the universal counterexample is put within a context, omitting some (easily inferred) labels to prevent clutter. The example is a correct proof forest for the formula

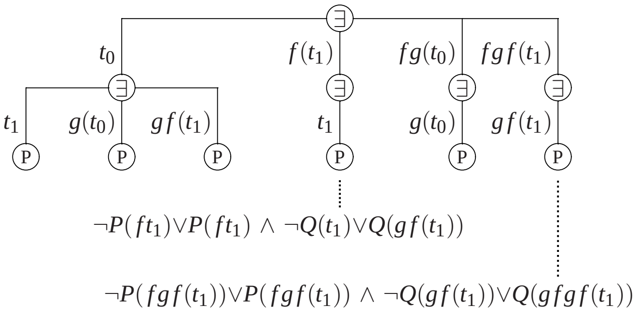
$$\exists xy.\ (\neg P(x) \vee P(fy))\ \wedge\ (\neg Q(y) \vee Q(gx))\ .$$

In the regular reduction relation $(\rightsquigarrow)$ the proof forest in Figure 7.7 has exactly two normal forms. The one below is the result of the reduction path in Figure 6.6, and of any other path that fully reduces the cut on the right before reducing that on the left.


∃
t_0
$f(t_1)$
$fg(t_0)$
∃
∃
∃
t_1
$g(t_0)$
t_1
$gf(t_1)$
t_1
$gfg(t_0)$
P
P
P
P
P
P
$\neg P(t_0) \vee P(fg(t_0)) \wedge \neg Q(gt_0) \vee Q(gt_0)$
$\neg P(fg(t_0)) \vee P(fgfg(t_0)) \wedge \neg Q(gfg(t_0)) \vee Q(gfg(t_0))$

The labels that are indicated are the two that are necessary for the correctness of the above proof forest. As in both labels the Q-atoms cancel out, the dual atoms $P(fg(t_0))$ and $\neg P(fg(t_0))$ make the disjunction over the propositional labels a tautology.

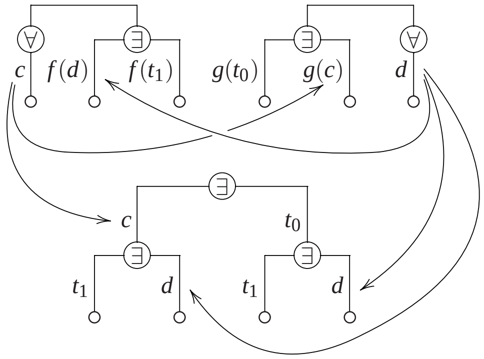
The other normal form of the proof forest in Figure 7.7, shown below, is reached by any reduction path that fully reduces the left cut before reducing the right cut.


∃
t_0
$f(t_1)$
$fg(t_0)$
$fgf(t_1)$
∃
∃
∃
∃
t_1
$g(t_0)$
$gf(t_1)$
t_1
$g(t_0)$
$gf(t_1)$
P
P
P
P
P
P
$\neg P(ft_1) \vee P(ft_1) \wedge \neg Q(t_1) \vee Q(gf(t_1))$
$\neg P(fgf(t_1)) \vee P(fgf(t_1)) \wedge \neg Q(gf(t_1)) \vee Q(gfgf(t_1))$

This time, the P-atoms cancel out, revealing dual atoms $\neg Q(gf(t_1))$ and $Q(gf(t_1))$. The difference between the two normal forms is thus not simply a matter of having different redundant existential branches: the two normal forms are perfectly symmetric, and provide symmetric solutions to the problem posed by the formula.

Much the same holds under the modifications to the reduction relation that were discussed. In the modified reduction relation $(\Rrightarrow)$ (Definition 6.4.6), the reduction paths that reduce one of both cuts fully before reducing the other are still available. In addition, the paths that interleave reduction steps in either cut, such as that in Figure 6.10, are normalising in $(\Rrightarrow)$ due to pruning. When reducing the proof forest in

Figure 7.7 via the reduction path Figure 6.10 the symmetry of the proof forest is preserved right up until the moment that pruning is needed—this point in the reduction is illustrated below.



From this point there are two possible reduction steps in $(\mathrel{\approx\!\!\!>})$, both involving pruning, that after one more step lead to two normal forms, each similar to one of the normal forms described above but with additional existential branches. Neither simultaneously contains both pairs of dual atoms of the other normal forms, $\neg Q(gf(t_1))$ and $Q(gf(t_1))$, and $P(fg(t_0))$ and $\neg P(fg(t_0))$.

Finally, reducing the example in Figure 7.7 in $(\rightsquigarrow)$ augmented with conquest would have the two initial reduction steps shown in Figure 7.8. From there, the example would reduce as in $(\rightsquigarrow)$, to reach one of the two normal forms described. In different contexts, however, using a conquest step in a reduction may lead to a different normal form than a normalising path in $(\rightsquigarrow)$ without conquest.

The non-confluence exhibited by these examples seems fundamental, and it looks improbable that simple modifications can make reduction confluent.

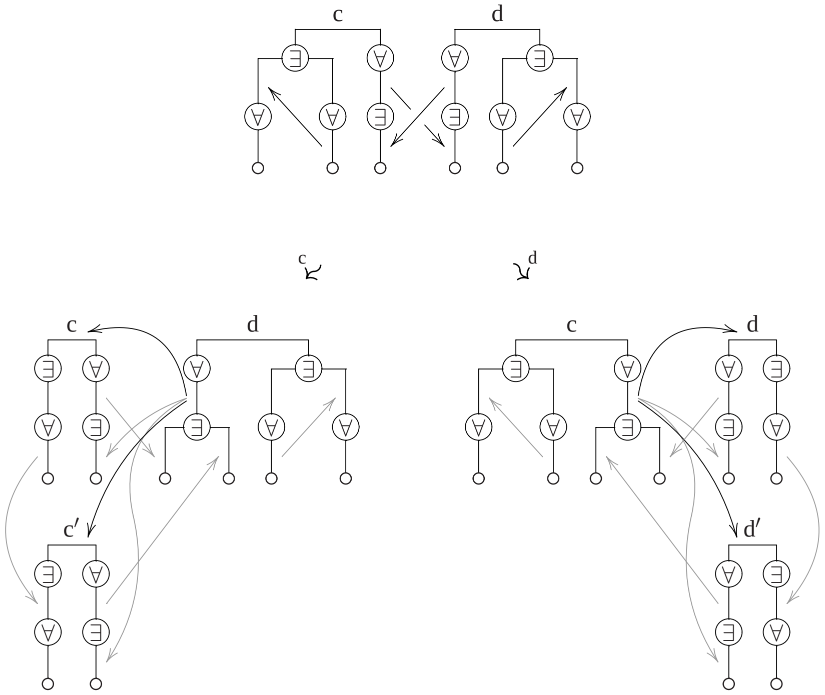


Figure 7.8: The universal counterexample with conquest

Chapter 8

Conclusions

8.1 Summary

In this dissertation two canonical representations of proof were discussed. The present section will briefly summarise the motivations for, and the results of the work on these formalisms, presented in the previous chapters. Section 8.2 will suggest angles for future investigations.

Proof nets for additive linear logic

In Part I a new notion of proof net was presented, that is canonical for additive linear logic with units. As was argued in the introductory chapter (Section 1.3), this logic is a simple but rich fragment of linear logic, that exhibits many of the problems with composition in syntactic representations of linear logic (cf. [2] and [64]). Its semantics is that of bi-Cartesian or sum–product categories, categories with finite products and coproducts, which are ubiquitous throughout mathematics. Additive linear logic forms a term calculus for such categories, whose equational theory was described in [25]; an effective (polynomial-time) decision procedure was given recently in [23]. Still, the problem of finding proof nets for this fragment, canonical representations for the morphisms of free sum–product categories, remained unsolved. As set out in Chapter 2, earlier proof nets by Hughes and Van Glabbeek [59], here presented (in slightly modified form) as sum–product nets, were not canonical for the units.

The main contribution of Part I of the dissertation are the saturated nets that are canonical representations of proof in additive linear logic. The theory of saturated nets, as presented here, covers all the essential notions required of proof nets, including a correctness condition, a sequentialisation algorithm, and a direct definition of composition (i.e. not via translations with the sequent presentation, or even unsaturated sum–product nets). Saturated nets themselves are characterised, in Chapter 3, in three ways:

- as the result of saturating sum–product nets—when combined with the translation from proof terms to sum–product nets, this comprises the direct translation from proofs in additive linear logic to saturated nets;
- as the union over an equivalence class of sum–product nets;
- as the prenets that satisfy the correctness criterion of Proposition 3.4.5.

The characterisation of free sum–product categories was completed by the description of composition as relational composition plus saturation, in Section 3.3. Finally, saturation constitutes an efficient decision procedure, by translating (cut-free) proof terms to saturated nets to compare these for syntactic equality. Both saturation itself, and the decision procedure it enables, operate in linear time in the product of the sizes of the source and target formulae.

The central technical contribution behind the results of Part I is the proof presented in Chapter 4, that the decision procedure of comparing saturated nets is sound for free sum–product categories.

Classical proof forests

Part II of the dissertation discussed a canonical proof formalism for first-order classical logic, called ‘classical proof forests’, and presented an investigation into composition via cut-elimination for this formalism. The motivations for this work and for the design of the proof forests may be summarised as follows. To find a good notion of proof identity for propositional classical proof is problematic. At the same time, Herbrand’s Theorem shows that propositional proof, being decidable, can be ignored in a formalism for first-order proof. This allows an approach to canonical proof that finds the essential content of first-order classical proof in the assignment of witnessing terms to the quantifiers. Such a route has been taken before: by Miller in [79], to find an efficient representation for higher-order classical proof, and by Coquand in [26], to give a semantics for classical arithmetic based on games. For the present work, a main motivation was the idea that a canonical representation of first-order proof based on witnessing information might support a good notion of composition.

This representation of proof, classical proof forests, was introduced in Chapter 5. Two main views of classical proof forests were discussed in this chapter. Firstly, they describe a natural notion of strategy for a two-player backtracking game. As a strategy for ∃loise, a proof forest prescribes the moves to be made by ∃loise, and their dependence on moves by ∀belard; but no further order on moves is forced. A second view, detailed in Section 5.5, compares classical proof forests to a first-order sequent calculus. The fact that classical proof forests factor out the bureaucracy of permutations in the sequent presentation, is a main reason why they may be considered canonical. However, it is also shown that in the presence of cuts, the correspondence between dependency in proof forests and impermutability in the sequent calculus is not exact.

Mainly due to the divergence between these concepts, reduction steps in proof forests behave differently from reduction in the sequent calculus, even though, in spirit, reduction steps in both formalisms are comparable. In Chapter 6 this leads to the puzzling fact that, from a perfectly acceptable proof forest called the ‘universal counterexample’, reduction steps produce cuts that are *unsafe*. These are cuts where both sides have common dependants, an unnatural configuration which prevents them from being reduced. Fortunately, the correctness condition for proof forests, based on the game-theoretic interpretation, allows the shared dependants to simply be removed, in an operation called *pruning*. Next, by grouping reduction steps together, the one known cause of infinite reduction paths is prevented from occurring. For the modified reduction relation $(\mathrel{\approx\!\!>})$, which implements these two solutions, a weak normalisation theorem is proven, and strong normalisation is conjectured.

In Chapter 7 a *strong safety* property was defined, closely related to the property of being the translation of a sequent proof, implying the absence of unsafe cuts in a forest. A careful analysis of the universal counterexample, and the difference between the dependency of proof forests and impermutability in the sequent calculus, then led to the identification of a class of reduction steps in forests that preserve strong safety. By showing that at least one such a reduction step must apply if a proof forest has a cut, weak normalisation was shown for the original reduction relation $(\rightsquigarrow)$. The remainder of Chapter 7 was split between a discussion of alternative modifications to the reduction relation, including an exploration of related work, and an overview of non-confluence in proof forest reductions.

The two approaches to weak normalisation, one via pruning and one via strong safety, can be viewed as corresponding to the two different interpretations of proof forests, as strategies in backtracking games and as an abstraction over the sequent calculus. In some respects, the modified reduction relation $(\rightsquigarrow)$ is the more natural solution: it is simpler, it has a single, uniform first-order reduction step instead of three different ones, and pruning is naturally suggested by the game interpretation of the cut. In contrast, while the approach via strong safety succeeds in finding an interpretation of the cut in proof forests that corresponds more closely to the (multiplicative) cut of the sequent calculus, it seems that, to capture the full behaviour of the sequent calculus, axiom links as in McKinley's Herbrand nets are indispensable. That being said, the characterisation of a multiplicative interpretation of the cut in the absence of axiom links, by strong safety and the *straddling* relation, is undoubtedly of interest.

8.2 Further work

Further work on sum–product nets

This section presents a brief list of possible angles for future work based on proof nets for additive linear logic. The presentation of saturated nets in this dissertation leaves few open questions on saturated nets themselves—though one such problem is listed first, below. Nonetheless, there are numerous interesting areas for future investigations that take saturated nets as their starting point.

A simpler soundness proof The soundness proof of saturation as a decision procedure, in Chapter 4, is long and complex, especially given the simplicity of the saturation algorithm. Surely it must be possible to give a simpler proof that saturated nets are canonical. The current presentation reflects the order in which the results were obtained, and for that reason it is unlikely to be optimal. In particular, the correctness condition for saturated nets was found last. It is plausible that its proof, in Section 4.9, could lead to a simpler proof of the canonicity of saturated nets.

Bicomplete categories Since products and coproducts are discrete limits and colimits, a natural question is whether sum–product nets and saturated nets can be adapted to characterise the free completion with all finite limits and colimits. This would require to include equalisers and co-equalisers, on top of the existing machinery. A notion of proof nets along these lines would be canonical for Joyal's bicomplete categories [63], restricted to finite limits and colimits.

Infinite products and coproducts One extension of the present work would be to investigate canonical representations of categories with infinite products and coproducts. A direct, finite graphical depiction of such infinite objects is of course impossible, but it would be interesting, and possibly even useful, to see whether, abstractly, canonical forms are possible. Combined with the above suggestion of proof nets for finite limits and colimits, the question of canonical representations could even be extended to all limits and colimits.

Games semantics of additive linear logic Games semantics is an important branch of research on linear logic. While fully complete models for linear logic are known ([77]), as was mentioned in Section 1.3 it has proven hard to move away from the alternating, interleaving approach, towards a true concurrency approach. It is to be expected that a game-theoretic interpretation of saturated nets will be a useful contribution towards this goal.

Completeness as a model for EEC The enriched effect calculus of Egger, Møgelberg, and Simpson [35] is a promising model of computation, that incorporates a rich theory of computational effects. Its term calculus operates in two domains, one of *values* and one of *computations*. Its models consist of a category of computations, enriched in a category of values, with an adjunction between the two categories. The sum–product completion of the empty set $\Sigma\Pi(\varnothing)$, which is enriched in the category of finite sets, forms a family of basic such EEC-models (differing only in the choice of adjunction). An interesting question is whether such models are complete for the enriched effect calculus.

Proof nets for linear logic The search for canonical proof representations is a fundamental, long-standing open problem in full classical linear logic. It is hoped that saturated nets will prove a useful contribution towards solving this problem. In addition, while the techniques in this dissertation are quite specific, it is hoped that the general ideas and overall approach will prove to be useful in the search for proof nets for all of linear logic.

Further work on classical proof forests

The treatment of classical proof forests in this dissertation leaves a few open questions, the most important of which is the strong normalisation conjecture for the modified reduction relation. This, and several other angles for future research, are listed below.

Strong normalisation for the modified reduction relation The main open question in this dissertation is the strong normalisation conjecture for the modified reduction relation $(\Rightsquigarrow)$, Conjecture 6.4.10. Despite some effort, attempts to apply existing techniques to this problem, most notably the approach in [93], were unsuccessful.

Conquest reductions A focal point in the discussion of alternative modifications to the reduction relation is the possibility of *conquest* steps in reductions, steps that include straddling into the dependency. This approach is yet to be formalised, and although weak normalisation is almost immediate from the weak normalisation of $(\rightsquigarrow)$, it is quite plausible that a reduction relation employing conquest steps could be strongly normalising.

Infinite normal forms One drastic approach to obtaining confluence would be the following. Consider a process where cuts are duplicated before they are reduced, retaining both the original cut and the reduced one in the proof forest. It is plausible that in the limit, this process is confluent, producing an infinite ‘proof forest’ that contains, at least, the results of all genuine reduction paths to normal forms. Then cuts can be removed, leaving an infinite normal form. Although the appeal of such a formalism would be mainly theoretical, there are interesting questions to be considered, for example on confluence, and on composition of infinite normal forms. In addition, if such infinite normal forms can be finitely represented, for example by a grammar or automaton, they may also be of practical interest.

Computational content of witness assignment The computational meaning of cut-elimination in proof forests can be seen as lying in the changes to the witnessing information, effected during the normalisation process. A clear example of this is given by the normal forms of the reduction of the universal counterexample in Section 7.4. Naturally, such computation occurs in other formalisms for first-order classical logic, too. Nevertheless, because of their canonicity, it would be interesting to see how classical proof forests can be employed computationally. A more specific question in this direction is whether classical proof forests can simulate computation in first-order intuitionistic proof normalisation.

Bibliography

[1] Samson Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[2] Samson Abramsky. Sequentiality vs. concurrency in games and logic. *Mathematical Structures in Computer Science*, 13(4):531–565, 2003.

[3] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.

[4] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 1996.

[5] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LiCS'99)*, 1999.

[6] José Bacelar Almeida, Jorge Sousa Pinto, and Miguel Vilaça. A local graph-rewriting system for deciding equality in sum-product theories. *Electronic Notes in Theoretical Computer Science*, 176:139–163, 2007.

[7] Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29(2):149–177, 2000.

[8] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Believe it or not, AJM's games model is a model of classical linear logic. *Proc. 12th Annual IEEE Symposium on Logic in Computer Science (LiCS'97)*, pages 68–75, 1997.

[9] Franco Barbanera and Stefano Berardi. A strong normalization result for classical logic. *Annals of Pure and Applied Logic*, 76(2):99–116, 1995.

[10] Michael Barr. *-Autonomous categories and linear logic. *Mathematical Structures in Computer Science*, 1:159–178, 1991.

[11] Gianluigi Bellin, Martin Hyland, Edmund Robinson, and Christian Urban. Categorical proof theory of classical propositional logic. *Theoretical Computer Science*, 364(2):146–165, 2006.

[12] Gianluigi Bellin and Philip Scott. On the pi-calculus and linear logic. *Theoretical Computer Science*, 135:11–65, 1994.

[13] Gianluigi Bellin and Jacques van de Wiele. Subnets of proof-nets in MLL^{-}. In *Advances in Linear Logic*, pages 249–270, 1995.

[14] Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. Linear lambda-calculus and categorical models revisited. In *Proc. 6th EACSL Annual Conference on Computer Science Logic (CSL'92)*, pages 61–84, 1993.

[15] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56:183–220, 1992.

[16] Richard Blute. Proof nets and coherence theorems. In *Category Theory and Computer Science*, volume 530 of *Lecture Notes in Computer Science*, pages 121–137. Springer Berlin / Heidelberg, 1991.

[17] Richard Blute, Robin Cockett, Robert Seely, and Todd Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113:229–296, 1996.

[18] Richard Blute, Masahiro Hamano, and Philip Scott. Softness of hypercoherences and MALL full completeness. *Annals of Pure and Applied Logic*, 131(1–3):1–63, 2005.

[19] Kai Brünnler. *Deep Inference and Symmetry in Classical Proofs*. PhD thesis, Technische Universität Dresden, 2004.

[20] Samuel R. Buss. On Herbrand's Theorem. *Lecture Notes in Computer Science*, 960:195–209, 1995.

[21] Iliano Cervesato and Andre Scedrov. Relating state-based and process-based concurrency through linear logic. *Information and Computation*, 207(10):1044–1077, 2009.

[22] Robin Cockett and Craig Pastro. The logic of message passing. *Science of Computer Programming*, 74:498–533, 2009.

[23] Robin Cockett and Luigi Santocanale. On the word problem for ΣΠ-categories, and the properties of two-way communication. In *Proc. 18th EACSL Annual Conference on Computer Science Logic (CSL'09)*, volume 5771, pages 194–208, 2009.

[24] Robin Cockett and Robert Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, pages 133–173, 1997.

[25] Robin Cockett and Robert Seely. Finite sum-product logic. *Theory and Applications of Categories*, 8(5):63–99, 2001.

[26] Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, 1995.

[27] Haskell Brooks Curry and Robert Feys. *Combinatory Logic, Vol. I*. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1958.

[28] Vincent Danos, Jean-Baptiste Joinet, and Harold Schellinx. A new deconstructive logic: Linear logic. *Journal of Symbolic Logic*, 62:755–807, 1997.

[29] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[30] Nicolaas Govert de Bruijn. AUTOMATH, a language for mathematics. Technical Report T.H.-Report 68-WSK-05, Department of Mathematics, Technological University Eindhoven, 1968. Reprinted in *Automation and Reasoning, vol. 2, Classical papers on computational logic 1967–1970*, Springer Verlag, 1983, pages 159–200.

[31] Harish Devarajan, Dominic Hughes, Gordon Plotkin, and Vaughan Pratt. Full completeness of the multiplicative linear logic of Chu spaces. *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LiCS'99)*, pages 234–243, 1999.

[32] Paolo di Giamberardino and Claudia Faggian. Proof nets sequentialisation in multiplicative linear logic. *Annals of Pure and Applied Logic*, 155(3):173–182, 2008.

[33] Kosta Došen and Zoran Petrić. Bicartesian coherence. *Studia Logica*, 71(3):331–353, 2002.

[34] Kosta Došen and Zoran Petrić. *Proof-Theoretical Coherence*, volume 1 of *Studies in Logic*. King's College Publications, London, 2004.

[35] Jeff Egger, Rasmus Møgelberg, and Alex Simpson. Enriching an effect calculus with linear types. In *Computer Science Logic 2009*, volume 5771 of *Lecture Notes in Computer Science*, pages 240–254, 2009.

[36] Thomas Ehrhard. Hypercoherences: a strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3:365–385, 1993.

[37] Claudia Faggian and Mauro Piccolo. Partial orders, event structures and linear strategies. In *Proc. 9th International Conference on Typed Lambda Calculi and Applications (TLCA'09)*, volume 5608 of *Lecture Notes in Computer Science*, pages 95–111, 2009.

[38] Gilda Ferreira and Paolo Oliva. On various negative translations. In *Proc. 3rd International Workshop on Classical Logic and Computation*, volume 47 of *Electronic Proceedings in Theoretical Computer Science*, pages 22–33, 2010.

[39] Carsten Führmann and David Pym. Order-enriched categorical models of the classical sequent calculus. *Journal of Pure and Applied Algebra*, 204:21–78, 2006.

[40] Gerhard Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934–1935. English translation in: The Collected Papers of Gerhard Gentzen, M.E. Szabo (ed.), North-Holland 1969.

[41] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[42] Jean-Yves Girard. A new constructive logic: Classical logic. *Mathematical Structures in Computer Science,* 1:255–296, 1991.

[43] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Logic and Algebra,* pages 97–124, 1996.

[44] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[45] Timothy Griffin. A formulae-as-type notion of control. In *Proc. 17th ACM Symposium on Principles of Programming Languages*, pages 47–58, 1990.

[46] Stefano Guerrini. Correctness of multiplicative proof nets is linear. In *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LiCS'99)*, pages 454–463, 1999.

[47] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical Methods in Computer Science*, 4(1:9):1–36, 2008.

[48] Willem Heijltjes. Classical proof forestry. *Annals of Pure and Applied Logic*, 161(11):1346–1366, 2010.

[49] Willem Heijltjes. Proof nets for additive linear logic with units. In *Proc. 26th Annual IEEE Symposium on Logic in Computer Science (LiCS'11)*, pages 207–216, 2011.

[50] Jacques Herbrand. Investigations in proof theory: The properties of true propositions. In Jean van Heijenoort, editor, *From Frege to Gödel: A source book in mathematical logic, 1879–1931*, pages 525–581. Harvard University Press, 1967.

[51] Stefan Hetzl and Alexander Leitsch. Proof transformations and structural invariance. *Lecture Notes in Computer Science*, 4460:201–230, 2007.

[52] William A. Howard. The formulae-as-types notion of construction. In Jonathan P. Seldin and J. Roger Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, Boston, MA, 1980.

[53] Hongde Hu. Contractible coherence spaces and maximal maps. *Electronic Notes in Theoretical Computer Science*, 20:1–11, 1999.

[54] Hongde Hu and Andre Joyal. Coherence completions of categories and their enriched softness. *Electronic Notes in Theoretical Computer Science*, 6, 1997.

[55] Hongde Hu and Andre Joyal. Coherence completions of categories. *Theoretical Computer Science*, 227:153–184, 1999.

[56] Dominic Hughes. A canonical graphical syntax for non-empty finite products and sums. Technical report, Stanford University, 2002.

[57] Dominic Hughes. Simple free star-autonomous categories and full coherence. Technical report, Stanford University, 2005.

[58] Dominic Hughes. Proofs without syntax. *Annals of Mathematics*, 164(3):1065–1076, 2006.

[59] Dominic Hughes and Rob van Glabbeek. Proof nets for unit-free multiplicative-additive linear logic. *ACM Transactions on Computational Logic*, 6(4), 2005.

[60] Martin Hyland. Abstract interpretation of proofs: Classical propositional calculus. *Lecture Notes in Computer Science*, 3210:1–16, 2004.

[61] Martin Hyland and Luke Ong. Fair games and full completeness for multiplicative linear logic without the mix-rule. Unpublished manuscript, 1993.

[62] Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

[63] Andre Joyal. Free bicomplete categories. *C.R. Math. Rep. Acad. Sci. Canada*, XVII(5):219–224, 1995.

[64] Andre Joyal. Free lattices, communication and money games. *Proc. 10th Int. Cong. of Logic, Methodology and Philosophy of Science*, 1995.

[65] Thong Wei Koh and Luke Ong. Explicit substitution internal languages for autonomous and *-autonomous categories. *Electronic Notes in Theoretical Computer Science*, 29, 1999.

[66] Yves Lafont and Thomas Streicher. Games semantics for linear logic. *Proc. 6th Annual IEEE Symposium on Logic in Computer Science (LiCS'91)*, pages 43–51, 1991.

[67] François Lamarche and Lutz Straßburger. Constructing free Boolean categories. *Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LiCS'05)*, pages 209–218, 2005.

[68] François Lamarche and Lutz Straßburger. Naming proofs in classical propositional logic. *Lecture Notes in Computer Science*, 3461:246–261, 2005.

[69] Joachim Lambek and Philip Scott. *Introduction to higher order categorical logic*. Cambridge University Press, 1986.

[70] Sam Lindley. Extensional rewriting with sums. In *Proc. 8th International Conference on Typed Lambda Calculi and Applications (TLCA'07)*, pages 255–271, 2007.

[71] Saunders Mac Lane. *Categories for the working mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1998.

[72] Richard McKinley. *Categorical Models of First-Order Classical Proofs*. PhD thesis, University of Bath, 2006.

[73] Richard McKinley. Expansion nets: proof nets for propositional classical logic. *Logic Programming and Automated Reasoning*, 2010.

[74] Richard McKinley. Proof nets for Herbrand's Theorem. arXiv:1005.3986v1, 2010.

[75] Paul-André Melliès. Asynchronous games 3: An innocent model of linear logic. In *Proc. 10th Conference on Category Theory and Computer Science*, pages 1–21, 2004.

[76] Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. *Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LiCS'05)*, pages 386–395, 2005.

[77] Paul-André Melliès. Asynchronous games 2: The true concurrency of innocence. In *Selected Papers of CONCUR 2004*, volume 358 of *Theoretical Computer Science*, pages 200–228, 2006.

[78] Paul-André Melliès and Samuel Mimram. Asynchronous games: innocence without alternation. In *Proc. CONCUR 2007*, Lisboa, 2007.

[79] Dale A Miller. A compact representation of proofs. *Studia Logica*, 46(4):347–370, 1987.

[80] Samuel Mimram. The structure of first-order causality. *Mathematical Structures in Computer Science*, 21(1):65–110, 2011.

[81] Hanno Nickau. Hereditarily sequential functionals. *Lecture Notes in Computer Science*, pages 253–264, 1994.

[82] Michel Parigot. $\lambda\mu$-Calculus: an algorithmic interpretation of classical natural deduction. *Lecture Notes in Computer Science*, 624:190–201, 1992.

[83] Dag Prawitz. *Natural deduction. A Proof-theoretical Study*. Almqvist & Wiksell, Uppsala, 1965.

[84] Dag Prawitz. Ideas and results in proof theory. In J E Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 235–306, 1971.

[85] Edmund Robinson. Proof nets for classical logic. *Journal of Logic and Computation*, 13(5):777–797, 2003.

[86] Robert Seely. Linear logic, *-autonomous categories and cofree coalgebras. *Contemporary Mathematics*, 92, 1989.

[87] Peter Selinger. Control categories and duality. *Mathematical Structures in Computer Science*, 11:207–206, 2001.

[88] Lutz Straßburger. On the axiomatisation of Boolean categories with and without medial. *Theory and Applications of Categories*, 18:536–601, 2007.

[89] Lutz Straßburger. *Towards a Theory of Proofs of Classical Logic*. Habilitation à diriger des recherches, École Polytechnique, Palaiseau, 2010.

[90] Lutz Straßburger and François Lamarche. On proof nets for multiplicative linear logic with units. *Proc. 13th EACSL Annual Conference on Computer Science Logic (CSL'04)*, pages 145–159, 2004.

[91] Thomas Streicher and Bernhard Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, 1998.

[92] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1996.

[93] Christian Urban and Gavin Bierman. Strong normalisation of cut-elimination in classical logic. *Fundamenta Informaticae*, 45(1-2):123–155, 2001.

[94] Jan Von Plato and Gerhard Gentzen. Gentzen's proof of normalization for natural deduction. *The Bulletin of Symbolic Logic*, 14(2):240–257, 2008.

[95] Philip Wadler. Linear types can change the world! In M. Broy and C. Jones, editors, *Programming Concepts and Methods*. North-Holland Publ. Co., Amsterdam, 1990.

[96] Glynn Winskel. *Events in Computation*. PhD thesis, University of Edinburgh, 1980.

Index